Ontological interpretation of network monitoring data

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/15762](https://dspace.lboro.ac.uk/2134/15762)

Publisher: © Ian Napier

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: [https://creativecommons.org/licenses/by-nc-nd/4.0/](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Please cite the published version.
Ontological Interpretation of Network Monitoring Data

by

Ian Douglas Napier

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of

Loughborough University

20th August 2014

Copyright 2014 Ian Douglas Napier
Abstract

Interpreting measurement and monitoring data from networks in general and the Internet in particular is a challenge. The motivation for this work has been to investigate new ways to bridge the gap between the kind of data which are available and the more developed information which is needed by network stakeholders to support decision making and network management. Specific problems of syntax, semantics, conflicting data and modeling domain-specific knowledge have been identified. The methods developed and tested have used the Resource Description Framework (RDF) and the ontology languages of the Semantic Web to bring together data from disparate sources into unified knowledgebases in two discrete case studies, both using real network data. Those knowledgebases have then been demonstrated to be usable and valuable sources of information about the networks concerned. Some success has been achieved in overcoming each of the identified problems using these techniques, proving the thesis that taking an ontological approach to the processing of network monitoring data can be a very useful technique for overcoming problems of interpretation and for making information available to those who need it.
Acknowledgements

During the period of my Ph.D. studies, I have experienced a number of significant life-events, some good and some not so good. The birth of a lovely baby daughter, the decline in health and subsequent death of a much loved Mother, a major road accident and a move to a house, requiring far too much refurbishment work, in a new and unfamiliar town being a few. Although these have all caused some disruption to my work, Loughborough University in general and my supervisor Dr Iain Phillips in particular have been patient, understanding and supportive throughout. I would particularly like to thank my second supervisor Dr Lin Guan for her wise advice early on, Mrs Judith Poulton for her friendly and efficient assistance, willingly given whenever needed and Dr Iain Phillips for his guidance, encouragement and friendship during my time at Loughborough, as well as his ability to give a sense of perspective when needed over a coffee or the occasional beer. I would also like to thank Mr Martyn Bright of TRM Ltd for his assistance with data gathering and for sharing his knowledge of real networks and Dr Olaf Maennel, who along with Dr Iain Phillips provided me with comprehensive Internet topology data which was vital to this work.

At home I have had the unstinting support of my family. Without it I could never have completed this work. My son Andrew and daughter Alex have both believed in me and helped me in their different ways throughout this time. My Mom Edna always believed I could achieve anything I put my mind to, which always helped me while she was with me and still does now that she is resting in a better place. My wonderful wife Heidi has made countless sacrifices, without ever once complaining, so that I could enjoy the privilege of being a student at a stage in our married life when it simply would not have been an option for most people. She has patiently helped me and encouraged me, always knowing the right thing to say. Her loving support has sustained me often when things seemed too difficult. The home-made soup, blueberry muffins and constant stream of tea have been pretty good too!

Thanks to you all.
Contents

Abstract 2
Acknowledgements 3

1 Introduction 11
  1.1 Report Structure .............................................. 11
  1.2 History ....................................................... 12
  1.3 Internet Usage Problems .................................... 13
  1.4 Ontology Modelling .......................................... 13
  1.5 Research Aims and Scope ................................... 14
  1.6 Research Contribution Summary ............................ 16

2 Internet Monitoring Issues 17
  2.1 Chapter Introduction ......................................... 17
  2.2 Motivation for Performing Internet Monitoring and Measurement 17
  2.3 Introduction to Measurements of Interest .................. 18
  2.4 Monitoring Methods and Problems .......................... 20
  2.5 Measurement Categories .................................... 21
  2.6 Research Groups and Projects .............................. 22
  2.7 Data Repositories ........................................... 24
  2.8 Chapter Summary ............................................ 26

3 An Ontological Approach 27
  3.1 Chapter Introduction ......................................... 27
  3.2 The Interpretation Problem .................................. 27
  3.3 Introduction to Ontology .................................... 30
  3.4 Semantic Web Background ................................... 32
  3.5 The Resource Description Framework ....................... 34
  3.6 Ontology Languages ......................................... 36
  3.7 Chapter Summary ............................................ 39
## 4 Related Work

4.1 Chapter Introduction ................................................. 41
4.2 CYC ................................................................. 41
4.3 FOAF ................................................................. 42
4.4 Multi-perspective Ontologies for Logistics ......................... 42
4.5 Computer Network Management Studies ............................ 43
4.6 Chapter Summary ..................................................... 44

## 5 Design and Methodology

5.1 Chapter Introduction ................................................ 46
5.2 Approach Summary .................................................. 46
5.3 Building Blocks and Frameworks ................................... 48
5.4 Development Environment ........................................ 51
5.5 Jena ................................................................. 52
5.6 The Reference Ontologies .......................................... 54
5.7 General Methodology Description ................................. 56
5.8 The Case Studies ..................................................... 58
5.9 Chapter Summary ..................................................... 59

## 6 Reference Ontology Implementation

6.1 Chapter Introduction ................................................ 62
6.2 Reference Ontology Concept and Content .......................... 62
6.3 The LIOL Organisation Autonomous Ontology ..................... 64
6.4 The LIOL Protocols Autonomous Ontology ......................... 67
6.5 The LIOL Traffic Autonomous Ontology ........................... 70
6.6 The LIOL Infrastructure Autonomous Ontology ................... 73
6.7 The LIOL Applications Autonomous Ontology ..................... 76
6.8 The LIOL Monitoring Autonomous Ontology ....................... 78
6.9 Chapter Summary ..................................................... 78

## 7 The VPN Case Study

7.1 Chapter Introduction ................................................ 81
7.2 Background and Motivation ........................................ 81
7.3 The Organisation A Network ....................................... 83
7.4 Methodology ......................................................... 85
7.5 Data Sources and Collection ....................................... 86
7.6 Data Converters ...................................................... 88
7.7 Assessment of the Conversion Process ............................. 93
7.8 Local Linking Ontology ............................................. 100
7.9 Application and Testing .............................................. 104
CONTENTS

7.10 Rules .................................................. 110
7.11 Case Study Summary and Review .......................... 112

8 The AS Topology Case Study ............................. 114
  8.1 Chapter Introduction .................................. 114
  8.2 Background and Motivation ............................ 114
  8.3 Methodology and Data Sources ........................ 116
  8.4 Data Conversion Analysis .............................. 118
  8.5 The AS-Level Ontology ................................. 119
  8.6 Final Conversion and Testing ........................... 123
  8.7 Case Study Summary and Review ........................ 125

9 Conclusions, Contribution and Future Direction ........... 128

A Industrial Partner Meeting Notes ......................... 140
  A.1 TRML Meeting ........................................ 140

B Programming Framework Selection ........................ 142
  B.1 Assessment Criteria .................................... 142
  B.2 Redland .............................................. 143
  B.3 RDFLib .............................................. 144
  B.4 Sesame ............................................... 144
  B.5 Jena .................................................. 145
  B.6 Conclusions .......................................... 145

C LIOL RDF/XML Listings .................................. 147
  C.1 The Organisation AO RDF/XML Listing .................. 147
  C.2 The Protocols AO RDF/XML Listing .................... 148
  C.3 The Traffic AO AO RDF/XML Listing ................... 149
  C.4 The Infrastructure AO RDF/XML Listing ............... 150
  C.5 The Applications AO RDF/XML Listing ................ 152
  C.6 The Monitoring AO RDF/XML Listing ................... 152

D Local Ontology Listing .................................. 154
  D.1 VPN Study Local Ontology RDF/XML Listing .......... 154

E AS Level Ontology Listing ................................ 159
  E.1 AS Level Ontology RDF/XML Listing ................... 159

F Key to Methodology Diagrams ............................ 161

G Key to Ontology Diagrams ............................... 162
# List of Figures

2.1 The Three Internet Elements ............................................. 18

3.1 Illustrating the need for Knowledge-Based Applications to interface between stakeholders and existing data and tools .................. 30
3.2 Simplified Semantic Web “Layer Cake” ................................. 34
3.3 RDF: 3-Dimensional Space Concept ...................................... 35
3.4 RDF: Descriptive vs Non-Descriptive Labelling ......................... 37
3.5 A basic networking ontology ............................................. 38

5.1 The General Methodology ................................................. 57
5.2 Case Studies Relevance to Research Problems ......................... 59

6.1 The LIOL Organisation Autonomous Ontology .......................... 64
6.2 Organisation AO Protégé Test ........................................... 66
6.3 Organisation AO Inference Test ........................................... 66
6.4 The LIOL Protocols Autonomous Ontology ............................ 67
6.5 Protocols AO Protégé Test .............................................. 69
6.6 Protocols AO Inference Test .............................................. 69
6.7 The LIOL Traffic Autonomous Ontology ............................... 70
6.8 Traffic AO Protégé Test ................................................ 72
6.9 Traffic AO Inference Test ................................................. 72
6.10 The LIOL Infrastructure Autonomous Ontology ....................... 73
6.11 Infrastructure AO Protégé Test ......................................... 75
6.12 Infrastructure AO Inference Test ........................................ 75
6.13 The LIOL Applications Autonomous Ontology ....................... 76
6.14 Applications AO Protégé Test ........................................... 77
6.15 Applications AO Inference Test ......................................... 77
6.16 The LIOL Monitoring Autonomous Ontology .......................... 78
6.17 Monitoring AO Protégé Test ............................................. 80
6.18 Monitoring AO Inference Test ........................................... 80

7.1 The Organisation A Network ............................................. 83
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>The VPN Study Methodology</td>
<td>85</td>
</tr>
<tr>
<td>7.3</td>
<td>Packet-Level Conversion Rate as a Function of Triplestore Size</td>
<td>97</td>
</tr>
<tr>
<td>7.4</td>
<td>Flow-Level Conversion Rate as a Function of Triplestore Size</td>
<td>97</td>
</tr>
<tr>
<td>7.5</td>
<td>The Organisation A Local Linking Ontology</td>
<td>101</td>
</tr>
<tr>
<td>7.6</td>
<td>The Organisation A Local Ontology Process</td>
<td>105</td>
</tr>
<tr>
<td>8.1</td>
<td>The AS Topology Study Methodology</td>
<td>117</td>
</tr>
<tr>
<td>8.2</td>
<td>The AS Level Ontology</td>
<td>121</td>
</tr>
<tr>
<td>F.1</td>
<td>A key to the Methodology Process Diagrams</td>
<td>161</td>
</tr>
<tr>
<td>G.1</td>
<td>A Key to the Ontology Diagrams</td>
<td>162</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Semantic Web Framework Comparison Summary . . . . . . . . . . 50

7.1 PCAP to RDF Conversion Data . . . . . . . . . . . . . . . . . . 94
7.2 PCAP to RDF Conversion Times . . . . . . . . . . . . . . . . . . 96
7.3 Packet-Level Query Times . . . . . . . . . . . . . . . . . . . . . . 99

8.1 Traceroute and BGP dump to AS path Conversion . . . . . . . . . . 119
8.2 AS Number to RDF Conversion . . . . . . . . . . . . . . . . . . . 127
8.3 AS Links by Ontology Class View . . . . . . . . . . . . . . . . . . 127
Listings

7.1 Packet-level PCAP to RDF Conversion snippet .......................... 89
7.2 Flow-level PCAP to RDF Conversion snippet .......................... 91
7.3 Example Branch Office Configuration ................................. 102
7.4 Key Statements from the Local Linking Ontology ................... 104
8.1 AS-level Ontology Code .............................................. 122
C.1 The Organisation AO RDF/XML Listing .............................. 147
C.2 The Protocols AO RDF/XML Listing ................................ 148
C.3 The Traffic AO RDF/XML Listing .................................... 149
C.4 The Infrastructure AO RDF/XML Listing ............................. 150
C.5 The Applications AO RDF/XML Listing .............................. 152
C.6 The Monitoring AO RDF/XML Listing ............................... 152
D.1 The VPN Study Local Linking Ontology RDF/XML Listing ....... 154
E.1 The AS Case Study Ontology RDF/XML Listing ...................... 159
Chapter 1

Introduction

1.1 Report Structure

This research has involved bringing tools and techniques developed in the field of Knowledge Management and the Semantic Web to bear on existing problems of interpretation of network measurement and monitoring data. The structure of this report reflects that cross-disciplinary approach. This introductory chapter gives brief a historical background to these two threads and lays out the reasons why bringing them together was judged to be both important and appropriate as a research exercise.

In Chapter 2, some background research on the Internet monitoring issues of interest is presented. The chapter explains the motivation for measuring and monitoring, discusses the various types of measurements, elaborating on the problems of both obtaining and subsequently interpreting them. It then goes on to review some of the important research initiatives and organisations active in the field. Various existing data gathering tools and available datasets are also discussed here.

Chapter 3 begins by expanding on the problems of data interpretation identified in Chapter 2 and then separating these out to some extent from the problems of initially obtaining basic data. These problems are then categorised. This chapter introduces the concept of Ontology and then builds on that to explain the origins of the Semantic Web, its principal mark-up languages and tools.

In Chapter 4, some related work is described and its influence on the direction of this research is explained. The related work described ranges from an early general ontology through ontologies which contain knowledge specific to networks
in the general sense, culminating in some specific cases in the computer networking domain.

Chapter 5 gives a detailed description of the practical approach taken and methodology adopted in this research. The overall design is explained here, with some detailed comparison of the available tools and frameworks. The reasons for selecting those used in this particular application are stated. The chapter also introduces some key new concepts conceived and developed as part of the research, which are expanded in subsequent chapters. These are the Loughborough Internetworking Ontology Library (LIOL) and the real-world case studies which were undertaken.

Chapter 6 further explains the concept presented of an ontology library for the networking domain. The Loughborough Internetworking Ontology Library (LIOL) which was developed as part of this project is justified and described in detail using symbolism which was also created within the project. In each case validation results are shown.

Chapters 7 and 8 give detailed descriptions of the VPN case study and the AS topology case study respectively, including evaluation work. These case studies were the principal means to test the practicality and utility of the ontological approach developed.

Chapter 9 gives a detailed evaluation of the results from the case studies and brings the threads from these together to give a comprehensive conclusion to the research as a whole, with answers to the original research questions laid out in Section 1.5.

1.2 History

The evolution of the United States Government ARPANET of the 1970s to the global Internet of 2013 has opened up many new possibilities and created Internet stakeholders, whose businesses or organisations depend on this network of independently managed networks. This evolution is graphically illustrated by the Hobbes’ Internet Timeline [71]. The Internet is now of major importance to society and it’s use for business applications and e-commerce has increased significantly in recent years. For example, the United Kingdom Government Office of National Statistics (ONS) produced a statistical bulletin in December 2013 [18] which shows that in 2012, 95% of businesses in the UK had broadband Internet connections, and 82% had a web presence. In the same year, overall e-commerce sales as a
proportion of total sales reached 18% . The same report asserts that the most popular method of connection is by Digital Subscriber Line (DSL), accounting for 86% of business connections. With this increase in the use of and dependence on the Internet for non-trivial applications for which it was not originally intended, particularly by small and medium sized enterprises, come a number of problems in the areas of network management, monitoring and performance measurement. The interconnected nature of business, which has been made possible by the Internet, necessarily creates more complexity in local networks and company intranets. The complexities involved also mean that interpretation and understanding of performance related data is increasingly difficult.

1.3 Internet Usage Problems

Use of the Internet for e-commerce, inter-site data communications and other business critical applications inevitably results in some loss of control when compared to more traditional communication methods, such as using leased line private circuits from a single national telecommunications company. This is in part because neither the end user nor the application provider have full visibility of the routes taken by their vital data and must rely on the best effort, end-to-end philosophy on which the Internet is built [60]. They may well have no contractual relationship with the network providers on whom they depend for transporting their packets and may not even know who those operators are. This is a consequence of the decentralised, commercial nature of the Internet, which relies on private peering arrangements and policies. Internet stakeholders include businesses, ISPs, governments and others. They can experience difficulties with harnessing available network monitoring data to give much needed visibility of the structure and performance of those parts of the Internet of direct relevance and indeed their own internal networks and intranets. The difficulties encountered when measuring the performance of the Internet are explored more fully in Chapter 2. The original motivation for this research was to find new ways to address these problems and give useful information to decision makers and network administrators.

1.4 Ontology Modelling

In 2001 an article by Berners-Lee, Hendler and Lassilla entitled “The Semantic Web” was published in Scientific American magazine [36]. The article gave wide visibility to the notion of extending the World Wide Web (WWW) to give more meaningful, machine readable links using concepts which already existed in the fields of graph theory and description logic. This idea has been taken forward by
the World Wide Web consortium (W3C), who have produced standard specifications for the languages used within this area. At the heart of this technology are the Resource Description Framework (RDF), RDF Schema (RDFS), the Web Ontology Language (OWL) and various related tools for reasoning and querying. These are described more fully in Chapters 3 and 5. Although the original motivation for the development of these languages was the Semantic Web (or Web 3.0 as it is sometimes known), these exact same languages can also be applied to data modeling projects not necessarily related to the www, by the building and exploitation of suitable ontologies. This divergence was emphasised as a very important development by James Hendler, (one of the co-authors of the referenced article [36]), in a lecture series he ran at DeMontford University, UK, between 30th March and 1st April 2009, which influenced the direction of the work presented here. Three functions which such modeling is designed to perform and which appear to offer useful solutions to the problems in the area of networking generally and the Internet in particular have been chosen for evaluation. These are:

**Embedding expertise** specific to the networking domain and to the specific local conditions, with a view to reducing the level of human diagnosis of any pathologies present, and reducing repetition. For example, it may be possible to assert in an ontology that packet loss in a particular network segment is likely to be caused by failure of a particular router. This knowledge is thus embedded once and does not need to be repeatedly discovered by humans.

**Combining disparate data** from multiple sources, some public (general) and some private (network specific) into a common format. A fundamental requirement here is to facilitate data sharing and overcome both syntactical and semantic inconsistencies. For example, the results of a locally-conducted traceroute probe may be linked to a publicly available IP address to AS number mapping database to give useful routing information.

**Mediating variability** between different data sources. This can be apparent when attempting to derive a set of results by different investigative methods. For example, different methods for discovering network topologies may yield conflicting results which could be brought together with some adjudication to give a single consistent set of results.

### 1.5 Research Aims and Scope

The large-scale use of the Internet by both commercial and non-commercial organisations has resulted in a level of complexity which causes problems in performance
monitoring and network management. Some of these problems concern the interpretation of performance data and measurements, to give useful information to decision makers. The aim of the research presented here is to apply some of the tools and techniques developed and standardised for the Semantic Web, to the areas of network management, network performance monitoring and measurement in new ways. Their utility, advantages and limitations when so applied are then assessed. This is about taking these established, standardised techniques for handling data and knowledge, and trying them out on known problems in this particular domain of interest. Embedding expertise, combining disparate data and mediating variability between sources have been identified as potentially beneficial functions of the languages developed for the Semantic Web. Realising these potential benefits requires the construction of some form of machine-readable knowledgebase, ontology or data-model, with the ability to automatically reason over the data and assertions contained within it. The ultimate aims of any application drawing on such a knowledgebase are to present useful information to the user and possibly to generate actions as a consequence of the results derived. The scope of this work includes the development of such constructs and testing them in real industrial networks. It also includes an investigation into the scaling issues, in terms of amount data and more particularly in reasoning power which can be applied.

The specific objectives of this research were therefore to conceive and then develop ontological solutions to four interpretation problems within the domain of network monitoring and measurement, using tools from the Semantic Web. These four problems can briefly be described as questions to be answered:

1. The Syntactical Problem; How useful are ontologies to overcome the problems caused by combining network datasets which contain the same type of data but are constructed with different syntactical arrangements and formats?

2. The Semantic Problem; How useful are ontologies in combining datasets containing data of completely different types which have a different meaning but whose combination yields useful higher-level information?

3. The Expertise Problem; How can human expertise be embedded into an ontology to automate and improve network monitoring, particularly in the identification and rectification of pathologies?

4. The Conflict Resolution Problem; How useful are ontologies in resolving apparent conflict between data sources which appear to give contradictory information?
The success criteria within this project are the extent to which comprehensive answers can be given to these questions when the techniques conceived and developed are applied to real-world case studies. Selection of suitable case studies which enabled these questions to be examined as fully as possible was therefore also part of the necessary work.

1.6 Research Contribution Summary

The contribution made by this research is discussed more fully in concluding Chapter 9 but in summary:

- A library of autonomous reference ontologies has been created, which provides a comprehensive, well documented and partially-populated framework for modeling the domain of computer networking. The *Loughborough Internetworking Ontology Library* (LIOL) has been rigorously tested for inferencing correctness and provides a view of this domain, solidly based on knowledge gained from the early reading and research carried out as part of the work. It has also been tested in case studies of real networks. LIOL is available in standard RDF serialisation form to other researchers and is a one of the major contributions of this work.

- The methodology developed and design work carried out has been applied to case studies on real networks. This has clearly demonstrated one approach which has been successfully used to store and interpret large quantities of disparate monitoring and measurement data utilising the tools and techniques developed for the Semantic Web. It has been proven that this approach can be used to provide high level information of value to network stakeholders, which is a major contribution.

- The generalised difficulty of network data interpretation has been broken down into four specific problems; *syntactical, semantic, expertise embedding* and *conflict resolution*, giving rise to the questions posed in Section 1.5. Some success has been achieved in tackling each of these problems and so it has been possible to give positive answers to each question, again proving the concept and making a further contribution.

- In the case of the conflict resolution problem, a solution has been demonstrated which uses basic set theory implemented using an ontology to give one solution method. This fresh approach has been applied with some success in resolving difference between separate methods of revealing the topology of the Internet, another novel contribution.
Chapter 2
Internet Monitoring Issues

2.1 Chapter Introduction

This chapter has a strong focus on the Internet. It gives some background on the important monitoring issues of interest and explains the motivation behind some of the research work in this area. It is an important foundation for the later chapters. The historical changing nature of Internet traffic, infrastructure and applications is explored in this context. Discussed then is a simplified taxonomy of measurement types. There is an emphasis on some of the problems which must be faced when attempting to monitor and measure the Internet; this is a thread which is later picked up in Chapter 3 in the context of interpretation issues. The later sections give a summary of some of the significant research projects in network monitoring, along with short descriptions of the institutions themselves who control those initiatives. The final section lists and briefly describes some of the available datasets and the tools used to produce them.

2.2 Motivation for Performing Internet Monitoring and Measurement

Interest in Internet measurements and particularly in their interpretation arises for several reasons. These reasons are categorised by Crovella and Krishnamurthy [33] as commercial, social and technical. The Internet is an important tool for global commerce. Information such as usage patterns, download speeds, efficiency of information transfer and congestion patterns can be valuable. Organisations may wish to have performance data on such things as proportions of dropped packets or available capacity when selecting an Internet Service Provider (ISP) for example. They may be interested in the range of capacities available to their target customer demographic before deciding on the design strategy for their web pages. Govern-
ments and researchers may be interested in Internet usage statistics as a guide to social trends or to answer essentially technical questions. Measurements of the volume and nature of traffic on the Internet are important in optimizing network topologies and in the design of network components such as routers and links. New applications can grow in popularity quickly and may change requirements, creating new stress points in the Internet. One example is the rise in popularity of social networks as described by Alan Mislove et al [39]. They report specifically on a study of YouTube, but also refer to e-mail groups and skype users, describing all these as overlay networks, which have the potential to be bandwidth intensive. These trends must be monitored and measured if the Internet is to continue to meet the changing demands placed on it. Development of new, more appropriate protocols, and avoidance of unacceptable levels of congestion and failure crucially depend on monitoring and measurement.

2.3 Introduction to Measurements of Interest

A convenient starting point in classifying Internet measurements is described by Crovella and Krishnamurthy [34]. A three layered approach is described, beginning with the physical and logical infrastructure, which carries the traffic, which in turn is generated by the applications. These three are interdependent, in that new applications generate more traffic which drives the growth of the infrastructure. More infrastructure capacity leads to the possibility of more bandwidth intensive applications and so on. This is illustrated in Figure 2.1.

**Infrastructure.** Infrastructure consists of the basic building blocks of the networks which together form the Internet. This includes the routers, switches and links, as well as the lower layer protocols which enable their operation. It also includes the topology or logical interconnection patterns of those building blocks at the levels of autonomous systems (ases), points of presence (POPs), routers and router interfaces. The Internet infrastructure is often represented by graphs.
at these four levels [34], with edges representing the links and vertices the network nodes. When considering this network topology, monitoring techniques are required to discover the links which form the edges of the various graphs, their interconnection points and their capacities. These links are often combined in complex ways and the parameter of interest may be the total throughput between two ASes for example. This discovery is particularly fraught with difficulties due to the sensitive commercial nature of much of this information. A significant amount of research has been carried out using various techniques to derive these graphs, particularly at the AS level [69]. Important measurements here include link propagation delays, capacities, packet losses, packet delays and jitter [35]. In addition, routers may be evaluated in terms of their forwarding delays, protocols supported and buffering capacities.

Traffic. The infrastructure described above carries the Internet traffic, which when considered at the network layer consists of IP packets traveling from source to destination via a series of forwarding nodes. The traffic can be analysed at various levels, for example single packets, packet trains or packet flows [34]. The statistical properties associated with the traffic are of considerable interest to researchers and can have a major impact on network performance analysis. To understand and characterise Internet traffic flows, it is necessary to collect sample flows, analyse them and then make some inference about the overall traffic from the necessarily small sample collected. Capturing full packet flows from high capacity networks requires a lot of storage, and so techniques are employed to gather samples of packet headers. This is one of the problems encountered and addressed in the masts project described in Section 2.6. Flows are normally considered to be a continuous exchange of packets between two points, normally distinguished by their source and destination IP address. The nature of TCP flows in the Internet is described by Kundu et al [10] as being mainly either mice or elephants. Mice are short lived flows carrying a small number of packets, whereas elephants are long lived large flows, which account for only 1-2% of the traffic volume.

Applications. The traffic on the Internet is generated primarily by the applications running on end systems, and it is to serve these user applications that the Internet exists. Introduction of new applications and their growth in usage are very important measurement issues, which can have a major effect on equipment design, network engineering and capacity planning. The Hobbes Internet Timeline [71] gives an insight into the growth of new “killer” applications on the Internet. For example in 1973, e-mail accounted for 75% of Internet traffic, having only been introduced just one year before. Likewise in 1996 the World Wide
Web overtook ftp-data to become the dominant application, with the number of websites exceeding 250,000, up from only 2,500 two years before. In a subsequent development, peer-to-peer file sharing applications such as Napster grew rapidly from nowhere in 1999 to account for 50% of Internet capacity by 2002 [21]. More recently, Video-on-Demand services have show rapid growth. Hossfeld et al [24] state that the number of YouTube videos viewed per day has increased from 200 Million in 2007 to more that 4 Billion in 2012. Monitoring activities are vital to provide an early warning of impending problems caused by such rapid change of use.

2.4 Monitoring Methods and Problems

In the early years of the ARPA net, extensive measurement and monitoring facilities were built-in, including the ability to measure the traffic flowing between any two nodes. As the Internet grew, and eventually became a commercially operated, decentralized set of interconnected autonomous systems (ASes), much of this measurement capability has been lost. The present day Internet presents some major challenges to effective measurement.

Layer Abstraction. The layered approach to networking, as described in the OSI model [31] means that components operating at each layer are substantially independent of the other layers. Routers operate at the network layer (layer 3 of the OSI model) using the Internet Protocol (IP) [54] and in general hold no state information about end-to-end connections. This simple approach reduces per-packet processing overhead and delay, but means that adding monitoring functionality would be expensive in both these factors. The ignorance at the network layer of the underlying link layer technologies on which it is built causes problems in measuring and understanding network performance, and a heavy reliance on choosing the most appropriate physical viewpoint at which measurements are made.

Administrative Issues Internet Service Providers (ISPs) have legitimate commercial reasons for keeping details of the performance and configuration of their networks private. For this reason, protocols such as ICMP [53] which are used in topology discovery may be partially blocked at the ISP’s gateway routers. Additionally, there may be private peering arrangements between ASes which are not widely advertised as routes using the Border Gateway Protocol (BGP) [58], the routing protocol of the Internet.

Experimental Measurement Error Internet measurements, like any other ex-
perimental results, are subject to errors, and it is important to have a good understanding of the precision and accuracy of any data gathered. A suitable approach to ensuring sound measurements is given by Vern Paxson in his 2004 paper [48].

2.5 Measurement Categories

Active Measurements Several networking protocols are supported which aim to derive network topology and performance information and to diagnose network pathologies. The traceroute utility makes an attempt to plot out the route taken through the Internet between points. It sends out a sequence of packets to the destination address with a deliberately low Time To Live setting in the IP datagram, actually increasing incrementally from “1”. This causes the receiving router at each hop to return a time exceeded error message back to the source, thus revealing it’s IP address. The resulting sequence of IP addresses represents a possible end to end route. This theoretical route however does not necessarily represent the true route taken by any individual packet, because forwarding decisions may be different at different times and peering arrangements may change [32]. Also, some routers are configured so as not to send the required error message, leaving an unknown IP address in the chain. Nonetheless traceroute is still a useful method for working-out routes taken in the Internet and it is used extensively in network research. One large-scale exercise which uses this technique is the archipelago project (Ark), managed by the Co-operative Association for Internet Data Analysis (CAIDA) [5]. Ark, like it’s predecessor Skitter, derives and collects topology data by sending out packets from a number of key nodes in the internet. Information of forward IP paths and round trip times (RTT) is obtained, and can be used to generate inter-as graphs.

Passive Measurements Network monitors capture copies of packets, principally to derive information about the nature and volume of traffic on a particular network. Packet analysers can be specific custom hardware, such as Endace DAG [8] devices, or general purpose computers configured using one of the standard libraries (such as libpcap) and with network interfaces configured in promiscuous mode to pass up all packets, not just those intended for the host itself. Other passive techniques involve using existing routing tables to derive graphs. For example, BGP [58] is the ubiquitous routing protocol between ASes. BGP tables can be used to derive AS graphs in particular. A major initiative which uses passive techniques is the CAIDA network telescope [62]. The network telescope monitors unsolicited packets received on a full class A IPv4 address range. Information can be derived on misconfigurations resulting in packets being forwarded to the wrong address.
Additionally, the scale of security threats such as worms and DoS attacks can be estimated. As the address range represents $1/256$ of the total IPv4 address space, some projections can be made about the entire Internet from these measurements.

### 2.6 Research Groups and Projects

**CAIDA**

The Cooperative Association for Internet Data Analysis (CAIDA) is a collaborative undertaking among organizations in the commercial, government, and research sectors aimed at promoting greater cooperation in the engineering and maintenance of a robust, scalable global Internet infrastructure. CAIDA provides a neutral framework to support cooperative technical endeavors [5] and is based at the University of California’s San Diego Supercomputer centre. CAIDA is a prolific organisation in terms of producing ground breaking papers in this area and has an extensive website with links to many useful datasets which are available to academic researchers worldwide. It is regarded as one of the leading organisation in the field of Internet measurement and monitoring. CAIDA have ongoing projects in Internet topology analysis, routing and security.

**Team Cymru**

Team Cymru Research NFP\(^2\) is a US-based non-profit organisation with team members in various countries around the world. They provide a range of services, free to the research community, including IP to AS mapping which draws its input from over 50 BGP peers and updates every 4 hours. The Team Cymru website is regularly updated with charts showing such information as trends in transport-layer protocols being carried by IP. There is also a newsroom page, with regular updates about significant Internet incidents, security issues and newly recognised trends. It is named after the family heritage of its founders rather than any direct connection with Wales.

**European Union Framework Programmes**

The Information Society Technologies (IST) programme was funded by FP5 and FP6 and provided an umbrella for a number of specific projects, including the MOnitoring and MEasurement project (MOME), which concluded in 2006. MOME offered a platform for knowledge and tool exchange, and for coordinating activities in the field of IP monitoring and measurement between IST projects and other European partners. The platform provided information on the interoperability of monitoring and measurement tools, as well as measurement data in a common format [41]. Under FP7, IST has been superceded by the ICT programme. There is a significant ICT element in

---

1[^1]: [http://www.caida.org](http://www.caida.org) Last viewed Sept 2013

2[^2]: [http://www.team-cymru.org](http://www.team-cymru.org) viewed September 2013
two of the four FP7 elements; Co-operation and Capabilities. A follow-up project to MOME is approaching completion under FP7, called MOnitoring and MEasurement in the Next generation Technologies (MOMENT). MOMENT aims to build on earlier EU initiatives to add value by integrating results and researching into semantic representation and retrieval. MOMENT is reviewed in more detail in Section 4.5. Although MOME has completed, there is an intention to maintain the database. Projects also associated with the EU framework programmes include GEANT³ and DIMES.⁴

**MASTS** ⁵ Measurement at All Scales in Time and Space (MASTS) was a UK project created to measure the JANet(UK) and JANet Lightpath networks. The aim was to collect packet header information to be anonymised and made available to the research community. The participating institutions were Loughborough University (CS and EE departments), University College London (EE department) and Cambridge University (Computer Lab). The ultimate aim was to provide a database consisting of data from packet header at layers 2, 3 and 4 of the well known Internet protocol stack [67], this from four monitoring points, three of which were carrying scientific/technical data on the JANet lightpath network and one on the main JANet network. The various challenges faced and features of the MASTS project are fully explained by Clegg et al [14].

**Planet Lab**⁶ PlanetLab is a research network administered by a consortium of major research establishments and Universities. The PlanetLab Consortium is managed by Princeton University, the University of California at Berkeley, and the University of Washington. The University of Cambridge in the UK is also represented.

**EMULAB**⁷ Emulab is a network testbed, originated as one of the PlanetLab projects, giving researchers a wide range of environments in which to develop, debug, and evaluate their systems. The name Emulab refers both to a facility and to a software system. The primary Emulab installation is run by the Flux Group, part of the School of Computing at the University of Utah. There are also installations of the Emulab software at more than two dozen sites around the world, ranging from testbeds with a handful of nodes up to testbeds with hundreds of nodes. Emulab is widely used by computer science researchers in the fields of

---

³http://www.geant.net Last viewed September 2013
⁴http://www.netdimes.org/new Last viewed September 2013
⁵http://www.mastsproject.org Last viewed Sept 2013
⁶http://www.planet-lab.org Last viewed September 2013
⁷http://www.emulab.net Last viewed September 2013
networking and distributed systems.

**PerfSONAR**

PerfSONAR is an infrastructure for network performance monitoring which aims to help with analysis of end-to-end performance data. The PerfSONAR software is a set of tools with a GUI, with an emphasis on visualization of end-to-end connections.

### 2.7 Data Repositories

**The Internet Measurement Data Catalog**

The CAIDA Internat Data Measurement Catalog (IMDC) is a public system for registering internet measurement datasets which are available. It is effectively meta-data only, and not in itself a source of data files. The catalog is administered by CAIDA, but is intended to reference both CAIDA and non-CAIDA generated data. Datasets in the form of pcap format files are available to download from sources which can be identified in this catalog [29]. The IMDC is by design a simple list of sources (meta-data only), and does not prescribe storage or access methods, or contain actual measurement data itself.

**IST MOME Database**

The MOME database contains data on packet traces, flows and routing obtained during the active phase of the MOME project. It differs from the CAIDA data catalog in aiming for a common, standard format throughout. The database is intended to support the former IST projects and future ventures in EU member countries with tools and data storage for IP data monitoring and measurement [17]. The MOME datasets are available to download, normally in pcap format and in addition, partial analysis has been performed on some to produce, for example a breakdown of traffic volume by application layer protocol. The MOME database has datasets up to 2005 only.

**Routeviews**

The Routeviews project at the University of Oregon established the routeviews repository, which is a collection of Border Gateway Protocol (BGP) views from many autonomous systems (ASes) [43]. This is a useful source of data for both AS level topology discovery and research into the operation and performance of BGP as the Internet exterior gateway protocol.

---

8http://www.perfsonar.net Last viewed September 2013
9http://imdc.datcat.org Last viewed September 2013
10http://www.routeviews.org Last viewed September 2013
RIPE Reseaux IP Europeans (RIPE)\textsuperscript{11} is one of five Regional Internet Registries (RIRs), operating under the auspices of ICANN to provide Internet resource allocation and registration services. RIPE covers Europe, the Middle East and parts of central Asia [59]. RIPE also plays a role in co-ordinating measurement activities within its region and has results available from several large scale projects, including the Routing Information Service (RIS), which passively collects BGP routing information. RIPE also operates one of the 13 root DNS servers, k.root-servers.net. Of the five registries, RIPE appears to be the favoured one for use by researchers, having the most comprehensive database of Internet measurements and packet traces.

Skitter and Archipelago Skitter (aka. the Macroscopic Topology Project) is a CAIDA programme which ran for over 10 years before being discontinued in February 2008 [7]. It used active probe packets launched from twenty-five beacons around the world to determine unidirectional forward IP paths and round-trip times. Traceroute probes were sent to over a million IPv4 addresses, aiming to get a response from each /24 segment. The successor to skitter is Archipelago (Ark) [6]. Ark is a more sophisticated measurement infrastructure, which emphasises co-ordination and the use of tuple spaces for easier data retrieval. Rather than using fixed addresses as in skitter, a random address in each /24 prefix is probed at least once in every 48 hours. The actual probing tool used is scamper [19], which has the added value of recording Round Trip Time (RTT) for each intermediate router, not just the end-to-end RTT as was the case with skitter.

Internet2/Abeline Observatory The former Abeline Network, now known as the Internet2 Network, links universities and research establishments in the USA [1]. As such it has similarities with the European Geant network, but has the fundamental difference that it does not peer with other ISPs and is therefore not strictly part of the Internet. The Abilene observatory, an activity within Internet2, has built up different databases with a variety of network related data. This data is available online through a variety of programmatic interfaces. The databases comprise a large correlated database for use by the research community, at international level, allowing retrieval of network related data on usage, netflow routing, latency and throughput.

\textsuperscript{11}http://www.ripe.net Last viewed September 2013
2.8 Chapter Summary

The Internet can be considered in terms of traffic, applications and infrastructure, three elements which are interrelated. Measurement of these elements is important for commercial, social and technical reasons. This is illustrated by the changing nature of the applications, such as file transfer, the WWW, peer-to-peer and video streaming which have dominated the traffic contribution historically at different times. There is a significant amount of research being carried out, with many institutions involved. One of the most prolific of these is CAIDA. The two principal categories of measurement are:

- **Passive Measurements**, which simply observe traffic without adding to it, such as capturing data packets of packet headers at viewpoints in the network.

- **Active Measurements**, using some kind of network probing technique such as ping or traceroute, which introduce some form of test data onto the network and record a response.

Datasets generated from measurements of both types are widely available from a number of publicly accessible repositories, and can be generated locally in most cases. The Team Cymru data service for example has been of particular value in this research.
Chapter 3

An Ontological Approach

3.1 Chapter Introduction

This chapter begins by focusing on the problems of interpretation of data to generate useful, higher level information. These problems are then categorised. The need for such information is highlighted following meetings with two potential industrial partner companies. Section 3.3 introduces the concept of Ontology from its philosophical roots to its more recent use in Computer Science. Section 3.4 gives the background to the Semantic Web; this is followed by a more detailed discussion of the Resource Description Framework (RDF) and the Semantic Web ontology languages in Sections 3.5 and 3.6 respectively. Section 3.6 then proceeds to explore ways in which these tools could be used to add value to data derived from the network monitoring domain.

3.2 The Interpretation Problem

Some of the difficulties faced when attempting to measure networks are described in Chapter 2, however there remains another set of issues which, although related, are not simply about obtaining these basic sets of monitoring data, but are more concerned with extracting meaningful, useful information from them. This higher-level information is ultimately what is needed by Internet stakeholders, to give more visibility and assist with decision making. This requirement has been highlighted in meetings with two potential industrial partner organisations, each of who’s customers are dependent on the Internet for the successful operation of their businesses. They are:

- Technical Resources (Midlands) Ltd (TRML), an IT support company, whose customers include many dealerships in the automotive industry, a chain of solicitors offices and of particular interest later, “Organisation-A” (real name...
withheld for security reasons), a service organisation operating in the UK and the Republic of Ireland.

- Thomson Reuters, a large multinational organisation which provides data networking facilities to the financial services industry and many other customers.

The specific problems highlighted in discussions with these two companies helped to more clearly define the more general problems which became the objects of this research. Notes from the information gathering meeting held with TRML representatives are included in Appendix A. Initially it is important to clearly identify those general problems which make it so difficult to extract and present useful, user-level information when so much raw data is available from so many ongoing monitoring and measurement initiatives and local sources. The issues for a system administrator requiring on-going broad-based information are of course different from those for a researcher conducting a one-off study in a narrow field of interest. In each case, however, some combination of the following problems will exist.

- **User knowledge and expertise.** In the meeting with TRML, minuted in Appendix A, the Managing Director (Mr Bright) said that there were times in the operation of their business when only junior staff were available to deal with complex networking issues which could arise when no senior staff were present. Although those staff members were capable of resolving basic desktop issues, some networking problems were beyond their level of expertise and training. It is the practical experience of the author of this report that this same problem does exist in other organisations. One potential solution is to embed such knowledge in a knowledgebase accessed by related applications.

- **Syntactical combinational problems.** Internet monitoring data and measurements are produced in many different formats. Examples include PCAP files, CSV, HTML, spreadsheets, databases, text based log files etc. Aggregating multiple data formats to a single study presents syntactical problems, which although not always conceptually difficult can be laborious to tackle. Automated tools are therefore desirable for key data formats. To have a single, well defined data storage and transfer may be difficult to achieve in practice, but RDF (Section 3.5) does offer one such solution.

- **Conflicting or inconsistent data.** It can be the case that different datasets which nominally represent the same situation may conflict to some degree.
Mediating between apparently conflicting data requires some domain-specific knowledge and context awareness, which again must to some extent be embedded. An example of this type of problem is highlighted by Mahadevan et al [69], in which three valid but different sources are used to obtain information about the AS level topology. CAIDA Skitter data, RIPE and ROUTEVIEW are all trusted sources, but three different results are obtained. In that example, the authors are able to offer an explanation for these differences but only by the application of significant domain-specific knowledge which would be outside the scope of most network administrators. By embedding some logic within the data, it may be possible to resolve the conflicts between these data sources, by favouring one over the others under certain circumstances, for example.

- **Semantic problems.** Understanding the exact meaning of the available data also requires domain-specific knowledge. Different sources may have different names for the same concept or conversely the same name for completely different concepts. The situation becomes more complex when the two elements are *almost* but *not quite* the same. Without machine-readable semantics associated with the data, a significant amount of human intervention is needed on a case by case basis. As an example, one data set may refer to the concept of an IP address as "IPADDR" and another may instead use the name "IPADDRESS". A person with domain knowledge, may be able reasonable to deduce that these are actually slightly different names for the same concept. This would be less easy for that person if a less obvious name for the concept had been chosen, such as “IA”, for example. The combination of these different hypothetical data sets would be much simplified and less prone to error if all these names could be understood to be equivalent at a machine level and combined accordingly. Taking this another step onward, the need for expert human intervention could possibly be reduced further if the application combining these data sets could also understand in some way that an IP address can be associated with a particular network node or host and when found in a certain packet can indicate that that packet was in-flight to that particular node, for example.

The field of network monitoring and measurement is not unique in encountering such problems. They are of a much more general nature and concerned with how to reason and represent knowledge in whichever domain is of interest. Nardi and Brachman [4] state that “Research in the field of knowledge representation is usually focussed on methods for providing high-level descriptions of the world that can be effectively used to build intelligent applications. In this context ‘intelligence’
Figure 3.1: Illustrating the need for Knowledge-Based Applications to interface between stakeholders and existing data and tools

refers to the ability of a system to find implicit consequences of its explicitly represented knowledge. Such systems are therefore categorized as knowledge-based systems.” The approach taken in this thesis therefore has been to bring to bear existing techniques which have emerged from research in the area of knowledge representation and reasoning to a specific situation in one domain of interest, the performance of the Internet and local networks as it relates to user perceived functionality. The experimental work reported on has been primarily concerned with building knowledge-based applications using semantic web technologies and to evaluate their potential usefulness and efficacy for network administrators. In other words, this is an attempt to bridge the gap between the type of available data and tools described in Chapter 2 and the higher-level information really needed by Internet stakeholders. This is illustrated in Figure 3.1.

3.3 Introduction to Ontology

An accepted standard dictionary definition of Ontology, such as given in the Oxford Complete Wordfinder [68] is “The branch of metaphysics dealing with the nature of being”. It deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences. One of the early ontologists was Greek philosopher Parmenides, who proposed a system of categorisation for entities which exist in nature. Palmer [46] describes the works of Parmenides and suggests that he greatly influenced Plato, who was in turn interested in differentiating that which exists in reality from that which is merely a conception of the mind. Plato modified the theory of Parmenides by introducing an intermediate
state of “becoming”, between that of “being” and “non-being”, which is analogous to the modern scientific idea of systems being in flux and entities such as particles being brought into and out of existence. There have been many other historical ontologists, including Saint Anselm of Canterbury, an 11th century English theologian who presented his ontological argument for the existence of God.

The above references are included to demonstrate that Ontology is therefore not a new concept. In recent years though, it has to some extent, been appropriated into the field of computer science. In that domain, rather than referencing ontology as a branch of philosophical science, it is more usual to refer to an ontology, meaning a model or knowledgebase which defines concepts within a domain of interest. John F Sowa [65] provides a description of how this evolution has taken place and then defines two sources of ontological categories, observation and reasoning. This is an important concept in the work presented here. The perceived value in the use of ontologies in network monitoring and measurement is not just about observing and storing data, but more about the inclusion of logic which enables useful information to be revealed using reasoning. Sowa also stresses the importance of selecting the correct ontological categories when building any knowledgebase, which proved to be valuable advice. The classic definition often cited within computer science works is that coined by Thomas Gruber, who says that an ontology is “a formal and explicit specification of a conceptualization” [20]. However a more complete and useful description as a starting point for the research presented here is given by Marko Grobelnik [40]. “An Ontology is a structure capturing semantic knowledge about a certain domain by describing relevant concepts and relations between them.” This encapsulates the aim of the experimental work carried out.

Each ontology must be based on either the closed-world assumption or the open-world assumption. In a closed world ontology, any statement that is not explicitly declared as true is automatically assumed to be false. In other words, everything that can be known about the domain of interest is known. In the open-world case, the assumption is that if a statement is not declared, then that statement cannot be known to be true or false [49]. It is effectively an admission that the information contained in the data model may be incomplete. In the case of the network monitoring domain, the open-world assumption has been chosen for this work for the following reasons:

- Computer networks are by their very nature interconnected and distributed.

The Internet is a complex network of networks and it is clear from the preliminary research reported in Chapter 2 that there is little chance of
knowing the whole topology at any given point in time.

- Networks are constantly evolving and changing. The infrastructure, traffic and applications are not static and any realistic modeling system must accommodate that fact.

- The network modeling undertaken was restricted to particular network segments of interest but the ontologies produced needed to be extensible if new questions were to be asked of them to produce useful high-level information.

- The Reference Ontology Library approach outlined in Section 5.6 allows for a “mix and match” approach to building larger ontologies from smaller interconnected parts. This approach needs to have an open-world assumption to give the flexibility required.

### 3.4 Semantic Web Background

The World Wide Web (WWW) has experienced rapid growth as a linked document repository, using the HyperText Markup Language (HTML) to specify the presentation and layout of pages when viewed using a browser. This concentration on presentation however, causes some limitations. Using the hyperlinks to move around the web relies entirely on human interpretation of the text and images presented, because there is no inherent meaning, or semantics there that can be understood by machines. Making the content available in a machine-understandable form introduces the prospect of applications to give much more powerful searching and linking of the vast number of documents available on line. In effect this means that the web connects together at the level of the data, which is tagged in some way to give meaning, rather than simply at the presentation level. Data are made available in machine readable form and can therefore be processed automatically to create a more consistent, useful and smart web of joined-up information. This is the motivation for the development and standardisation of tools and languages which can facilitate the evolution of the Semantic Web. These languages have their roots in Description Logic. An overview of the history and development of these languages is presented by Horrocks et al [70], who describe the evolution of Description Logic ideas via such languages as DAML+OIL to produce the Web Ontology Language (OWL). This paper is included in “the Description Logic Handbook” [47], and one striking aspect from the collection of papers there is that although there have been many other earlier initiatives, it is with OWL that finally a consensus has been reached and some major application-led standardisation achieved in the Description Logic arena. Specification of OWL and the
other languages utilised by the Semantic Web is under the control of the World Wide Web Consortium (W3C). In terms of the research presented in this thesis in particular, the standardisation of these languages by W3C is a very important factor, as it opens up the possibility of data transfer between previously unrelated projects, using standard well defined semantics and rules. This is the principal reason for selecting RDFS and OWL for this research. The W3C Semantic Web homepage\(^1\) states that the Semantic Web is about two things:

- Common formats for integration and combination of data drawn from diverse sources.
- Language for recording how data relates to real world objects.

The above are very general statements, but in fact the languages which have been standardised by W3C are finding practical uses in two diverse development directions:

- Enhancing the WWW with semantic capabilities, the original motivation, sometimes referred to as “Web3”.
- Data integration and knowledge management/representation in the more general sense, to create models, usually referred to as ontologies, of the concepts and individual objects in real-world domains of interest, not necessarily related to the WWW.

The difference between these two paths is described at length by Allemang and Hendler [22], who point out many possibilities and benefits which can potentially be derived by the application of these techniques to real-world domains of interest which may not be WWW related. The research presented here belongs to that latter branch and is concerned with combining measurement data from different viewpoints and understanding how those data relate to the real parameters of interest to Internet stakeholders, specifically network administrators. Thus the problems faced in this arena are similar to the general case described and there exists the prospect of bringing well established tools from the Semantic Web and applying them to good effect in the network monitoring domain. This has the potential to produce tools of great utility for real users of networks in general and the Internet in particular. Hebeler et al describe the Semantic Web Layer Cake [50], which illustrates clearly how the Semantic Web languages and concepts are built up. A simplified version of the one cited is shown in figure 3.2.

\(^1\)http://www.w3c.org/2001/sw/ Last viewed September 2013
3.5 The Resource Description Framework

RDF is a well known data modeling method, therefore only briefly summarised here. A full description of RDF is given by Powers [55]. The use of RDF offers the prospect of solving the syntactical problems described in Section 3.2, because there are precise standards agreed by the W3C 2, which means that any application which conforms to those standards produces output which can be understood by any other conforming application without translation or syntactical ambiguity.

All the data with which we are concerned in the network monitoring and measurement domain can be expressed in the form of RDF triple statements. RDF therefore offers a universal data exchange format, although this will not necessarily be the most efficient in terms of storage or network capacity utilisation for transfers. A triple consists of a binary relationship between a subject and object linked by a predicate, or property. All subjects and properties are considered to be resources. Objects can be either resources or data literals. All resources are identified by a Uniform Resource Identifier (URI). The Universal Resource Locator (URL), familiar in computer science and well known as a method of identifying pages on the WWW is one type of URI. However there can be others, for example the International Standard Book Numbers system (ISBN) used by publishers and libraries.

Each subject, predicate and non-literal object is globally identified in terms of referencing because of the URI. This can be quite cumbersome to use in practice, so a shorthand notation, called qnames is often used. Each resource is identified by a namespace and an identifier with a colon between. For example the namespace, http://nets.lboro.ac.uk/idn could have a shortened name declared of lboro, and the identifier within that declared namespace may be called hostA. The qname for this resource would then be lboro:hostA. This resource may or may not be the same as a resource with the same identifier in a different namespace,

2http://www.w3.org/RDF Last viewed September 2013
we are only declaring it within namespace lboro. Standard namespaces and abbreviations have also been declared for RDF (rdfs), RDFS (rdfs) and OWL (owl), amongst others. A very good illustration to understand the concept of RDF statements or triples is given by Hebeler et al [51], which equates every possible triple combination to a single atomic point in a three dimensional space, whose axes, in no particular order, represent:

- All possible subjects
- All possible objects
- All possible predicates or properties

This idea is illustrated in figure 3.3, which has an example of a triple with source http://nets.lboro.ac.uk#hosta, predicate http://nets.lboro.ac.uk#HasIp and object http://nets.lboro.ac.uk#192.168.10.251, the type of information we may wish to include in a network monitoring knowledgebase. Note that each of the elements of the triple are globally referenced, being prefixed with a unique URL, but cannot be said to be globally unique because more than one name can be used for the same resource. Also important is the fact that the object is a resource.
called “192.168.10.251”, not simply a value literal. This 3D concept shows clearly three very important aspects about RDF:

- **No Order.** There is no way to determine when points were added to the space (statements added to the model).
- **Easy Merging.** Two sets of points can be overlayed, creating a new, richer graph. Crucially, this property makes the merging or monitoring of other datasets easy once converted to RDF triple format.
- **No Duplicates.** If two statements have the same subject, object and predicate, they occupy the same point in space and therefore adding the second statement does not change the model. This can be very useful when automatically generating network models, as it is not always necessary to check whether a triple already exists within a given model before attempting to add it.

A semantic net can be built up when the object of one triple becomes the subject of the next and so on. These triples can be expressed, output from a model and stored in a number of standard formats, or serialisations. The serialisations commonly used are RDF/XML, N-TRIPLES and TURTLE. The RDF/XML mark-up format is primarily used here.

### 3.6 Ontology Languages

As described in Section 3.5, RDF offers a method for creating a data model and describing or specifying the relationship of individual data items to each other in a data network using a system of triples. It can therefore be used as a standardised method of data storage and exchange, thus reducing syntactical problems when attempting to relate or combine different data sets. What it does not do, however is to facilitate the machine encoding of semantics, or meaning into the model. This is the role of the ontology languages, the RESOURCE DESCRIPTION FRAMEWORK SCHEMA (RDFS) and the WEB ONTOLOGY LANGUAGE (OWL). Initially, RDF triples can appear to contain meaning, as shown in the simple example in figure 3.4. The left half of the figure shows an RDF triple describing a TCP flow, which has defined source and destination IP addresses. This appears to convey some significant meaning to a human observer with some domain knowledge, but this is only by human interpretation of the names used for the resources, which is subjective anyway and could actually be erroneous. The right half of the figure shows exactly the same relationship but with non-descript resource names. To a machine these two are equally meaningless without some context within an
ontology. RDFS and OWL are the languages specified by the W3C to give this context. These ontology languages can be considered to offer a range of reasoning, or inference capabilities, occupying a spectrum, with RDFS at the lower end, rising in terms of their possibilities for semantic richness through the different sub-languages (syntactic sub-sets) of OWL. This hierarchy is explained in several books on the subject, with a particularly good account given by Allemang and Hendler [23]. As well as describing RDFS and the recognised syntactic sub-sets available at the time of publication, they present RDFS-PLUS, a sub-set of their own choosing, which brings in some very useful OWL constructs. These are chosen for their utility and because they are relatively easy to implement in real-world applications, and so of particular interest here. However, OWL is a constantly evolving language, with OWL2 the current standard at the time of writing. Possibly the best source of information is the extensive library of specifications and recommendations made available on the W3C website. The documents referred to most during this project are the RDF Vocabulary Description Language recommendation [57] and the OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax [66], however the OWL 2 Web Ontology Language Document Overview [44], OWL 2 Web Ontology Language Profiles [45] were also important documents. The citations given can be referenced for a full taxonomy of the RDFS and OWL languages with details of all the constructs supported; the following description of RDFS and OWL is focussed on those aspects deemed of special interest to this research in the network monitoring and measurement area.

RDFS has a defined vocabulary which allows concepts within the domain of interest to be defined as classes. Those classes can then be related to each other in a super-class/sub-class hierarchy to provide a simple ontology, or taxonomy of concepts within the domain. Individual resources (instances) can then be declared as members of one or more of those classes. This simple process immediately gives meaning and context to those resources, both within the ontology and potentially, with appropriate combination methods, beyond it. RDFS allows
a similar approach to properties, allowing one property to be defined as a sub-property of another. Properties can also be defined to have a domain and/or a range, which specify the classes to which individuals must belong in order to be used as the subject and object respectively in a triple using the particular property. Domain and range are not used as restrictions, but rather for inferencing, making them particularly powerful. All these class and property relationships are expressed as RDF triples, in exactly the same way as any individual statement would be. By using an appropriate RDFS or OWL reasoning engine, the constructs in the ontology are used to create new inferred triples or entailments in addition to those originally asserted. An example of how this works, showing one such ontology with instance data from a computer networking view is illustrated in Figure 3.5. In the ontology, a namespace lboro has two classes defined within it, lboro:tcpFlow and lboro:ipAddr. Three properties are also defined, lboro:hasValue, lboro:hasDstIp and lboro:hasSrcIp. Sub-classes and range properties are also asserted. Then some instance data is gathered about a particular TCP flow, tcpFlow#1, (in this case only the source and destination IP addresses are shown), and set within the context of the ontology by declaring the TCP flow as a member of the class lboro:tcpFlow. Note that IP addresses when used in this way are resources, not simply numerical or text values. Their value needs to be asserted separately. Having seen what has been asserted, or
stated explicitly, what can now be inferred using a reasoning engine to enrich the information available?

- The resource `lboro:ipAddr#18` has no meaning in the instance data except for human interpretation from the name chosen. Because of the link with the ontology, we know that it is a member of the class `lboro:srcIp`. We know this because the range of the property of which it is the object (`lboro:hasSrcIp`) tells us that it must be.

- The resource `lboro:ipAddr#18` must also be a member of the class `lboro:ipAddr`, because of the sub-class which is declared.

- Similar information can also be inferred about resource `lboro:ipAddr#6`.

This simple case illustrates the potential utility of inference in our domain of interest. The inferences made in this example are a necessary, logical consequence of the assertions made in the ontology using `rdf:type`, `rdfs:range`, `rdfs:subClassOf`, and `owl:class` but many more constructs are available, particularly in the OWL namespace. The inferred information, or entailments, will also take the form of RDF triples, which can either be added to the resulting model as a permanent addition, or created at query time. The information could be further enriched by integrating the ontology with other instance data sources which can be then aligned with the existing dataset using other OWL constructs as appropriate. Even in this trivial example, constructs from the `rdf`, `rdfs`, and `owl` namespaces have all been mixed freely with each other and with user-defined classes and properties, illustrating the flexibility of these languages and the possibilities of this approach.

As discussed, reasoners, or inference engines are available to take the asserted statements in an ontology, reason over them and produce entailments which must be true as a necessary consequence of those assertions and the logical constructs available in RDFS or one of the specified OWL profiles. There are occasions however, when the RDFS and OWL reasoning methods do not adequately describe the local conditions. In these cases rules reasoners can be used as replacements or in addition to the pre-defined reasoners. The specific reasoners and rule engines used in this work are discussed further in Section 5.5.

### 3.7 Chapter Summary

Interpretation of network monitoring data presents a range of problems, which have been categorised as:

- User expertise and knowledge.
• Syntactical inconsistency when combining data from different sources.

• Semantic issues when combining disparate data into a single knowledge system.

• Resolving apparently (or actually) conflicting data.

The term Ontology originated in classical philosophy, where it is defined as “The branch of metaphysics dealing with the nature of being” [68]. In more recent times in Computer Science Grobelnik [40] defines an ontology as “a structure capturing semantic knowledge about a certain domain by describing relevant concepts and relations between them.” Certain languages and tools have been developed and standardised as part of the Semantic Web initiative and these can be used to develop data models and ontologies for any knowledge domain. Of particular interest here is their use in the field of network monitoring and measurement. The Resource Description Framework (RDF) is the data modeling language of the Semantic Web. Enhanced meaning, or semantics can be given to such a data model using the ontology languages of the Semantic Web, the Resource Description Framework Schema (RDFS) and the Web Ontology Language (OWL). To harness the potential of these languages, further tools in the form of reasoners (or inference engines) and query languages are needed to produce and find inferred knowledge.
Chapter 4

Related Work

4.1 Chapter Introduction

As described in Chapter 3, Ontology as a philosophical concept has been around for a long time. The more recent use of an ontology, to mean a conceptual or data model in computer science is also not new, so there are many examples of ontologies in research and in field use. Some examples of different types which have influenced and informed this project are discussed in this chapter. It begins with some well established, seminal ontologies and then develops to discuss some examples of more recent ontological work relating to networks in the general sense of any system having nodes and edges, culminating with some work of direct relevance to computer networks. In Sections 4.2 and 4.3 respectively, two well known ontologies, Cyc and foaf are briefly discussed. Section 4.4 goes on to discuss a proposal to use an ontological approach in a logistical parcel delivery network, analogous but not identical to a data network. In Section 4.5, other directly related research in the computer networking monitoring domain is discussed, in particular the MOMENT project.

4.2 CYC

An early seminal example is cyc (name derived from encyclopedia), primarily conceived as an Artificial Intelligence project in 1984 [42]. CYC was originally set-up to take sentences from an encyclopedia and uses a language called cyeL to represent them. The basic idea is to capture the concepts in the sentence which a human reader would already need an understanding of to make sense of the sentence before capturing the meaning in the sentence itself. cyc adopts a large-world approach to ontology design, in which wide domain boundaries are set and the ontology is populated with information about the world at large as a backdrop
from which smaller sub-domains can build. This hierarchical approach to building-up knowledge from a solid foundation is widely used as a method of increasing the usefulness of data, and would apply to the domain of computer networking. For example, it is necessary to have some understanding of the concept of a packet before making sense of the idea that a TCP Flow is made up of packets. Cyc has now been developed and is managed by a commercial company, Cycorp who list several success stories on their website, including modeling electrical parameters and test structures in a semiconductor yield management project [64], a similar area to network monitoring in terms of modeling quantities of technical data.

4.3 FOAF

An ontology which is sometimes used as the basis for worked examples in the semantic web texts [52], [55] is FOAF, (Friend-Of-A-Friend). FOAF [56] is a format for supporting descriptions of people and their relationships to each other. The relationships between people are semantically very similar to the relationships between organisations, or between computers on a network, both of which are relevant in networking. FOAF uses mainly RDFS inferencing rather than OWL, and therefore illustrates that reasoning at the simpler end of the semantic web language spectrum can be used successfully in relationship modeling. The idea of achieving a lot in terms of utility with nominally less powerful inferencing proved to be an important feature of the design work carried out in this project and discussed in detail later in concluding Chapter 9.

4.4 Multi-perspective Ontologies for Logistics

In a 2011 paper by Xu et al [13], the authors describe the idea of multi-perspective ontologies to provide different views of the same basic system. The motivation is the need to continually monitor the performance of a logistics operation, which is approached by seeking to identify exceptions to the normal, successful conditions and generate actions accordingly. This was interesting and of relevance because it is analogous to the idea of monitoring a computer network to rapidly identify, diagnose and repair any network pathologies, or exceptions. The ontologies proposed are:

- A simple taxonomy of possible exceptions and causes thereof.
- A social dependency ontology, with the departments or entities as nodes and the actions they must perform for each other as the edges.
• A dynamic ontology, which models the system state at any particular point in time.

As well as the idea of looking for exceptions, this approach was also interesting and potentially useful in the computer networking domain, where relatively static infrastructure serves applications by passing data traffic. Some of these theoretical ideas had some influence on the design work described in Chapter 5. The paper describes two theoretical case studies, one in which the delivery address was wrong and another in which planning had gone wrong and no driver had been allocated to the job. The approach described appears sound, although no practical working system is actually reported or evaluated.

4.5 Computer Network Management Studies

In 2009, The Journal of Network and Systems Management published a collection of papers called *A Snapshot of Ontological Approaches for Network and Service Management*. One of these papers [11] gives a brief summary of a number of case studies and lists some lessons learned from the use of OWL, SPARQL and the Semantic Web Rules Language (SWRL), which gave some useful insights into the practicalities of these languages. In total three case studies are presented, two being of particular interest here:

• An extensive study by Lopez de Vergara et al [37], in which the authors describe how OWL was used to represent relationships between services, devices, configurations and user profiles in a home gateway system in which the operating telecommunications company could deploy new services to existing customers. The information of interest had to be inferred from data which were merged from different sources, just as would be the case in a network monitoring model. A system is described in which SPARQL queries are addressed to a high-level ontology, which is linked to the instance data using a custom mapping ontology. This is analogous to the reference and linking ontologies created in this project as introduced later in Section 5.6 and described more fully in Chapter 6, which combine to link to the monitoring instance data.

• As mentioned in Chapter 2, the European Union have funded projects as part of their FP5, FP6 and FP7 framework programmes which have relevance to network measurement in general and monitoring of the Internet in particular. MOnitoring and MEasurement in the Next generation Technologies (MOMENT) [27], is a major work package within the European Union FP7
framework. The overall objectives of MOMENT are the closest found in initial research to the aims of this project, in that semantic web techniques have been used to bridge across a wide range of completed and ongoing EU projects, their tools and their resulting data. A paper was presented at the 1st European Workshop on Mechanisms for Mastering the Future Internet [12] which gives a short insight into the thinking behind MOMENT, but had no real examples or useable methodology at that point. A much more comprehensive internal report [16] has a large list of measurement and monitoring tools and some interesting class diagrams. Several papers of interest to this work have been produced under the MOMENT umbrella, in particular “A Semantically Distributed Approach to Map IP Traffic Measurements to a Standardized Ontology” [61] which helped with the development of the case study methodologies used in this work. The paper describes the MOMENT ontologies and how a layered approach to using them could be adopted for a measurement exercise. The separate ontologies which can be combined are for data, metadata, anonymisation and an upper ontology for general network measurement concepts and measurement units. These ontologies address the kind of large-scale data repositories and measurement projects which fall under the EU umbrella as described in Chapter 2, but some of the design ideas were of direct relevance to the smaller-scale work carried out, although a different approach to ontology hierarchies was eventually chosen here.

Interestingly the authors of the paper [11] felt that OWL lacked the necessary expressiveness for some of their purposes. Their solution was to compensate using semantic web rules, in particular the Semantic Web Rules Language (SWRL). Rules are also used in one of the case studies carried out in this project, for the purpose of entering specific if-then constructs to add expertise to the knowledgebase.

4.6 Chapter Summary

A number of very different ontologies at various levels of development have been reviewed. Some very useful pointers and ideas for the later design work were obtained from this related research, in particular:

- Fundamental, foundation concepts must have meaning in any ontology before more detail can be built upon them.

- A number of smaller ontologies can be combined, with care, to form a hierarchy. This led to the concept of a library of ontologies within the network monitoring domain, expanded upon in Chapter 6.
• Ontologies giving different views of the same system are sometimes needed to accurately model a real situation.

• Computer networks have much in common with other types of networks in terms on modeling with ontologies.

• The well-known Semantic Web tools (RDF, RDFS, OWL, SPARQL and SWRL) have been shown to be theoretical solutions in some circumstances for modeling networks.

• The OWL constructs can be enhanced by adding semantic rules to customise inference when needed.
Chapter 5

Design and Methodology

5.1 Chapter Introduction

This key chapter explains the overall approach taken to the practical work undertaken. In Sections 5.3 and 5.4, the tool options and development environments which were available are explained in some detail and the selections made are justified in the context of the network monitoring and measurement domain. Section 5.5 has a detailed description of Jena, the framework chosen to implement the required semantic modeling and therefore at the heart of the programming work. Relevant Jena-related Java code snippets are included. The concept of using a library of reference ontologies is presented in Section 5.6, with specific reference to the Loughborough Internetworking Ontology Library (LIOL); a major part of the research contribution of this work. This is expanded upon in Chapter 6. A generalised methodology for ontology design and build is explained in Section 5.7. The two practical case studies which were carried out, The VPN Study and The AS Topology Study are also introduced in Section 5.8, to be expanded upon in Chapters 7 and 8 respectively.

5.2 Approach Summary

The fundamental aim of the practical work reported here was to build and evaluate software models, which use available basic data to give improved understanding of the operation of networks. Identification of any pathologies present and their possible causes was an important aspect of the work. The techniques (and specifically the languages) of the semantic web described in Chapter 3 were applied to some of the network monitoring and measurement issues described in Chapter 2 in ways not known to have been attempted before. The approach taken to the ontology design has been pragmatic, in that the design evolved throughout the
process. It is important to state that there is not only one possible correct model but rather that the concepts and relationships within the network monitoring and measurement domain could be modeled in different ways, or more descriptively from different viewpoints. The ontologies developed and used in this research do present one correct representation but it must be acknowledged that there could be other equally valid viewpoints which would look very different. The ontologies which form part of the library developed and used in this research have evolved in an iterative process to be of practical use in answering real questions about networks. Although some of the work was directed towards the Internet in particular, many of the issues tackled are common to computer networking generally.

The data used were gathered wherever possible from working, live networks and well-known simple monitoring tools were used for data gathering. Two distinct case studies were undertaken, one with the co-operation of an industrial partner, using data captured from one of their customer networks and one within Loughborough University, using data generally available to researchers within the Internet monitoring and measurement community. Using these two, with some overlapping and combination in the final analysis, techniques are demonstrated which:

- embed machine-understandable, domain-specific knowledge within a knowledge-base, to automate some fault diagnosis and reduce repetitive manual testing.

- combine locally-gathered data with publicly available datasets to provide new, high-level, useful information.

- resolve conflict between information derived from different monitoring approaches.

The overall, step-by-step approach taken to the practical work can be very briefly summarised as follows:

1. Construct a library of general purpose, top-level reference ontologies for the networking domain, defining the major concepts and their relationships to each other.

2. Construct lower level, local ontologies for each of the specific case studies, with certain predefined issues in mind in each case.

3. In each case study where it is needed, capture appropriate local data, and where required also gather general Internet data.

4. Convert those data into standard form RDF triples, creating further triples to associate the data with concepts in the local ontology thus giving them meaning.
5. Merge the local and top-level ontologies, where necessary and appropriate.

6. Apply semantic reasoning, custom rules and queries within applications, in an attempt to achieve the wider objective of using the standard semantic web tools to improve understanding by stakeholders within the networking domain.

7. Analyse results and perform further iterations to test the utility and scalability of the applications.

In some situations, particularly when the problem is about embedding expertise, only the ontology and some rules may be required, with no need for instance data as such. This is the expert system approach. The idea here is that expert knowledge is embedded in the ontology by the resources and properties defined and by the rules asserted. Less experienced operatives can enter the observed symptoms of a particular network problem or pathology and the application can then use the asserted and inferred knowledge in the system to suggest possible causes, for example. In other situations, a significant amount of instance triples will be created, derived from real data about the network. The approach outlined here aims to cater for both of these situations.

### 5.3 Building Blocks and Frameworks

The languages of the semantic web are briefly described in Sections 3.5 and 3.6. They are essentially languages for building a data structure and making general ontological statements. In practice they need further utilities and application software to create real knowledgebases and to realise the potential benefits they offer. Figure 3.2 shows a simplified version of the semantic web layer cake, illustrating the building blocks of the semantic web concept. Having created an ontology of classes, specified the properties (predicates) which relate them to each other and linked this to instance data, it is then necessary to apply semantic reasoning to produce inferred knowledge in the form of additional RDF triples or entailments. All of this must be retained in some way, usually in a persistent triple storage method. To use the resulting information, applications must of course have some method of querying the knowledgebase created. Therefore in order to make practical use of an ontological approach in the networking (or any other) domain, it is necessary to have some sort of programming framework which has facilities to generate these various slices of the cake. Using such a framework, the programmer must be able to:
• build RDF models of actual instance data, such as packet traces, flow data, routing and addressing information, packet loss data and time delays. Although operations on models will take place in memory, of course persistent storage of the resulting RDF triples is also required and must take some form of disk-backed or other non-volatile system.

• create ontologies, where concepts within the network of interest are specified, such as server functions, relationships between nodes in the network and at a higher level the definition and relationship between fundamental concepts, such as flows, addresses and protocols. Within the ontology slice, there must be a reasoning engine which can be applied to the asserted model to implement the logical inferences which the chosen ontology languages RDFS and/or OWL specify. This inferred information in the form of new triples, or entailments is where the value of this approach lies.

• apply network-specific rules to the ontology as a method of extending, or tuning the logical constructs provided in RDFS and OWL. In addition to the knowledge embedded by asserting resources and properties in the ontology and then applying RDFS and/or OWL reasoning, rules offer another method of adding what may be called domain-specific logic. In this way expertise on the likely consequences of particular situations can be included and used as another way to add or remove triples from the knowledgebase. For example, a rule may be used to state that if server-x ping test from node-y fails then application-z will fail. In this situation a new triple would be created, which some user-focussed application could detect and then generate an action or warning.

• query the knowledgebase to extract the data needed by user-facing applications to deliver useful information to stakeholders.

A number of packaged tool sets, or frameworks, both commercial and open-source are available to download for this purpose, based on various programming languages. They all provide some level of facilities for building and linking ontologies, converting data to a standard RDF, storing RDF triples, querying and in most cases the (all important) reasoning and inference tasks. In order to choose the most appropriate tools for the planned research, an initial web-based search was done and four possible frameworks using different programming languages and different approaches were identified as potentially useful; Redland\(^1\), RDFLib\(^2\), Sesame\(^3\) and

\(^1\)http://librdf.org, last viewed 1st August 2013

\(^2\)http://github.com/rdflib, last viewed 1st August 2013

\(^3\)http://openrdf.org, last viewed 1st August 2013
Jena\(^4\). These were assessed against the following criteria:

- Triple store availability.
- Reasoning (Inference) functionality.
- Query and Rules functionality.
- Usability (ease of coding, availability of documentation and tutorials etc.).
- Available user community support.

Fuller explanation of the criteria used and the results of the assessments made are available in Appendix B; a very brief summary of the findings is shown in table 5.1. On the basis of these results, Jena was found to be the most comprehensive pro-

<table>
<thead>
<tr>
<th>Criterium</th>
<th>Redland</th>
<th>RDFLib</th>
<th>Sesame</th>
<th>Jena</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>Native C, with bindings to Python, Perl, PHP, Ruby)</td>
<td>Python</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td><strong>Triple Store</strong></td>
<td>Many options, including MySql</td>
<td>Available using Berkeley db</td>
<td>Many, flexible options, including OWLIM</td>
<td>Many options, Native TDB very attractive for ease of use.</td>
</tr>
<tr>
<td><strong>Reasoning</strong></td>
<td>RDF only</td>
<td>RDFS/OWL by using FUXI</td>
<td>RDFS, with possibility of OWL if using OWLIM storage</td>
<td>RDFS/OWL support integral</td>
</tr>
<tr>
<td><strong>Query and Rules</strong></td>
<td>SPARQL support using RASQAL lib</td>
<td>SPARQL</td>
<td>SPARQL</td>
<td>SPARQL</td>
</tr>
<tr>
<td><strong>Useability</strong></td>
<td>Very limited descriptive documents, no tutorial found. Difficult start-up</td>
<td>Excellent tutorials. Easy start-up</td>
<td>Excellent tutorials. Easy start-up</td>
<td>Excellent tutorials. Very Easy start-up</td>
</tr>
<tr>
<td><strong>User Community</strong></td>
<td>Some mail lists and fora</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Excellent. Many on-line postings, tips and examples</td>
</tr>
</tbody>
</table>

\(^4\)http://jena.apache.org, last viewed 1st August 2013
gramming framework, meeting all the stated criteria and was therefore chosen as the principle semantic web tool-set to use for this research. It should be noted that this initial assessment suggested that both Sesame and RDFLib could also have met the requirements for the project, but there was more uncertainty about the ability to integrate separate modules to form a cohesive framework. Redland, being implemented in C, occupies a more specialist niche and may be the best option in applications other than this, when flexibility is needed and robustness and speed of development are less of an issue. In addition to the frameworks already described, the well known open source ontology editor Protégé\(^5\) offers an alternative approach. Protégé is based on Java but unlike Jena and the other frameworks considered here it has a Graphical User Interface which provides a more “What-You-See-Is-What-You-Get” method for easy creation of ontologies. It has plug-ins which are compatible with OWL and RDF\(\text{s}\) [30]. Protégé version 4.0 was initially downloaded and found to be very useful as an aid to learning about ontologies and the principles involved. It was decided however that the programming API approach, as used by Jena, would ultimately offer a more flexible and comprehensive methodology for this work. Protégé version 4.3 has proved extremely useful though and has been used fairly extensively in the later stages as a means of checking the validity of the ontologies produced using Jena. The RDF/XML serialisation used by Jena to store RDF triples is completely compatible with Protégé, so it was possible to import them and view them in the Protégé GUI window. The class hierarchies, property characteristics and individuals could be displayed in an easily readable form there thus exposing any errors or unintended consequences. The Protégé OwlViz tab was also very useful to view class hierarchies as diagrams.

5.4 Development Environment

The Java programming language has been used throughout the practical work reported on here. This was a natural consequence of the choice of Jena as the principal semantic web programming framework. The initial development work was done on a Macbook Pro, running MacOS X vers 10.5.8, with 2 GHz Intel Core Duo CPU and 2GByte memory. Java version 1.5.0.30 was used, and coding was carried out using the Netbeans Integrated development Environment (IDE) version 6.8. During the later stages, development was moved onto a Dell Inspiron 530 desktop machine, running Ubuntu version 12.0.4, with 2.6 GHz Intel Core 2 Duo CPU and 4GByte memory. On this machine Java version 1.6.0.27 (openJDK)

\(^5\)protege.stanford.edu Last viewed 1st sept. 2013
was used, and a change was also made to use the Eclipse IDE version 3.7.2. The change of hardware platform was for practical reasons as the original machine neared the end of its useful life. The original choice between Netbeans and Eclipse as development environments on the macbook was arbitrary; either would have been perfectly acceptable. However, when moving to the Ubuntu operating system there were repeated compatibility issues with Netbeans causing screen freezes, so work was moved across to Eclipse as a pragmatic solution.

5.5 Jena

The Jena framework is accessed by downloading a number of Java archives (JARs) and adding them to the Java classpath. The many Jena packages are then available to import as needed. The term model is the terminology used widely in Jena to represent entities which could otherwise be referred to as knowledgebases, ontologies, triple stores or datasets depending on the context. Jena is a comprehensive framework with many functions and often several different ways of achieving the same objective as a programmer. Some of the most important packages, classes and methods used in this project are outlined here to give some background to the implementations discussed in subsequent chapters.

**The ModelFactory Class.** This is part of the com.hp.hpl.jena.rdf.model package. The methods from this class are used extensively to create and access the different types of model which are needed. Of particular interest here are the createOntologyModel and createInfModel methods, which as the name suggests are used to create (and access if already in existence) ontology models and Inference models respectively. In the Jena sense, ontology models can contain not only simple RDF statements, but resources such as owl classes and properties. Inference models are usually bound to both an ontology model and a reasoner. They contain the original ontological and instance assertions from the ontology model and also the entailments which are created by applying that particular reasoner to it. Example code snippet:

```java
OntModel ontModel = ModelFactory.createOntologyModel();
```

**The OntModel Class.** This is part of the com.hp.hpl.jena.ontology package. Methods from the OntModel class are used to build up a particular model by creating, accessing and configuring entities within such as classes, properties and individuals. At this stage it is possible to create many different types of properties, using methods such as createTransitiveProperty and
createSymmetricProperty, whose names clearly indicate their function in the model. Example code snippet:

```java
OntClass Port = traOnt.createClass(defaultNameSpace + "Port");
```

The ReasonerRegistry Class. This is part of the com.hp.hpl.jena.reasoner package. Methods from this class are used to get instances of the various built in reasoners, for example the getOwlReasoner and getRDFSReasoner. There are many built-in reasoners with different levels of reasoning capability, aligned with OWL and RDFS. Example code snippet:

```java
Reasoner reasoner = ReasonerRegistry.getOWLReasoner();
reasoner = reasoner.bindSchema(ontModel);
```

The GenericRuleReasonerFactory Class. Rule reasoners fit into the Jena system in the same way as any other reasoner in that they are bound to an ontology model and produce entailments in an inference model. The key difference is that the entailments are created not as a consequence of standard OWL or RDFS logical constructs, but rather as a consequence of custom Jena rules. These rules can be used to create and/or remove triples as a consequence of the prevailing conditions in the ontology model whenever it is touched. Rules are written either in the body of the code or more normally in a text file using a particular syntax and then parsed into a suitable format to be passed to the GenericRuleReasoner constructor when creating the reasoner instance. Example code snippet:

```java
List rule = Rule.rulesFromURL("file:/home/ian/rule2.rul");
Reasoner reasoner = new GenericRuleReasoner(rule);
reasoner = reasoner.bindSchema(ontModel);
```

The QueryFactory Class. This is part of the com.hp.hpl.jena.query package. In Jena a query is first created by processing a query string, which is created by the user in the standard SPARQL syntax. SPARQL (SPARQL Protocol and RDF Query Language) is the standard query language for the semantic web [15]. The string is processed to form a query object which is then used to execute a query against the model. The queryExecutionFactory class is brought to bear by calling the create method to do this. Example code snippet:

```java
StringBuffer queryStr = new StringBuffer();
queryStr.append("PREFIX lboro: <http://nets.lboro.ac.uk/liol#>");
queryStr.append("PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>");
queryStr.append("PREFIX owl: <http://www.w3.org/2002/07/owl#>");
queryStr.append("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>");
queryStr.append("select ?a where{?a rdf:type lboro:tool} ");
Query query = QueryFactory.create(queryStr.toString());
// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query, infmodel);
ResultSet results = qe.execSelect();
//Output query results
ResultSetFormatter.out(System.out, results);
qe.close();
```
The TDBFactory Class. Part of the com.hp.hpl.tdb package. The TDBFactory.createDataset("/home/mychosenlocation") method is used to create a disc backed dataset and return a dataset object. TDB then creates a directory (in this case /home/mychosenlocation), with a number of files. Further methods are then used to create a model associated with the dataset, which is in turn used to create a model of the required type, for example an ontology model. Example code snippet:

```java
Dataset ds = TDBFactory.createDataset("/home/ian/tdb51");
Model model = ds.getDefaultModel();
OntModel ontModel = ModelFactory.createOntologyModel(null, model);
```

tdb is a database system which generates a number of tables within the directory for storing information about the various RDF nodes [28]. It can be implemented on both 32-bit and 64-bit machine architectures, and has built-in caching of the results of queries executed within a Java Virtual Machine (JVM). A number of systems have been developed for benchmarking the performance of triplestores, including the Berlin Sparql Benchmarking (BSBM). The BSBM is introduced by Bizer and Schultz and used to compare the performance of several RDF triplestores and traditional relational databases [3]. In terms of query response speed the fastest relational databases were shown to be up to 8.5 times faster than the fastest RDF based store. Another study [38] uses a different benchmarking system to compare Virtuoso-open source, Jena sdb, Swift OWLIM, 4Store and Jena TDB. The results between the triplestores are not conclusive overall in either paper and vary greatly with query type, however TDB was normally neither the best nor the worst in the tests reported, but its performance did tend to drop off more than the best performers with very large datasets of 100 million triples.

5.6 The Reference Ontologies

The basic requirement for a high-level reference ontology is to provide wider meaning and context to the concepts and instance data in the detailed ontologies and data models further down in the hierarchy, by linking appropriately to them. This has value in the local sense, giving semantics to the concepts defined and instances generated internally, but it also facilitates meaningful linking to the outside world.

The vision within this work was to create a library of relatively small, high-level ontologies which can be selected, linked together and loaded in various combinations into models depending on the case in hand, before linking to the case-specific
data. This offers an agile, flexible methodology which aims to avoid the need to use unnecessarily complicated ontologies, without sacrificing the required semantic richness in any particular situation. It is essentially a “horses for courses” approach.

The library which has been created, the *Loughborough Internetworking Ontology Library* (LIOL) consists of a number of autonomous ontologies (AOs), each addressing different aspects, or viewpoints of networked computer systems. Each AO contains various owl:Class definitions pertaining to that particular aspect, which are formed into taxonomies using rdfs:subClassOf. Individual resources are also specified within the AO when they are considered to be generic across the networking domain. For example, the concept of the Internet Protocol (IP) will be used in individual case studies, but because it is a commonly understood idea across the whole domain, it is appropriate to include it at the level of the reference ontologies rather than at a lower level. Where necessary properties, with associated inference capabilities, are also asserted. Initially six Autonomous Ontologies have been created, loosely based on Crovella and Krishnamurthy’s three Internet elements, described in Section 2.3 and illustrated in figure 2.1. Their traffic and applications elements are retained as single AOs, however their infrastructure element has been expanded into three distinct AOs: organisations, infrastructure and protocols, also a new element, monitoring has been added. After identifying the logical structure of classes which would be needed to describe the network monitoring domain, this six-category split was felt to be a more logical approach for ontology modelling, and a better representation of the real-world concepts of interest here. The LIOL ontologies form an important cornerstone of the implementation work carried out. The Library is also intended to have the potential for wider use, and so is in itself a part of the contribution made by this project. The detailed implementation of the LIOL ontologies, along with test results is described in Chapter 6.

The content, design and final structure of each AO was arrived at by an iterative approach. There was a significant amount of trial and error, testing at each stage and making careful checks that the concepts embodied in the classes and properties were indeed correctly aligned with the real concepts in computer networking in general and the Internet in particular. In this way, each of the AOs evolved throughout the project and was gradually refined to the form presented in Chapter 6. In particular, the inferences inherent in the structures were tested at each stage and sometimes unexpected results were obtained. On some occasions the properties used needed to be changed or classes redefined. This iterative, agile method has meant that the AOs in the form presented are actually quite
mature and robust. A similar approach was adopted to the design of the more localised ontologies used in conjunction with the AOs in the case studies reported in Chapters 7 and 8.

5.7 General Methodology Description

In Chapter 3 some of the problems in interpreting data and test results to understand the behaviour of computer networks are summarised. Some ideas for an ontological approach to these problems using the semantic web languages and tools are then presented, along with an introduction to those techniques in general terms with some theoretical and historical context. In Chapter 4, some related work in this and other domains of interest is discussed in this context. In this chapter, an overall methodology is developed, including a step-by-step process. Framework and tool selection is justified and those tools considered most appropriate for the job in hand are brought into focus and explained in more detail. A number of the Jena framework techniques which are of particular importance are illustrated with code snippets. In addition, the Loughborough Internetworking Ontology Library (LIOLO) is introduced as an important cornerstone of the methodology. All these threads are brought together in the proposed General Methodology, which is can be represented as a process diagram. A key to this diagram and all other methodology process diagrams used in future chapters is shown in Appendix F. The General Methodology is illustrated in figure 5.1, which is the synthesis of the discussion so far. The General Methodology is applied to the case studies introduced in Section 5.8 and presented in detail in subsequent chapters, but it is also offered, as the name suggests, as an approach which could be used for other cases in the networking domain and beyond. The General Methodology can be briefly explained with reference to figure 5.1:

- The necessary data in various syntactical formats are gathered from both local and public/general sources. Data converters, specific to the role and implemented as Java classes, are used to convert the data into RDF triples in standard format. Appropriate Jena classes are used in the Converter code as needed, as are other imported Java classes. Note that even at this early stage some form of “hook” must be coded to the instance data so that they can be associated with the ontology model later.

- The resulting RDF instance data can be either stored as a Jena TDB dataset or brought straight into the Jena Ontology Model in memory, as appropriate.

- The Local Ontology and Linking Ontology are specific to the case, but of
course refer closely to the LIOL ontologies chosen. They are created in Java using the necessary Jena classes, and stored as RDF/XML files.

- The Jena Ontology Model is the place where the chosen general LIOL ontologies, the local ontological statements about this specific case and any linking ontological statements (read in from RDF/XML files) are all brought together. This is made very simple because of course they are all serialised as RDF triples in standard form, and need not be in any particular order. The instance data triples are also imported in the same way.

- The Jena Ontology Model can be queried directly without the use of a specific reasoner using the built-in option OntModelSpec when creating the model to specify the inference level. If this option is chosen, the entailments will be included and returned in the query result set, but not persisted in the model.

- If either a persistent copy of the inferred statements is required or if rules are to be implemented, a Jena Inference model must be created. The appropriate OWL, RDFS, and/or RULE reasoners are loaded and bound to the two models to facilitate the persistent inferencing. Queries can then be applied to the Inference Model.
5.8 The Case Studies

A significant focus of this research has been on developing new ideas into actual ontologies and models which have the potential to be of practical use in real networks. It is natural therefore that the environments in which these concepts have been tested are, wherever possible working commercial networks. In Section 3.2, it is stated that discussions which were held with two commercial organisations helped to identify the problems which this work would address and to specifically define the questions which it would attempt to answer. It was necessary to select suitable case studies which together could provide a vehicle to test the concepts and techniques developed. Both Thompson Reuters and TRM Ltd were very helpful and willing to assist in the research as far as they were able and both had networks under their control which could potentially have been used as case studies. In the Thompson Reuters case however, the nature of their work was such that only very limited access could be granted to the researcher for security reasons. All IP addresses needed to be obfuscated, meaning that relating internally derived data to public repositories would not have been possible. Active measurements such as sending various probes into the network was also not allowed, which would have been very restrictive for this project. TRM Ltd on the other hand were prepared to allow use of real IP addresses and also to allow a test server to be installed at the heart of one of their customer’s networks, providing an ideal solution for performing both active and passive tests. For these reasons TRM’s “Organisation-A” network was chosen as the basis for one of the cases, the VPN study. In addition to this, the work of Mahadevan et al [69] cited in Section 3.2 had already inspired the idea of using ontologies specifically to resolve perceived conflict between the various methods of deriving the AS level topology of the Internet. Suitable data were available for this purpose from work carried out at Loughborough University. This was chosen to form the basis for a second case, the AS Topology Study. The case studies are described at length in their respective chapters, however it is important to understand how they collectively address the original interpretation problems identified in Section 3.2, which are themselves closely aligned to the possible advantages of taking an ontological approach identified in Section 1.4.

The end user organisation who own the network used for the industry-based case study is referred to as Organisation-A for security reasons because at least some parts of their network topology are exposed. Organisation-A is supported by Technical Resources (Midlands) Ltd (TRML), who were the main liaison point for that particular study. The names given to the case studies give a general indication of the network environments in which they were conducted. They were:

1. The VPN Study; with organisation A, a service organisation with branches
around the UK and Ireland.

2. *The AS Topology Study*; internal to Loughborough University.

The four problems addressed can be summarised as:


2. *The Syntactical Problem*; overcoming syntactical difference problems in disparate data.


4. *The Conflict Resolution Problem*; resolving conflicts between data sources.

The VPN study was the major part of the work, addressing the first three problems. The Internet topology study was targeted particularly at the conflict resolution problem and concerned modeling two different techniques for discovering AS level links in the Internet.

The relationship between the case studies and the problems addressed within them is illustrated in Figure 5.2

![Figure 5.2: Case Studies Relevance to Research Problems](image)

5.9 Chapter Summary

The basic methodology was to:
1. embed machine-understandable, domain-specific knowledge within a knowledge-base.

2. combine locally-gathered data with publicly available datasets to provide new, high-level, useful information.

3. resolve conflict between information derived from different monitoring approaches.

The overall, step-by-step approach taken to the practical work can be very briefly summarised as follows:

1. Construct a library of general purpose, top-level reference ontologies for the networking domain, (*Loughborough Internetworking Ontology Library (LIOL)*), defining the major concepts and their relationships to each other.

2. Construct lower level, local ontologies for some specific case studies, with certain predefined issues in mind in each case.

3. In each case study where it is needed, capture appropriate local data, and where required also gather general Internet data.

4. Convert those data into standard form RDF triples, creating further triples to associate the data with concepts in the local ontology thus giving them meaning.

5. Merge the local and top-level ontologies, where necessary and appropriate.

6. Apply semantic reasoning, custom rules and queries within applications, in an attempt to achieve the wider objective of using the standard semantic web tools to improve understanding by stakeholders within the networking domain.

7. Analyse results and perform further iterations to test the utility and scalability of the applications.

The practical coding implementation was based on the following choices:

- *Jena* was found to be the most comprehensive semantic programming framework, meeting all the stated criteria and was therefore chosen as the principle semantic web tool-set to use for this research.

- *Eclipse* was chosen as the most appropriate and useful development environment for generating the Java classes required, being judged as flexible, reliable and easy to use.
The four problems to be addressed, as stated in Section 1.5 can be summarised as:

1. *The Syntactical Problem*; overcoming syntactical difference problems in disparate data.

2. *The Semantic Problem*; overcoming semantic problems in disparate data.


4. *The Conflict Resolution Problem*; resolving conflicts between data sources.

Two case studies were selected as appropriate to test the methodology against those problems:

1. *The vpn Study*; with “Organisation A”, a service organisation with branches around the UK and Ireland.

2. *The AS Topology Study*; internal to Loughborough University.
Chapter 6

Reference Ontology Implementation

6.1 Chapter Introduction

This chapter presents a detailed description of each individual autonomous ontology (AO) within the Loughborough Internetworking Ontology Library (LIOL), with comprehensive diagrams and test results. The concept of (and detailed methodology behind) the design of the library (LIOL) has been laid out previously in Section 5.6. This chapter begins in Section 6.2 by explaining what exactly has been created and delivered in practical terms to form the autonomous ontologies (AOS) of the Loughborough Internetworking Ontology Library (LIOL). These deliverables include a full RDF/XML serialisation, diagrams and code. There is then a section dedicated to each of the individual AOs which make up the library which has been created. In these individual sections, there is a detailed description of the rationale behind the design of each ontology, with ontology diagrams and test results for each. A total of six AOs are presented, addressing the specific areas of organisation, protocols, traffic, infrastructure, applications and monitoring, each with their own respective section.

6.2 Reference Ontology Concept and Content

Each of the Reference Autonomous Ontologies has been generated separately using Java classes written as part of this project, which import Jena packages as required. For each one there exists the following:

- A serialised RDF/XML file containing the ontology statements. These are listed in Appendix C.
• A Java archive (jar), which, when invoked will create the RDF/XML file.

• Java Source Code for the above.

• An Ontology Diagram, included in this chapter and described in the individual section dedicated to each particular ontology. The key to the symbols used in all these diagrams is shown in Appendix G.

• An Inference sample test listing, with queries and their result-sets which demonstrate the inference capabilities. These are again included in the section dedicated to each particular ontology.

• A test class hierarchy diagram, produced directly from the RDF/XML using Protégé; also included in the section dedicated to each particular ontology.

The ontology diagrams illustrate the AO classes and class relationships defined, but where large numbers of classes, properties or individual class members exist, to preserve visual clarity they may not all be shown there. It will be seen that some classes are created with the same name in more than one AO. This inclusion occurs when it is necessary to preserve the required autonomy of each AO and the meaning of the class is identical. It also provides a useful “hook” to link the ontologies when needed. When loading an AO into a model alongside local case-specific class hierarchies and instance data, it is sometimes necessary to assert new sub-class, sub-properties or other relationships, as well as class memberships of individuals to provide the necessary logical linking. When two or more AOs are loaded into such a model simultaneously, they must also be similarly linked to each other as needed by the application. This task is done by creating a small case-specific linking ontology (LO) for each application.

The inferences intended in each AO have been sample-tested using the General Methodology described in Section 5.7. They have been loaded into a test ontology model along with some test resources and statements. These models have then been reasoned over to produce an inference model, and queries made in the normal way. The code used to create the test statements, the queries and query result-sets are shown in each case.

An important feature of all these ontologies is that although they each provide a logically correct framework according to current conventions, they are not intended to be fully populated with all individual class members or necessarily to be correct for all time. In this sense they are meant to be extensible and subject to change in the following ways:

• New members can be added to the class extensions as required, for example a new protocol may be introduced, or a new measurement commissioned.
A complete new ontology may be required, alongside and linking with the current set if a new concept in the domain is invented.

New classes can be created within the existing class hierarchies, for example a new logical layer in the accepted protocol stack may be created.

The current structure may change, for example the authority structure within the governance of the Internet may be altered over time, meaning that the classes and properties in the Organisational Ontology may need to be correspondingly amended.

This built-in flexibility and acceptance of the inevitability of change in this domain of interest characterises the approach taken and also confirms the open-world ontology assumption as the correct one for this purpose.

### 6.3 The LIOL Organisation Autonomous Ontology

The Organisation AO, illustrated in Figure 6.1, is basically a class hierarchy or taxonomy, which describes the classes of organisation types which are involved as stakeholders, users and governance bodies of networking in general and the Internet in particular. The Protégé diagram in Figure 6.2 proves the structure to
be as intended. The first tier classes represent regulator (or governance) groups, ISPs, user organisations and research bodies. It is anticipated that any individual organisation involved in networking could be declared as part of the class extension of one or more of these classes, or their more specific sub-classes. In common with all of the LOILO ontologies though, it would be very easy to expand the scope in any new or unforeseen direction by adding new classes and properties. This would simply involve adding a few lines in the appropriate RDF/XML file. A number of individual organisations are included in this particular reference ontology because they are recognised bodies involved in Internet management and governance [26]. Also included at this level is a single property from the lboro namespace, which is used to assert the fact that the five Regional Internet Registry (RIR) organisations are lboro:overseenBy ICANN. This relationship is explicitly stated in the ontology for each RIR, rather than relying on inference using rdfs:domain and rdfs:range, leaving this property also available for other uses rather than being globally restricted. The five RIRs are also explicitly declared as members of the class lboro:rirOrg in the ontology (for clarity not show on the diagram). Inferences can be drawn and new entailment produced using the rdfs:subClassOf assertions. For example, lboro:ripe is asserted to be a member of the class lboro:rirOrg, but can also be inferred to be a member of the classes lboro:intGroupOrg, lboro:regulatorOrg and lboro:networkOrg. (see test results, Figure 6.3)
CHAPTER 6. REFERENCE ONTOLOGY IMPLEMENTATION

LILO Organisation Autonomous Ontology Tests

Test Statements and Resources:
Individual testCoLtd = endUserCo.createIndividual(defaultNameSpace + "testCoLtd");

Query 1:
Tests that Individual member lboro:ripe, an asserted member of class lboro:rirOrg
is inferred to be a member of the relevant super-classes.

queryStr.append (" select * where(lboro:ripe ?Property ?Object)");

Query Results

<table>
<thead>
<tr>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#rirOrg">http://nets.lboro.ac.uk/liol#rirOrg</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#overseenBy">http://nets.lboro.ac.uk/liol#overseenBy</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#ICANN">http://nets.lboro.ac.uk/liol#ICANN</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://www.w3.org/2002/07/owl#Thing">http://www.w3.org/2002/07/owl#Thing</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/2002/07/owl#sameAs">http://www.w3.org/2002/07/owl#sameAs</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#ripe">http://nets.lboro.ac.uk/liol#ripe</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#Resource">http://www.w3.org/2000/01/rdf-schema#Resource</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#regulatorOrg">http://nets.lboro.ac.uk/liol#regulatorOrg</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#intGroupOrg">http://nets.lboro.ac.uk/liol#intGroupOrg</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#networkOrg">http://nets.lboro.ac.uk/liol#networkOrg</a></td>
</tr>
</tbody>
</table>

Query 2:
Tests that a new (test only) member lboro:testCoLtd of class lboro:userOrg is also inferred to be
a member of the relevant super-classes.

Query Results

<table>
<thead>
<tr>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#endUserCo">http://nets.lboro.ac.uk/liol#endUserCo</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://www.w3.org/2002/07/owl#Thing">http://www.w3.org/2002/07/owl#Thing</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/2002/07/owl#sameAs">http://www.w3.org/2002/07/owl#sameAs</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#testCoLtd">http://nets.lboro.ac.uk/liol#testCoLtd</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#endUserCo">http://nets.lboro.ac.uk/liol#endUserCo</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/2002/07/owl#sameAs">http://www.w3.org/2002/07/owl#sameAs</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#endUserCo">http://nets.lboro.ac.uk/liol#endUserCo</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#networkOrg">http://nets.lboro.ac.uk/liol#networkOrg</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://nets.lboro.ac.uk/liol#networkOrg">http://nets.lboro.ac.uk/liol#networkOrg</a></td>
</tr>
</tbody>
</table>

Figure 6.2: Organisation AO Protégé Test

Figure 6.3: Organisation AO Inference Test
Figure 6.4: The LIOL Protocols Autonomous Ontology (See Appendix G for key to symbols)

6.4 The LIOL Protocols Autonomous Ontology

The principal super-class for the Protocols AO is lboro:networkProt which is the class of all protocols which could be encountered in the computer networking domain. The sub-class hierarchy follows the five layers of the well known Internet protocol stack [67]. Any conceivable protocol within the networking domain as presently defined could find a place within the class extension of one of the classes in the second tier of the ontology. Some of the well known protocols which are important in the case studies carried out in this research are included in both the rdf/xml ontology serialisation and the diagram; more could easily be taxonomised and included. The decision was made to create two sub-classes of application layer protocols, user protocols and routing protocols as a logical split in the functionality provided. Routing protocols however, are considered as belonging to the application layer within the hierarchy for the practical reason that they are encapsulated within the transport layer protocols within IP packets within networks, rather than as a comment on their function within any networked system. The Protégé diagram in figure 6.5 demonstrates that the class hierarchy is as intended. In the Protocols AO, inferences can be made and entailments produced from the rdfs:subClassOf assertions. For example the individual lboro:HTTPProt is
asserted as a member of lboro:userProt, but can in addition be inferred to be a member of lboro:appLayerProt and lboro:networkProt. In this ontology further inferences can be made using the rdfs:domain and rdfs:range assertions which are defined for the two properties created in the lboro namespace. These RDFS constructs are not used to provide type restrictions in the traditional programming sense, or to generate errors if non-sensical assertions are made, but rather to infer further information about any resource which uses the property to which they apply. They therefore provide very powerful reasoning but must be treated with caution. For example, if the following statement is asserted about two individual resources:

lboro:brandnewProt1  lboro:hasDefaultPort  lboro:newPort72

An RDFS or OWL reasoner will infer from the domain and range declarations that resource lboro:newProt1 must be a member of the class lboro:appLayerProt and that lboro:Port56 must be a member of the class lboro:Port. New entailments in the form of RDF triples will be generated accordingly. The following would be expected:

lboro:newPort72  rdf:type  lboro:Port
lboro:brandnewProt1  rdf:type  lboro:appLayerProt

In addition of course, the new protocol would logically be expected to be a member of all super-classes of class lboro:appLayerProt. This type of inference can be very useful if the entailments created are indeed true to the intentions of the programmer and the reality of the domain. If however, this property is used elsewhere within the model in a different context, unintended inferences may quite logically be drawn by a reasoner to produce non-sensical class memberships. Much care is therefore needed in the application of the rdfs:domain and rdfs:range axioms.
Figure 6.5: Protocols AO Protégé Test

Figure 6.6: Protocols AO Inference Test
Figure 6.7: The LIOL Traffic Autonomous Ontology (See Appendix G for key to symbols)

6.5 The LIOL Traffic Autonomous Ontology

The super-class of the Traffic AO is lboro:IPCommunication, which is the class of all communication instances using IP. This indicates that the interest here is at the network layer abstraction and above; this ontology does not address the operation of the physical or datalink layers. If that were required as a future extension of scope, a further super-class above with different branches would need to be added, but that is outside the scope of this research. Figure 6.7 shows the class and property hierarchy for the sub-class lboro:TCPFlow only, however the actual AO RDF/XML file also has similar assertions for the sub-class lboro:UDPFlow. These are omitted from the diagram to preserve clarity. The Traffic AO uses a range of inferencing techniques. There is a hierarchy of classes of communication instance types. A relatively large number of properties have been created, and in this AO it was appropriate to declare a sub-property hierarchy as well as rdfs:domain and rdfs:range assertions. Although relatively complex, there is also much symmetry here. As an example, the property hasTDstPort is used to assert that the resource which is the subject has a tcp destination port. The rdfs:domain is therefore declared as lboro:TCPFlow, because only a TCP flow would be the subject of such a predicate. If that particular TCP flow has a TCP destination port,
it follows that it must also have a destination port and a port. A sub-property hierarchy is therefore asserted which follows this pattern of becoming less specific each time a higher level is reached. The branch of the ontology described is typical of the other branches; its test results are shown in figure 6.9, where two newly created resources \texttt{lboro:newFlow102} and \texttt{lboro:newPort22} are declared to be related by the property \texttt{lboro:hasTDstPort}. The tests demonstrate that these are inferred to be a \texttt{lboro:TCPFlow} and a \texttt{lboro:Port} respectively, and also that the appropriate super-properties are also inferred to relate the two resources.
LILO Traffic Autonomous Ontology Tests

Test Statements and Resources:

```java
OntModel ontquerymodel = ModelFactory.createOntologyModel(OntModelSpec.OWL_LITE_MEM_RULES_INF, querymodel);
Resource newFlow102 = ResourceFactory.createResource(defaultNameSpace + "newFlow102");
Property hasTDstPort = ontquerymodel.createProperty(defaultNameSpace + "hasTDstPort");
ontquerymodel.createResource(newFlow102);
Statement stmt = ontquerymodel.createLiteralStatement(newFlow102, hasTDstPort, 22);
ontquerymodel.add(stmt);
```

Query 1:
Tests that new resource lboro:newFlow102 is inferred to be a lboro:TCPFlow by the rdfs:domain of its property lboro:hasTDstPort. Also tests that it is related to the literal port number 22 not only by the asserted property, but also by the super-properties lboro:hasDstPort and lboro:hasPort

```java
queryStr.append(" select * where{lboro:newFlow102 ?Property ?Object} ");
```

<table>
<thead>
<tr>
<th>property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#hasTDstPort">http://nets.lboro.ac.uk/liol#hasTDstPort</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#TCPFlow">http://nets.lboro.ac.uk/liol#TCPFlow</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#hasDstPort">http://nets.lboro.ac.uk/liol#hasDstPort</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#hasPort">http://nets.lboro.ac.uk/liol#hasPort</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#TCPFlow">http://nets.lboro.ac.uk/liol#TCPFlow</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#hasDstPort">http://nets.lboro.ac.uk/liol#hasDstPort</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#hasPort">http://nets.lboro.ac.uk/liol#hasPort</a></td>
<td>&quot;22&quot;<a href="http://www.w3.org/2001/XMLSchema#int">http://www.w3.org/2001/XMLSchema#int</a></td>
</tr>
</tbody>
</table>

Figure 6.8: Traffic AO Protégé Test

Figure 6.9: Traffic AO Inference Test
6.6 The LIOL Infrastructure Autonomous Ontology

Although there is a top-level class defined of \texttt{lboro:network}, (the class of all networks), the main super-class is the \texttt{lboro:netElements} class. It is the class of all elements which can form part of a network. The first tier of sub-classes represent nodes and links. This ontology is concerned with networking within the network layer abstraction, and so the class \texttt{lboro:node} can be split into sub-classes which are all “IP aware”, \texttt{lboro:serverHW}, \texttt{lboro:clientHW} and \texttt{lboro:router}. The members of the class \texttt{lboro:LAN}, for example are not nodes in this sense because they are not IP aware. A number of properties are defined concerning links; these are all ultimately sub-properties of \texttt{lboro:linksTo}, and are formed in a hierarchy correspondingly. It was recognised as a requirement to be able to model the situation where any node is plugged into a LAN, and infer in the model the links which that creates. This is achieved using a symmetric property as a sub-property of a transitive property. If symmetric property \texttt{lboro:DataLink} has a subject which is a node of some sort and an object which is a LAN, then it applies symmetrically and so models a bi-directional link, as required. Note that \texttt{rdfs:domain} and \texttt{rdfs:range} cannot be used here, because that would quite
logically infer in the model that the subject and object are both node and LAN at
the same time. Because this symmetric property is a sub-property of the transitive
property localLink, this adds the transitive quality to the relationship. This
means that not only does the node have a bi-directional link with the LAN, the
link is also transitive to any other node which has a link to the same LAN. This
is the representation of the real-world situation which is required. One special
sub-class of the lboro:network class is specified. lboro:autonSystem, which
is the class of all Internet autonomous systems (AS). The domain of property
lboro:hasASNum will infer that any resource which is the subject of it is inferred
to be a member of lboro:autonSystem, and any resource given as its object must
be a member of class lboro:ASNumber.
LILO Infrastructure Autonomous Ontology Tests

Test Statements and Resources:

```java
OntClass serverHW = ontModel.getOntClass(defaultNameSpace + "serverHW");
OntClass clientHW = ontModel.getOntClass(defaultNameSpace + "clientHW");
OntClass LAN = ontModel.getOntClass(defaultNameSpace + "LAN");
Individual VLAN6 = LAN.createIndividual(defaultNameSpace + "VLAN6");
Individual mailServer = serverHW.createIndividual(defaultNameSpace + "mailServer");
Individual iansClient = clientHW.createIndividual(defaultNameSpace + "iansClient");
SymmetricProperty dataLink = ontModel.getSymmetricProperty(defaultNameSpace + "dataLink");
Statement stm1 = ResourceFactory.createStatement(mailServer, dataLink, VLAN6);
Statement stm2 = ResourceFactory.createStatement(iansClient, dataLink, VLAN6);
ontModel.add(stm1);
tonModel.add(stm2);
```

Query 1:
Tests many of the inferencing capabilities of this ontology.
Tests for resources which are connected by transitive property lboro:localLink AND are both members of class lboro:node. This tests the inference through the class hierarchy, because they are both created only as individuals of different sub-classes of lboro:node. It tests the sub-property inferencing, because they are declared as subject and object respectively only of sub-properties of lboro:localLink. It tests that the combination of symmetric and transitive properties accurately simulates the behaviour of two devices plugged into a LAN; bidirectional and fully connected.

```java
```

Query Results
```
+----------+----------+
| Node1    | Node2    |
|----------+----------|
|<http://nets.lboro.ac.uk/liol#iansClient> | <http://nets.lboro.ac.uk/liol#iansClient> |
|<http://nets.lboro.ac.uk/liol#iansClient> | <http://nets.lboro.ac.uk/liol#mailServer> |
|<http://nets.lboro.ac.uk/liol#mailServer> | <http://nets.lboro.ac.uk/liol#iansClient> |
|<http://nets.lboro.ac.uk/liol#mailServer> |<http://nets.lboro.ac.uk/liol#mailServer> |
```

Figure 6.11: Infrastructure AO Protégé Test

Figure 6.12: Infrastructure AO Inference Test
Figure 6.13: The LIOL Applications Autonomous Ontology (See Appendix G for key to symbols)

6.7 The LIOL Applications Autonomous Ontology

The Applications AO is not intended to be a taxonomy of possible applications, however the class lboro:appType is the class of all application types. There is no sub-class hierarchy. The most significant class in the ontology is lboro:appImpl, the class of all individual application implementations. This class is then declared as the rdfs:domain of a series of object properties. Each of these object properties has rdfs:range asserted as one of the classes from the other AOs in the library, thus linking the application implementation into the network. As an example, an individual web server may be of application type apache:apacheWebServer, it may be hosted on lboro:webServer2 hardware, use application layer protocol lboro:HTTPProt, use transport layer protocol lboro:TCPProt, overriding the standard port and using lboro:port8080. If these properties were all used in statements as the object of the respective properties, they would all immediately have meaning within the ontology as a consequence of the rdfs:range declarations.
LILO Applications Autonomous Ontology Tests

Test Statements and Resources:

Property hostedOn = ontquerymodel.createObjectProperty(defaultNameSpace + "hostedOn");
Property implPort = ontquerymodel.createDatatypeProperty(defaultNameSpace + "implPort");
Resource webServer2 = ontquerymodel.createResource(defaultNameSpace + "webServer2");
Resource ourWebServer = ontquerymodel.createResource(defaultNameSpace + "ourWebServer");
Statement stmt1 = ontquerymodel.createStatement(ourWebServer, hostedOn, webServer2);
Statement stmt2 = ontquerymodel.createLiteralStatement(ourWebServer, implPort, 8080);
ontquerymodel.add(stmt1);
ontquerymodel.add(stmt2);

Query 1:
Tests that new resource lboro:ourWebServer is inferred to be a member of class lboro:appimpl
by the rdfs:domain of property lboro:hostedOn
queryStr.append("select * where{?Subject rdf:type lboro:appImpl}");

<table>
<thead>
<tr>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://nets.lboro.ac.uk/liol#ourWebServer">http://nets.lboro.ac.uk/liol#ourWebServer</a></td>
</tr>
</tbody>
</table>

Query 2:
Tests the inference provided by the rdfs:domain of property lboro:implPort in that it caused resource
lboro:ourWebServer to be inferred as a member of class lboro:appImpl
queryStr.append("select * where{lboro:ourWebServer ?property ?object}");

| property | object |
|-----------------------------------------------|
| <http://nets.lboro.ac.uk/liol#implPort> | "8080" "<http://www.w3.org/2001/XMLSchema#int> |
| <http://nets.lboro.ac.uk/liol#hostedOn> | <http://nets.lboro.ac.uk/liol#webServer2> |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://nets.lboro.ac.uk/liol#appImpl> |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://www.w3.org/2002/07/owl#SameAs> |
| <http://www.w3.org/2000/01/rdf-schema#Resource> | <http://nets.lboro.ac.uk/liol#OurWebServer> |

Figure 6.15: Applications AO Inference Test
6.8 The LIOL Monitoring Autonomous Ontology

The Monitoring AO has three small, discrete class hierarchies. In one, the class `lboro:DataSource` has two sub-classes representing the classes of local data sources and external data sources respectively. One asserts that the class of monitoring application implementations, `lboro:M3AppImpl` is a subset of `lboro:AppImpl`, the class of all application implementations. (Note: M3 is used as an abbreviation for “Monitoring, Measurement and Management”). The third hierarchy has two sub-classes of `lboro:Tool`; `monitorTool`, the class of tools which directly “touch” the network to gather data, and `processTool`, the class of tools which are used to process data to produce `metrics`, or higher-level information of interest to stakeholders from that data. Like the Applications AO described in Section 6.7, further inference capability is introduced by the use of a number of object properties with `rdfs:range` and `rdfs:domain` assertions.

6.9 Chapter Summary

The Loughborough Internetworking Ontology Library (LIOL) has been created and initially contains six reference autonomous ontologies (AOs) addressing six different
aspects in the domain of Computer Networking:

- Organisation.
- Protocols.
- Traffic.
- Infrastructure.
- Applications.
- Monitoring.

For each of these ontologies the following have been created and are available:

- A serialised RDF/XML file containing the ontology statements.
- A Java archive (jar), which, when invoked will create the RDF/XML file.
- Java Source Code for the above.
- An Ontology Diagram.
- An Inference sample test listing.
- A test class hierarchy diagram.
LILO Monitoring Autonomous Ontology Tests

Test Statements and Resources:

Property sendsLocDataTo = ontModel.getObjectProperty(defaultNameSpace + "sendsLocDataTo");
Property metricUsedBy = ontModel.getObjectProperty(defaultNameSpace + "metricUsedBy");
Resource tcpdump = ResourceFactory.createResource(defaultNameSpace + "tcpdump");
Resource iansPcap = ResourceFactory.createResource(defaultNameSpace + "iansPcap");
Resource tripleStore1 = ResourceFactory.createResource(defaultNameSpace + "tripleStore1");
Resource userApp33 = ResourceFactory.createResource(defaultNameSpace + "userApp33");
Statement stm1 = ResourceFactory.createStatement(tcpdump,sendsLocDataTo,iansPcap);
Statement stm2 = ResourceFactory.createStatement(tripleStore1,metricUsedBy,userApp33);
tonModel.add(stm1);
tonModel.add(stm2);

Query 1:
Tests that new resource lboro:iansPcap is inferred to be a member of class lboro:locDataSource
by the range of property lboro:sendLocDataTo, and also that it is inferred to be a member of class
lboro:dataSource, by the class sub/super classing hierarchy.
queryStr.append (" select * where{lboro:iansPcap ?Property ?Object}");

Query Results
| Property                                          | Object                                          |
|-------------------------------------------------------------------------------------------------------|
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://www.w3.org/2002/07/owl#Thing>           |
| <http://www.w3.org/2002/07/owl#sameAs>            | <http://nets.lboro.ac.uk/liol#iansPcap>         |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://www.w3.org/2000/01/rdf-schema#Resource> |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://nets.lboro.ac.uk/liol#locDatSource>     |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://nets.lboro.ac.uk/liol#dataSource>       |
|-------------------------------------------------------------------------------------------------------|

Query 2:
Tests that being the object of property lboro:metricUsedBy creates the inference that new resource
lboro:userApp33 isa member of class lboro:m3AppImpl, and also by the class structure a member of
class lboro:appImpl
queryStr.append (" select * where{lboro:userApp33 ?Property ?Object}");

Query Results
| Property                                          | Object                                          |
|-------------------------------------------------------------------------------------------------------|
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://www.w3.org/2002/07/owl#Thing>           |
| <http://www.w3.org/2002/07/owl#sameAs>            | <http://nets.lboro.ac.uk/liol#UserApp33>        |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://www.w3.org/2000/01/rdf-schema#Resource> |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://nets.lboro.ac.uk/liol#AppImpl>          |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> | <http://nets.lboro.ac.uk/liol#m3AppImpl>        |

Figure 6.17: Monitoring AO Protégé Test

Figure 6.18: Monitoring AO Inference Test
Chapter 7

The VPN Case Study

7.1 Chapter Introduction

This chapter describes in detail the practical research work carried out in the VPN Case Study. Sections 7.2 and 7.3 give the background to the organisation and describe the VPN which is overlayed on the public Internet in some detail. Section 7.4 then goes on to explain the approach taken to the study, including the process used to build the necessary knowledge base. Sections 7.5 and 7.6 describe the data collection process in some depth, discussing the tools and techniques used to gather raw data and then convert the data obtained into RDF triples. Here the research hardware is specified and code snippets are included. This conversion process is then evaluated in Section 7.7. Having completed the basic data conversion to produce datasets in the standard RDF triple format, in Section 7.8 the design and function of the VPN Local Linking Ontology is described; this builds on the autonomous ontologies (AOS) within the Loughborough Internetworking Ontology Library (LIOL) to produce a knowledgebase framework custom fitted to this study. Section 7.9 reports on the way the resulting knowledgebase was applied to answer practical questions about the performance of the network from the base data gathered earlier. Section 7.10 is dedicated to the explaining the use of semantic rules, specifically JenaRules in the study. In particular, rules are demonstrated as a method of embedding expertise in the knowledgebase, to facilitate automatic diagnosis of pathologies and remedies to them. Section 7.11 reviews complete case study, summarising what has been achieved and demonstrated.

7.2 Background and Motivation

Organisation-A is a not-for-profit service organisation, with a customer base of some 30,000 people. The organisation has its headquarters in London, with satel-
lite branch offices in various parts of the UK and one in The Republic of Ireland. Data communication between these offices is achieved using a Virtual Private Network (VPN), overlayed on the public Internet. Technical Resources Midlands Ltd (TRML)\(^1\), a specialist IT support company, originally set up and now maintain this overlay network and other aspects of the IT infrastructure for them. In a meeting held with TRML staff, minuted in Section A.1, the Network Administrator supporting Organisation-A raised several important issues, which have motivated and helped to shape the direction of this research work. Whilst in his view the use of a VPN over the Internet is very convenient and cost effective, problems do occur from time to time and resolving them can be difficult. The issues raised can be summarised into three core problems:

- The high level of knowledge needed to understand the set-up means that junior staff at TRML are often unable to diagnose faults, which are then referred to senior staff, who would ideally be working on other tasks.

- Fault diagnosis often involves a time consuming process of tests which currently have to be performed manually. There is much repetition involved.

- The network configuration changes frequently, sometimes causing records to be out of date. This is a further barrier to diagnosing the cause of faults when they are reported by users.

- When the root cause of the problem is not local but related to the Internet (and connections to it), TRML are totally dependent on the various Internet Service Providers (ISPs) for information. There is a perception that no single ISP ever admits liability, and the only solution is to wait until the problem eventually goes away, usually with no satisfactory explanation or admission of responsibility. ISP responses can be slow and unpredictable.

Each office location connects to the Internet using one or more Asynchronous Digital Subscriber Line (ADSL) enabled copper-pair based telephone lines. In every case, the line is supplied by the national provider. The ADSL service itself is provided by a variety of ISPs, chosen locally on a cost and nominal bit rate basis. It has been established in Chapter 1 using information from the UK Government Office of National Statistics (ONS)\(^{[18]}\) that ADSL is easily the most popular way for businesses in the UK to connect to the Internet, accounting for 86% of connections in 2012. The Organisation A network therefore offers a suitable environment to conduct research into the use of ontologies and other semantic web tools to solve actual problems as described by the network administrators in a typical real-world,

\(^{1}\)http://www.trml.co.uk Last viewed September 2013
small to medium company setting. Organisation A and TRML both kindly agreed to support this research by allowing a research server to be installed at the London headquarters, with connectivity to the main router there and remote access for the researcher via a research VPN tunnel. The research server has been used to carry out both passive and active measurement activity on the VPN, and to run the local elements of the application.

### 7.3 The Organisation A Network

The topology of the organisation’s network reflects its administrative structure. There is a strong focus on the London headquarters, where most of the application servers and the majority of staff are based. Satellite branch offices are located in York, Manchester, Bristol, Glasgow and Dublin. The branch offices have no operational need for direct communications between themselves, only with the London headquarters, so a star-type network topology has been set up by TRML with London at the centre. The network topology is shown in figure 7.1. Each site has a Draytek router\(^2\) connecting it to the ISP via an ADSL enabled line supplied by BT. Various Draytek models are used, and sometimes need to be replaced with more

\(^2\)http://www.draytek.co.uk/mainmenu.html last viewed September 2013
current models when hardware faults occur. The VPN functionality is compatible between models, and configuration images can be saved and reloaded as needed. A VPN tunnel using IPsec has been configured between the London Draytek router and each of the counterpart routers located at the satellite branches. These tunnels are “built” when the routers are booted up. Tunneling using IPsec is described in some detail by Snader [63]. In summary though, if a packet arrives at the router internal interface with a destination IP address in a subnet configured to be at the other end of a particular tunnel, the packet, including its payload is encrypted and then encapsulated in an outer packet whose destination IP address is the external address of the router at the other end of the tunnel and whose source IP address is the external address of the local router. This packet is then forwarded through the Internet in the normal way. On receipt, the receiving router recognises the source address as that of the distant end of one of its tunnels, strips off the encapsulation, decrypts the packet and forwards it through its internal interface using the original destination address. Thus the idea of a physical tunnel as such is entirely notional, there is no separate connection through the Internet in the way the term tunnel may imply. This is one of the expertise problems which TRML have; junior staff tend to have the simplistic view of a tunnel as a kind of isolated, separate path through the Internet, which makes it difficult for them to diagnose connectivity failures on the VPN. The router at the London office was type Vigor 2950, which has a mirror port on the built-in switch. All packets received by any of the built in switchports are sent out on this mirror port, so it is a suitable place to observe or capture all packets on the VPN going to or coming from any of the satellite branches. The agreement with TRML was that only packet headers as far in as the Transport Layer would be captured, with no observation or recording of Application layer payloads. The research server was connected to the mirror port to enable packet capture and also for remote access. Traffic between nodes within the London site could not be observed at this point, but active probing of them using ping and other utilities was possible from the research server. The London office also has several other connections used for different purposes, but they are not relevant to this research. There is a single logical local network at each site, each occupying a private IP address space with /24 subnet mask, as shown in the diagram. No VLANs are used; the multiple network switches shown at the London site are there to fully connect nodes on different floors in the building and are simply daisy-chained together using the trunk ports. The real external public IP addresses were used in the research, and were indeed necessary to relate data gathered locally to external data sources but they are omitted from the diagram by agreement with TRML for security reasons. The number of client machines at each site varies and is actually unknown in any detail by TRML, however some
such information which was available linking client IP addresses to user names etc has been included in the modeling.

### 7.4 Methodology

The General Methodology described in Section 5.7 was used as the basis for the VPN case study, adapted to suit the Organisation-A environment and the research issues which were to be addressed. The more detailed case-specific methodology for the first part of the study (data-gathering, data-conversion and creation of the knowledgebase) is illustrated in figure 7.2. A variety of disparate data sources were combined into an ontology model-based application, which can be used to provide useful information to TRML network managers. The system was built up on a step-by-step basis and at each milestone tests were performed to assess how well the interpretation problems stated in Section 5.8 were being addressed. The Syntactical Problem and the Semantic Problem were examined at various stages by utilising the available data in ways that would have presented difficulties without using the approach made possible by the semantic web languages and tools. The Expertise Problem could only be examined in the final stages, when different reasoners, rules and queries were used to help with real-life network management problems. Scaling issues were also assessed where possible at each stage, both in terms of
handling large numbers of RDF triples and by testing the effects of using inference when querying the resulting models. Some of the data used were obtained locally, using both active and passive monitoring and measurement techniques from the research server, but also combined with publicly available data to give a broader view of the performance on the VPN as a whole.

7.5 Data Sources and Collection

The research server was a Viglen Genie machine, with 2.8 GHz Pentium 4 CPU and 1 GByte of memory. The operating system was Ubuntu vers 9.10. This machine was used to obtain all the local data for the research. Once installed, physical access to the research server was very difficult, as it was housed in a secure room at a central London location. One of the implicit agreements was that the research wouldn’t cause any disruption or nuisance to the Organisation A staff, so it was not possible to ask anyone on-site to do manual reboots or other tasks. The only physical access was with TRML engineers when they were visiting for other operational reasons. Much care was therefore needed in all remote interactions to avoid putting the server into an unreachable condition. Passive capture of packet headers was set up by enabling the mirror port on the London Draytek router and putting the network adaptor on the research server into promiscuous mode. The well known standard tcpdump utility was then invoked as a background process using the linux at job scheduling function. The tcpdump options were used to:

- capture packets from eth0, the main network adaptor.
- capture only packets with source or destination addresses in the subnets of the remote VPN sites (as agreed with TRML).
- send the captured headers pcap files, changing to a new file each 24 hours (86,400 seconds).

The command used was:

```
tcpdump -i eth0 -w /home/test/packet_traces/orga.%s -G 86400 net 192.168.20 or net 192.168.30 or net 192.168.40 or net 192.168.50 or net 192.168.60
```

This produced a set of one-day duration packet capture files of normally between 100 MBytes and 250 Mbytes, which were then compressed and copied back from the research server to the lab overnight so that no congestion problems were caused. A continuous capture for five weeks was achieved, plus some shorter durations. This fairly large dataset was suitable as the basis for scalability testing as well as to add network performance information to the model.

In addition to the passive monitoring, various active probes were sent out on an
hourly basis under the control of the research server. This was set up using the `expect` automation tool to manage the interactions between server and router and implement tests using `ping` and `traceroute`. The resulting output was then timestamped and directed to a text file. Scheduling was handled by entering the `expect` file as a regular job in the crontab. Some tests were performed directly from the research server interface, and some from the Draytek router by issuing commands to it over its telnet interface. This particular aspect had to be handled very carefully to avoid any inadvertent damage to the router settings which would have caused serious operational problems. A sample of the entries from the `expect` file is shown:

```
#!/usr/bin/expect -f
set timeout 240
spawn date +%s
expect "ian@ian-test:~$"
spawn ping -c 5 192.168.20.254
expect "ian@ian-test:~$
spawn ping -c 5 192.168.10.130
expect "ian@ian-test:~$
spawn telnet 192.168.10.254
expect "Password:"
send "*******\n" expect ">
send "ip ping 217.206.141.218\n" expect ">
send "ip tracert 217.206.141.218 WAN2\n" expect ">
send "exit\n"
expect "ian@ian-test:~$
spawn echo "ended safely!"
interact
```

The full `expect` command file executes:

- a ping test from the server interface to the *internal* interface of the routers at each satellite branch. A successful test requires the appropriate VPN tunnel to be working.

- a ping test within the London subnet to the main application servers.

- a ping test from the *external* interface of the router to the *external* interfaces of the routers at each satellite branch.

- a traceroute test from the *external* interface of the router to the *external* interfaces of the routers at each satellite branch.

The `traceroute` utility is described in Section 2.5, as are some of its limitations as a network diagnostic tool. Although not a perfect solution, `traceroute` gives at least a partial view of what is happening between the two points in the internet, and has been used here to give a view of the routing between the London server and the satellite branch servers. This is potentially useful to TRML in their dealings with the various ISPs used as any faults can be discussed, or even argued, from an

³http://expect.sourceforge.net/ Last viewed September 2013
informed viewpoint. It could also be a factor in ISP selection decisions.

The IP address information for each hop gives little useful information on its own, but can be enhanced by mapping those IP addresses to their Autonomous System numbers and thereby to the organisations who operate those networks. Therefore external, openly available data from Team Cymru\(^4\), discussed in Section 2.6 has been utilised to perform that mapping. This was another test of the ontological approach in bringing together disparate data sources with different syntax and meaning to produce useful information. The Team Cymru IP to AS mapping service was accessed using the netcat\(^5\) utility by submitting a list of IP addresses in a text file. A second list file was returned containing the mapping information. This file, along with the locally gathered raw data files was used as the input to the appropriate data conversion stage.

### 7.6 Data Converters

The basic method employed for converting the data was to write new Java classes, with imports from the various Jena archives and other generic Java classes. The methods in these new classes were then used to read in the raw data in its various formats item-by-item, create RDF resources, properties and literals from those items as appropriate and then form these into RDF statements (triples) for persistent storage in a Jena TDB disk backed triplestore or, for smaller numbers of triples, RDF/XML serialisation in a file. It is important to note that even at this level of triple storage, before any linking to the LIOL reference ontologies, some ontological statements, such as class membership declarations using rdf:type were included as “hooks” to enable the instance triples to be linked-in and given meaning later on in the process. To further investigate any potential problems with alignment of ontologies, a different XML namespace was used for each case study, rather than taking the easier option of using the lboro namespace throughout.

The PCAP converter utilises the JPCAP\(^6\) package to read in each packet from the pcap files sequentially. Jpcap is a Java wrapper for the standard Libpcap library, which allows the individual fields of the packet header to be accessed from within the Java code. The information of interest is then processed using the methods available through Jena to form it into meaningful RDF statements in

\(^4\)https://www.team-cymru.org/ Last viewed September 2013
\(^5\)http://netcat.sourceforge.net/ Last viewed September 2013
\(^6\)https://sites.google.com/site/sipinspectorsite/download/jpcap viewed Sep 2013
\(^7\)http://www.eden.rutgers.edu/~muscarim/jpcap/tutorial viewed Sep 2013
Listing 7.1: Packet-level PCAP to RDF Conversion snippet

```java
//set up tdb model and prefix
Dataset ds = TDBFactory.createDataset("/home/ian/pcapconv1_tdb");
Model vpnmodel2 = ds.getDefaultModel();
OntModel ontvpnmodel2 = ModelFactory.createOntologyModel(OntModelSpec.
OWL_LITE_MEM, vpnmodel2);
ontvpnmodel2.setNsPrefix(prefix, orgaNameSpace);
//Ontology statement creating an owl class for packets
OntClass modpacket = ontvpnmodel2.createClass(orgaNameSpace + "Packet");
//Showing SOME Properties being created
DatatypeProperty hasDstIP = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + "hasDstIP");
ObjectProperty hasSrcPort = ontvpnmodel2.createObjectProperty(orgaNameSpace + "hasSrcPort");
DatatypeProperty hasTestTime = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + "hasTestTime");
//Open the pcap file with new captor object and get packets
JpcapCaptor captor = JpcapCaptor.openFile("/home/ian/tssadata/orgacap.1310598603");
while(true){
    //read a packet from the opened file
    packet = captor.getPacket();
    ippacketcount++;
    long date = (packet.sec);
    //Create Triples for this packet
    Individual ind = modpacket.createIndividual(orgaNameSpace + "Packet4" + ippacketcount);
    ontvpnmodel2.add (ind, hasDstIP, ippacket.dst_ip.getHostAddress());
    ind.addLiteral(hasTestTime, date);
    if (packet instanceof TCPPacket){
        tcppacket = (TCPPacket)packet;
        tcppacketcount++;
        ind.addLiteral(hasSrcPort, tcppacket.src_port);
    }
}
```

the Jena TDB dataset. The packet capture files from the Organisation A network each contained a 1 day slice of all packet headers passing through the VPN and were around 150 MBytes to 250 MBytes in size. A rough estimate of 100 Bytes per packet header indicated that between 1.5 million and 2 million packets in total were passing through the combined tunnels each day. Two distinct methods were tried to convert the data representing these into RDF statements:

**Packet-level conversion.** Using this method, a set of up to six RDF statements characterising each individual packet were created and stored in the model. Each packet has a unique resource name derived from a name given for each file and the sequence number of the packet as read in from that file. All packets were associated with source and destination IP addresses, a timestamp and an ontological statement that they were members of the class orga:packet. Those identified as either TCP or UDP packets also had source and destination port numbers. The following code listing is not complete, but is a combination of example snippets of the converter code showing the most important statements for illustration is shown in Listing 7.1

**Flow-level conversion.** Using this method, packets are identified as carrying transport layer protocol payloads of either TCP, UDP or ICMP. They are then aggregated into flows and characterised as such by an ontological statement identifying them as members of the class orga:TCPFlow, orga:UDPFlow
or orga:ICMPComm as appropriate. Each flow is then described by a number of RDF statements derived directly from the Jpcap packet fields. In this method, ICMP packets have the potential to directly provide information on the prevailing network conditions, so although not strictly flows, they are also included in the flow-level conversion. Some aggregation was also possible here though, by simply using the natural RDF statement 3-dimensional space concept illustrated in figure 3.3. When a number of a particular type of ICMP message packets were observed between the same two points, it was only necessary to add a single triple each time after the first, with an extra timestamp as its object, rather than a complete set of triples characterising every such packet. The code listing shown in Listing 7.2 is not complete, but is a combination of example snippets of the converter code showing the most important statements for illustration.

In the usage planned for the packet header data, it was not envisaged that fine-grain detailed packet-level information would be required by the models created later in the process. This conversion exercise was carried out as an experiment in using the relatively large amount of data available to test the scaling issues and efficiency of storing network data in an RDF triple store. The flow-level was judged to be the right level of abstraction to give input to the subsequent inferencing and querying processes and so this was used as the input to the later stage of the modeling. It should be acknowledged that both approaches are lossy to some extent. The packet-level converter is lossy because although each packet is recorded as an individual in the triple store, only those fields available from JPcap which were deemed of interest were included as resources and properties. Thus some information about each packet is lost. This could easily be amended at the cost of creating more potentially useless statements for each packet. The flow-level converter loses the individuality of packets in the store as well as some of the JPcap fields which were not used. Packets become a simple count in the flow information. The counterpoint to this is that even by simply forming the JPcap fields which represent the packets into a logical set of RDF triples, a small amount of domain-specific knowledge has been embedded, some meaning has been added and the raw data has been converted into more useful and accessible information. Low-level detail, which is of no use to any application later on has been filtered out using domain-specific knowledge in favour of slightly higher-level information. If at a later date the needs of the modeling and any user applications were to change and the lost data were required, the original pcap dump files could be processed differently.

The ping-trace converter was written to convert the active measurement results
Listing 7.2: Flow-level pcap to RDF Conversion snippet

```java
//create ontology classes to "hook in" later
OntClass TCPFlow = ontvpnmodel2.createClass(orgaNameSpace + "TCPFlow");
OntClass UDPFlow = ontvpnmodel2.createClass(orgaNameSpace + "UDPFlow");
OntClass ICMPComm = ontvpnmodel2.createClass(orgaNameSpace + "ICMPComm");
//Shoving some Object Properties created
ObjectProperty hasDstIP = ontvpnmodel2.createObjectProperty(orgaNameSpace + " hasDstIP");
ObjectProperty hasSrcPort = ontvpnmodel2.createObjectProperty(orgaNameSpace + " hasSrcPort");

//Some of the datatype properties datatype Properties
DatatypeProperty hasIPValue = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + " hasIPValue");
DatatypeProperty hasIPSubnet = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + " hasIPSubnet");
DatatypeProperty hasPacketCount = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + " hasPacketCount");
DatatypeProperty hasTestTime = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + " hasTestTime");
DatatypeProperty hasICMPType = ontvpnmodel2.createDatatypeProperty(orgaNameSpace + " hasICMPType");

//Read in Packets
packet = captor.getPacket();
if (packet instanceof IPPacket){
    ippacket = (IPPacket)packet;
    if (ippacket.IPPROTO_IP == 4){
        ippacketcount++;
        long date = (packet.sec);
        if (packet instanceof TCPPacket){
            tcppacket = (TCPPacket)packet;
            //Manipulation of ip addresses to generate subnet mask (knowing all are /24 here)
            byte[] srcIPValue = ippacket.src_ip.getAddress();
            byte[] dstIPValue = ippacket.dst_ip.getAddress();
            byte[] srcSubnet = srcIPValue;
            srcSubnet[3] = 0;
            InetAddress srcSub = Inet4Address.getByAddress(srcSubnet);
            byte[] dstSubnet = dstIPValue;
            dstSubnet[3] = 0;
            InetAddress dstSub = Inet4Address.getByAddress(dstSubnet);
            //Some of the resources and statements for this flow
            Resource srcIPAddress = ontvpnmodel2.createResource(orgaNameSpace + ippacket.src_ip.getHostAddress());
            Resource dstIPAddress = ontvpnmodel2.createResource(orgaNameSpace + ippacket.dst_ip.getHostAddress());
            Resource srcIPAddress = ontvpnmodel2.createResource(orgaNameSpace + ippacket.src_ip.getHostAddress());
            Resource dstIPAddress = ontvpnmodel2.createResource(orgaNameSpace + ippacket.dst_ip.getHostAddress());
            //flowid combines dst and src IP addr and ports to give unique id for each flow
            flowid = ippacket.dst_ip.getHostAddress() + "_" + Integer.toString(tcppacket.dst_port) + "_" + ippacket.src_ip.getHostAddress() + "_" + Integer.toString(tcppacket.src_port);
            //Can "create" the flow with every packet, only one in model (3-D model concept)
            Individual ind = TCPFlow.createIndividual(orgaNameSpace + flowid);
            int newcount = 1;
            ontvpnmodel12.add (ind, hasDstIP, dstIPAddress);
            ontvpnmodel12.add (ind, hasSrcPort, Integer.toString(tcppacket.dst_port));
            Statement sip2 = ontvpnmodel2.createStatement(dstIPAddress, hasIPValue, ippacket.dst_ip.getHostAddress());
            ontvpnmodel2.add(sip2);
            Statement sip4 = ontvpnmodel2.createStatement(dstIPAddress, hasIPSubnet, dstSub.getHostAddress());
            ontvpnmodel2.add(sip4);
            //Process first packet received in this flow, identified by absence of a property in model
            if (ind.hasProperty(hasPacketCount)==false){
                tcpflowcount++;
                ind.addLiteral(hasTestTime, date);
                ind.addLiteral(hasReset, false);
            }
            //Process all other packets in this flow
            //Remove old packet count statement, increment value and recreate outside loop.
            if (ind.hasProperty(hasPacketCount)){
                Statement = ind.getProperty(hasPacketCount);
                int oldcount = s.getLiteral().getInt();
                newcount = oldcount + 1;
                ontvpnmodel12.removeAll(ind, hasPacketCount, null);
                //IF THIS PACKET has reset flag set, add the property for the flow
                if (((tcppacket.rst)&&(ind.hasProperty(hasReset)))
                        Statement s1 = ind.getProperty(hasReset);
                        if (!s1.getBoolean() == false)
                            ontvpnmodel12.removeAll(ind, hasReset, null);
                ind.addLiteral(hasReset, true);
                resetcount++;
            }
        }
    }
}
```

which were stored in text files. The output from the various ping and traceroute
tests were appended to a single file each hour, for the same five week period during
which the main packet capture exercise was performed. The converter was written
in two parts to make it more widely usable in other future cases. In the first stage,
the Java scanner class is used, along with some regular expressions to parse the
file and produce an interim text file with one line for each test. In this sense, a test
would be either a group of (normally five) ping echo requests, or an end-to-end
traceroute, with all the relevant information for each. Ping echo requests were
sent in groups of five when originating from any of the Draytek routers in the net-
work, because the telnet interface was rather limited and this was a standard test
which was not configurable by the user. This part of the converter would prob-
ably have to be adjusted and customised for each future case study to account
for differences in format of the basic data. However, this is simply about parsing
text into a standard format and has no reference to RDF or any other semantic
web tools and techniques. The individual elements of information about the test
are separated by a single white space. For example, the following line shows the
information about a ping test showing, in order:

testtype, epoch time, target IP Address, % packet loss, mean round trip time

localping 1378585801 192.168.20.254 0 49.285

and then a traceroute test showing, in order:

testtype, epoch time, target IP Address, Hop1 IP Address.........

traceroute 1378585801 82.110.55.2 82.109.214.41 87.82.61.208 87.86.72.19
89.200.130.7 * 82.108.10.127 82.110.55.1 82.110.55.2

The second stage of the converter takes the partly processed file from the first
stage and sequentially uses each line as its input. This most significant part of the
converter can be used on test data from any case, provided that the basic data is
first put into the one-row-per-test format shown. The Jena classes are then used
to create RDF resources, properties and statements and to store these in a TDB
dataset and/or serialise them into an RDF/XML file. As with the TCP and UDP
flows, each test has a unique name within the model. The following is a code
snippet for creating an individual ping test with associated properties and values:

1 while (t.hasNext()) {
2 item = t.next();
3 if (itemcount==1){
4 ind = pingTest.createIndividual(orgaNameSpace + "pingtest"+item+"-"+pingtestcount);
5 ontvpnmodel.add (ind, hasTestTime, ResourceFactory.createTypedLiteral(item,
6 XSDDatatype.XSDlong));}
7 if (itemcount==2){
8 ind.addProperty(hasTargetIP, item);
9 if (itemcount==3){
10 ontvpnmodel.add (ind, hasLossPercent, ResourceFactory.createTypedLiteral(item,
11 XSDDatatype.XSDfloat));}
ONTVPNMODEL.add (ind, hasAveRTT, ResourceFactory.createTypedLiteral(item, XSDDatatype.XSDfloat));

itemcount++;

The AS number to IP address mapping information which was needed to give more information about the Internet routing aspects of the VPN was obtained from the Team Cymru service. After loading in the traceroute data to the model, a query was issued to produce a list of all public Internet intermediate hop IP addresses which had been discovered using traceroute during the test period. During the whole test period, a total of 78 different hop addresses were recorded across the routes between London headquarters and the 5 satellite branch offices. The Team Cymru return list file provided an AS number and ISP name for 54 of these addresses, spanning 8 different ASes. Just as with the ping and traceroute local data, this converter was written in two parts to make the more significant RDF creation stage more universally applicable, with a smaller parsing stage needed for each separate case. The data was parsed using the Java scanner class and the resulting triples serialised into an RDF/XML file.

7.7 Assessment of the Conversion Process

The building of the data converters described in Section 7.6 and their testing on the real data obtained from the Organisation A network and described in Section 7.5 has, in itself enabled some evaluation of the ontological approach in network monitoring and measurement. The scaling and efficiency issues of converting, storing and querying large numbers of RDF triples have been assessed, particularly using the packet-level data. It was also the foundation stage for building the overall knowledgebase, which would be used for generating user information and for scalability inference testing. Two test machines with very different capabilities were used in parallel for much of the testing to compare triple loading times, querying times and other parameters of interest. These machines will be referred to by name for convenience from here on as prawn and lobster. Summary specification of these machines are:

Prawn. A 32-bit bus-width, desktop machine. Dell Inspiron 530, running Ubuntu version 12.0.4, with 2.6 GHz Intel Core 2 Duo CPU and 4GByte memory. Exclusive use of this machine was available throughout the project.

Lobster. A more powerful machine housed in a server room at Loughborough University, available using ssh. 64-bit bus-width, with twin Intel Xenon E5645 CPUs running at 2.4 GHz. 100 GBytes of memory available. Running Ubuntu version 13.04.
It was not possible to have exclusive use of Lobster, as it was a shared resource within the Computer Science department. Therefore the performance recorded could have been reduced from that which would otherwise have been possible, by unknown and variable other demands on the system resources. However, it is reasonable to say that the performance with exclusive use would have been at least as good as that recorded and possibly better.

As previously discussed, both the packet-level and flow-level pcap conversion methods incurred some loss of the original data, although the RDF triples formed by such a process take the data a step nearer to being useful information. The size of the resulting triple stores in relation to the size of the original data files is of particular interest. Ten one-day packet captures were converted to RDF triples and added sequentially to a TDB triplestore. This was done using both packet-level and flow-level converters respectively and was attempted on both Lobster and Prawn, all with the same set of ten pcap files. Table 7.1 shows the storage-size results, which are independent of the machine used. The metrics shown on the table are:

<table>
<thead>
<tr>
<th></th>
<th>PCAP Files</th>
<th>TDB (Packets)</th>
<th>TDB (Flows)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Packets</td>
<td>21,676,704</td>
<td>21,676,704</td>
<td>21,676,704</td>
</tr>
<tr>
<td>Total Triples</td>
<td>N/A</td>
<td>124,197,444</td>
<td>1,173,163</td>
</tr>
<tr>
<td>Total Storage</td>
<td>1.95 GBytes</td>
<td>18.59 GBytes</td>
<td>165.6 MBytes</td>
</tr>
<tr>
<td>Packets/kByte</td>
<td>11.1</td>
<td>1.2</td>
<td>130.9</td>
</tr>
<tr>
<td>Triples/Packet</td>
<td>N/A</td>
<td>5.73</td>
<td>0.05</td>
</tr>
<tr>
<td>Total Flows</td>
<td>N/A</td>
<td>N/A</td>
<td>163,107</td>
</tr>
<tr>
<td>Packets/Flow</td>
<td>N/A</td>
<td>N/A</td>
<td>132.9</td>
</tr>
<tr>
<td>Triples/Flow</td>
<td>N/A</td>
<td>N/A</td>
<td>7.19</td>
</tr>
</tbody>
</table>

• the total number of packet headers in the original 10 day sample, each of which is processed during conversion in some way.

• the total number of RDF triples generated from the 10 day sample by each of the conversion methods.

• the total amount of storage needed in each case.

• the number of packets per kByte of storage. In the case of the flow conversion, this indicates the number of packets which have been processed to form the flow-based RDF triples, the actual individual packet data are lost by the conversion.
- the number of triples per packet.
- the total number of flows (flow converter only).
- the number of packets per flow.
- the number of triples per flow.

A particular objective of the passive packet-capture exercise was to test the practicality of the system running on different types of hardware when handling large datasets gathered at different times and added sequentially to the existing triplestore. This was the motivation for the rather extreme experiments carried out using 10 one-day captures of packet-level data. The actual requirements of the final monitoring system were much less demanding in two key ways:

- Only single-day slices of data were expected to be needed to monitor and manage the network.
- Packet-level data was not needed at all, as the much smaller flow-level summarised datasets contained all the higher-level information which could be expected to be of interest.

The results show that the packet-level converter produced a TDB triplestore output which required approximately 9.5 times the storage space of the original pcap files. This was despite choosing not to keep some of the packet-level flag data, which would have been in the original headers and the pcap files. This demonstrates clearly that the techniques tested here for using RDF as a data storage method would cause a significant overhead if used as a standard method for storing packet header data on a per-packet basis. The advantage is that the syntactical problems of using the information in the headers alongside other types of data should be significantly reduced by expressing it in standard RDF triples for inclusion in a user knowledgebase. There would be some scope to reduce the overhead, by such techniques as optimising the resource names and omitting ontological “hooks” in the form of class membership assertions, however both of these would weaken the utility of the information by making it harder to align it with other data.

The flow-level converter had almost the opposite effect on storage requirements. The resulting TDB triplestore needed approximately 11.5 times less disk space than the total needs of the original 10 pcap files. This was not a direct effect of using RDF, but rather a consequence of the actual summarisation into flow-level data. The results show a mean number of triples used to describe each flow of
7.19, including the small number of ontology statements used to create the classes and statements. These were actually a fixed overhead regardless of the number of packets processed or flows described. Clearly, if only the flow-level triplestore were retained, the ability to extract information about individual packets would be lost. It could be argued though, that in this case study (and others like it) this is not an important issue. Although converting packets to RDF triples describing flows is not the only method to do this type of reduction, it has been demonstrated that with some consideration of the actual information requirements at an early stage, using RDF triples as storage need not necessarily cause an increased disk space requirement. If used with a sensible summarisation technique it can make the required information more easily accessible in a standard syntax and simultaneously reduce storage needs.

The ten pcap files were processed sequentially, and the resulting triples added to the TDB triplestore one-by-one as created. The cumulative time for processing all ten files, is shown in table 7.2, broken down by computer and by conversion type. As each file in turn was processed, the existing triplestore was gradually growing in size. In the packet-level conversion case, a mean number of triples per packet of over 5 was being generated and an increase in processing time was observed as the store grew. In the case of packet-level conversion on Prawn, the time increased significantly between pcap files and eventually the machine “froze” completely during the processing of the fourth pcap file. This happened consistently during three attempts.

The processing rates in packets per second are shown in charts 7.3 and 7.4 for the packet-level and flow-level converters respectively. Each of the ten points on each line represents a single pcap file conversion and the x-axis shows the size of the TDB triplestore at the point when that particular file conversion was started.

As would be expected, Lobster consistently outperformed Prawn throughout. Significantly, during the processing of the fourth pcap file in the packet-level conversion experiment, when the triplestore size was between 6.6 GBytes and 9.2
CHAPTER 7. THE VPN CASE STUDY

Figure 7.3: Packet-Level Conversion Rate as a Function of Triplestore Size

Figure 7.4: Flow-Level Conversion Rate as a Function of Triplestore Size
GBytes, Prawn reached a critical point, froze and needed to be rebooted. Although the performance of Lobster did also fall-away from the initial approximately 7000 packets per second, it reached a fairly steady rate of around 2000 packets per second for the last five pcap files. In the flow-level conversion experiment, the triplestore size at each stage is a factor of approximately 100 smaller, and Prawn processed at a fairly steady packet per second rate throughout.

The rather erratic performance of Lobster in the later stages may be explained by other users coming on line and sharing its resources. Each converter processes every packet in the pcap files to which it is applied. In the case of the flow-level converter, there are far fewer write operations than the packet-level converter, because when a packet arrives and is identified as part of an existing flow, the only operation is to remove the triple containing the flow packet count and create a new one with the incremented integer value. This would explain the overall higher processing rate for the flow-level converter on both machines in these tests.

The pcap files were all of a similar size, so the flow-level results for Prawn are as would be expected if the only limitation were the processing power of the machine; there is no obvious relationship between the processing rate and the triplestore size. In the packet-level case though, the performance of Prawn fell rapidly and collapsed as the (much larger) triplestore built up. This effect was not near so pronounced on Lobster, which had much more memory available. Given these results, it is reasonable to assume that Prawn had insufficient RAM to deal with the part of the triplestore which must be loaded into memory to perform the conversion. In practical terms for this particular case study, the large, packet-level triplestore was not needed, but in other applications this may be a limitation.

Although the packet-level conversion TDB triplestores were not used in the user level knowledgebase development, some queries were run against them on each of the two test machines to measure query execution times on these larger stores. The times recorded cannot be compared directly between the two machines, because the triplestore sizes were quite different at 18.5 GBytes (approx. 124 million triples) on Lobster but only 6.6 GBytes (approx. 45 million triples) on Prawn, for the reasons previously discussed. Three sparql select queries were constructed, applied and then repeated for a second pass:

1. A query which was deliberately non-specific, so as to generate a very large result set. Every individual packet collected should have a statement associated with it specifying its source IP address. The query searches for the name of any resource which is the subject of the predicate orga:hasSrcIP,
and so should return one solution for every packet.

queryStr.append("select ?anypacket where{?anypacket orga:hasSrcIP ?src}");

These results were sent to /dev/null because several millions of solution were expected.

2. A query which looks for the source IP address of any packet which has a destination IP address of 192.168.10.110. In fact this query answers the question in human terms: ‘Who is sending to the terminal services server?’. The query with a sample of solutions is shown.

queryStr.append("select ?src where{?packet orga:hasDstIP "192.168.10.110" . ?packet orga:hasSrcIP ?src}");

| "192.168.20.103" | "192.168.20.103" | "192.168.60.50" |

3. A more specific query still, which looks for the destination IP address of packets with a source address of 192.168.20.105 and a destination port of 80. In practice this is asking: ‘Which web servers are the client at 192.168.20.105 trying to access?’


| "192.168.10.19" | "192.168.10.19" | "192.168.10.19" |

<table>
<thead>
<tr>
<th>Lobster (1st)</th>
<th>Lobster (2nd)</th>
<th>Prawn (1st)</th>
<th>Prawn (2nd)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>232 sec</td>
<td>185 sec</td>
<td>out of memory</td>
</tr>
<tr>
<td>Query 2</td>
<td>27.7 sec</td>
<td>19.3 sec</td>
<td>42.5 sec</td>
</tr>
<tr>
<td>Query 3</td>
<td>28.7 sec</td>
<td>18.8 sec</td>
<td>38.1 sec</td>
</tr>
</tbody>
</table>

Response times are shown in table 7.3. In the case of query 1, which would have generated a very large result-set Prawn returned an out-of-memory message, despite setting the Java heap size to the maximum. Lobster took over 3 minutes, but did return the results, although they were redirected to /dev/null. The other queries both returned their responses in what could be deemed an operationally acceptable time, although it must be remembered that there was no inference at
this stage, just the use of asserted statements to produce answers. The TDB results caching described in Section 5.5 appeared to be operating, as second attempts at the same queries were much faster.

Interestingly, although Lobster outperformed Prawn on the first attempt as would be expected, the roles were reversed on the second pass at queries 2 and 3, with Prawn being the faster. This could be explained by the fact that TDB performs caching differently on 32-bit and 64-bit machines, by the fact that the presence of other users on the system is taken into account when TDB reserves memory space for caches [28], or simply by the smaller triplestore. As well as testing the query response times, these simple queries begin to demonstrate that even at this relatively basic level, sensible questions which a network administrator may wish to ask can be easily answered, because already a small amount of domain knowledge has been embedded in the data by virtue of the statement formations chosen. This was packet-level converter output testing, but similar queries could also be addressed at the flow-level with much smaller triplestores and yet still return the same information, reinforcing the decision to use that level in the rest of the study.

In addition, it was necessary create RDF assertions of a more static nature than the test data, such as statements linking IP addresses of the interfaces of key nodes to server names, and statements linking satellite branch locations to the subnets used at those sites. These simple individual pieces of local knowledge are then given further meaning and significance by the ontological statements they link to and by the reasoners which are subsequently bound to the models. Some of these were derived from a dump from the DNS server, but because many of the client machines were identified by people’s names and other potentially sensitive information was present, this was parsed separately in the first stage and dummy names substituted, before the RDF statements were produced in the normal way using Jena in a converter program. Other information, particularly about the site, was included in the Local Linking Ontology.

7.8 Local Linking Ontology

Having developed the LIOL autonomous ontologies (AOSs), constructed the dataconverters and generated TDB triplestores ready to receive the various types of instance data, the next stage was to select the appropriate AOSs to cover the semantics needed and then to create case-specific ontological statements to:

- define any concepts and relationships particular to the Organisation-A net-
work in general and this VPN case study specifically, which were not already part of one or more of the autonomous ontologies selected.

- align those newly defined concepts and relationships with those in the selected AOs to form a combined ontology.

- link the instance data to that combined ontology, to give to further context and meaning thus forming a knowledgebase.

Many of the concepts needed to model the Organisation-A network were already defined in the AOs, but two types in particular, the idea of a site and a vpn tunnel needed to be addressed locally. Figure 7.5 shows a simplified illustration of the Local Linking Ontology which, as with the Autonomous Ontologies was serialised to an RDF/XML file for later import to the main knowledgebase. Also included in this ontology were the individuals of the classes shown, for example orga:Bristol_Branch as an rdf:type of orga:site. Of the six LIOL Autonomous Ontologies, four were deemed to be of particular relevance to the objects of the study and were added into the knowledgebase. They were the Traffic, Infrastructure, Protocols and Monitoring ontologies. In the Local Ontology, OWL and RDFS constructs are used to make appropriate links between the classes and
Listing 7.3: Example Branch Office Configuration

```java
//Create individual Site
Individual Bristol_Branch = site.createIndividual(orgaNamespace + "Bristol_Branch");
//Site Properties
Bristol_Branch.addLiteral(hasName, localOnt.createLiteral("Bristol"));
Resource brip = localOnt.createResource(orgaNamespace + "192.168.60.254");
localOnt.add(Bristol_Branch, hasLocalIP, brip);
brip.addLiteral(hasIPValue, localOnt.createLiteral("192.168.60.254"));
Resource brrip = localOnt.createResource(orgaNamespace + "123.123.123.123");
localOnt.add(Bristol_Branch, hasPublicIP, brrip);
brrip.addLiteral(hasIPValue, localOnt.createLiteral("123.123.123.123"));
Resource brLAN = localOnt.createResource(orgaNamespace + "brLAN");
brLAN.addLiteral(hasName, localOnt.createLiteral("Bristol_LAN"));
localOnt.add(Bristol_Branch, hasSiteLAN, brLAN);
Resource brRouter = localOnt.createResource(orgaNamespace + "brRouter");
brRouter.addLiteral(hasName, localOnt.createLiteral("Bristol_Router"));
localOnt.add(Bristol_Branch, hasRouter, brRouter);
```

properties in the `orga` namespace and those in the `lboro` namespace, aligning the ontologies and allowing the higher-level concepts to be used locally. The Local Ontology introduces two new classes, `orga:site`, as the class of all physical sites, such as satellite branch offices, and `orga:vpnTunnel`, the class of all individual tunnels which together form the links between the sites. Each site is characterised by a `local IP address` (actually the internal interface address of its router), and a `public IP address` (the external interface address of its router). As with the other ontologies, the IP addresses are `resources`, not simply text or literal values. Where appropriate, `datatype properties` are used to allocate to them an `IP value` and an associated `IP subnet name` (network address). Each site is associated with its router and LAN, which are inferred as members of the appropriate classes by the the `rdfs:range` of the properties `orga:hasRouter` and `orga:hasSiteLAN`. The code to create the statements defining the Bristol site is shown in Listing 7.3 as an example. The public IP address has been obfuscated.

One of the important roles of this ontology is linking the concepts held in the two initially separate namespaces `lboro` and `orga`. In the case of the classes representing the concepts of IP addresses, TCP flows, UDP flows and ICMP communications, the standard OWL property `owl:equivalentClass` is used to assert that any resource which is an individual of one, or any class which is a sub-class of one, also has that same relationship with the equivalent class in the other namespace. By doing this, we can know by inference, for example that any individual TCP flow from the namespace in this case is also a member of the class `lboro:IPCommunication`, because that is declared as a super-class of `lboro:TCPFlow` in the Traffic AO. In the case of properties, those used to directly allocate destination and source IP addresses in the various flows in this case are also directly linked to the corresponding ontological concepts in the traffic AO, this time using `owl:equivalentProperty`,
to similarly extend the ontological constructs given to those properties at the higher level. In addition to using them to allow lower-level instance data to inherit higher-level semantics as done here, these OWL constructs offer a very simple way to align data from different sources in cases where the concepts are already understood to be the same. In the networking domain, both the idea of an IP address and a TCP or UDP flow, for example are well defined, so these constructs offer one way of reducing the syntactical and semantic problems of interest in this research.

In the Infrastructure AO as described in Section 6.10, the idea of a symmetric property and a transitive property are combined to simulate the situation where a network device is plugged into a LAN. At the network layer, the connection is both symmetric (2-way) and transitive (device connects to other devices connected to the same LAN). This concept is utilised in this case for the individual clients and servers in the base data, to model their connection to the various site LANs. This is done by asserting that the local linking properties, for example orga:hasGlLink, are declared as sub-properties of lboro:dataLink. Any two resources used as subject and object respectively of orga:hasGlLink, will also be connected by the properties further up the sub-property chain. In this way, the symmetric and transitive properties between the two can be either observed or ignored depending on which property in the hierarchy is used as the basis for a query. In the case of the connections between sites by VPN tunneling, these are symmetric but not transitive; clients at Bristol and Glasgow for example can each connect to any machine in the London subnet, but there is no route for them to “talk” to each other. This is a case where care is needed to avoid allowing the sub-property/super-property inferences to generate unwanted outcomes, which would be the case if the wrong lower-level property were used for this relationship. Instead a new property, orga:interSiteTunnel was used as a property with the routers at each end of the tunnel as the subject and object. Where possible, rdfs:domain and rdfs:range were also set for the new properties, although again care then needed to be taken with their use to prevent any inappropriate inferences being drawn by reasoners later.

The domain and range declarations were a useful way of inferring that Resources not originally declared as Individuals of any particular class could be inferred to be members of the class simply by being subjects or objects as appropriate of these statements. Some examples of the code to create the ontological statements are shown in Listing 7.4 The full Local Linking Ontology RDF/XML serialisation is shown in Appendix D.
Listing 7.4: Key Statements from the Local Linking Ontology

```java
// Equivalent Classes. These statements equate
IPAddrl.addEquivalentClass(IPAddro);
hasSrcIP1.addEquivalentProperty(hasSrcIPo);
hasIP.addEquivalentProperty(hasIPAddr);
dataLink.addSubProperty(hasBrLink);

// Equivalent and sub Properties
hasIP.addEquivalentProperty(hasIPAddr);
hasSrcIPl.addEquivalentProperty(hasSrcIPo);
dataLink.addSubProperty(hasBrLink);

// Add Range/Domain
hasRouter.addDomain(site);
hasRouter.addRange(router);
hasIPValue.addDomain(IPAddro);
interSiteTunnel.addDomain(router);
hasLocalIP.addRange(IPAddro);
```

7.9 Application and Testing

The knowledgebase structure is illustrated in Figure 7.2, which shows all the necessary Java classes which were needed to convert the raw base data and import the various ontological statements described in the preceding sections. The next stage was to load up the base data for a chosen period for the application of reasoning and querying operations. A single day’s passive packet capture file was chosen, and the active probe testing data for the same period extracted from the file. A day was chosen at random of 12th August 2011 (Incidentally, in this context there was no connection with grouse shooting!). The rationale for initially choosing a one day slice of data was that this would be the likely largest timeframe needed in one batch for operational reasons to monitor the performance of the network.

The data were loaded-in, along with the ontologies, to the knowledgebase. A Java class, Querygen was then written to:

- import the knowledgebase into a Jena Ontology Model in memory.
- select various built-in reasoner options using the Jena ModelFactory options.
- attach a Rule-Reasoner to allow Jena Rules to be applied from a text file.
- create a memory-resident Inference Model to hold the Inferred and asserted statements together.
- create a SPARQL query which could be easily modified to test the various aspects of the model.

In this way, queries could easily be executed against the knowledgebase with:

- no inference.
- RDFS inference from the built-in reasoner.
- OWL Lite inference from the built-in reasoner.
inference from custom rules only, created locally in a text file.

- inference from custom rules combined with either RDFS or OWL Lite.

This was basically implementing the second part of the General Methodology from Section 5.7, the specific implementation for this study is illustrated in figure 7.6 and forms a testbed for evaluation of the ontological approach in this study. The information which may be required from the knowledgebase may be divided into two categories:

**Management information** This concerns the network infrastructure, connections, hosts and addresses. This is relatively static and in this case comes mainly from the DNS data and the Local Ontology. It informs on how things are connected, which hosts are at which physical sites, which are the servers and (from the client names only in this case) who are the users.

**Performance information** This is more dynamic and informs about recent events, actions and any pathologies within the network. In this case this is mainly derived from the passive and active measurements which are performed from the research server.

The approach to testing was to pose questions for which a network administrator may need the answer, design SPARQL queries to mimic those questions, apply
those queries and check the result-sets. Some typical examples of queries are shown, with a plain language question, the query submitted to the knowledgebase and the result set returned. The results are illustrated basically as formatted by the standard Jena SPARQL results formatter with no embellishments, in order to demonstrate their authenticity, however it would be an easy task to output these in a more user readable form if required. The prefix declarations for the queries are omitted from individual query listings as they are the same in all cases, and are listed once only.

Prefix declarations, used in all queries

```
PREFIX lboro: <http://nets.lboro.ac.uk/liol#>
PREFIX orga: <http://nets.lboro.ac.uk/orga#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Question 1: What are the names, ip addresses and Locations of all the servers?

Test Machine: Prawn

Query:

```
select ?ServerName ?IPAddress ?SiteName
where{(?a orga:hasIPAddr ?b .
?a orga:hasServerName ?ServerName .
?b orga:hasIPValue ?IPAddress .
?b orga:hasIPSubnet ?f .
?g orga:hasLocalIP ?h .
?h orga:hasIPSubnet ?f .
?g orga:hasName ?SiteName)}
```

Query Results:

```
<table>
<thead>
<tr>
<th>ServerName</th>
<th>IPAddress</th>
<th>SiteName</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Domain_Controller&quot;</td>
<td>&quot;192.168.10.130&quot;</td>
<td>&quot;London&quot;</td>
</tr>
<tr>
<td>&quot;Glasgow_Voicemail&quot;</td>
<td>&quot;192.168.10.19&quot;</td>
<td>&quot;London&quot;</td>
</tr>
<tr>
<td>&quot;Terminal_Services&quot;</td>
<td>&quot;192.168.10.253&quot;</td>
<td>&quot;London&quot;</td>
</tr>
<tr>
<td>&quot;Webmail_Server&quot;</td>
<td>&quot;192.168.10.19&quot;</td>
<td>&quot;London&quot;</td>
</tr>
<tr>
<td>&quot;NAS_Drive&quot;</td>
<td>&quot;192.168.10.100&quot;</td>
<td>&quot;London&quot;</td>
</tr>
<tr>
<td>&quot;London_Voicemail&quot;</td>
<td>&quot;192.168.10.17&quot;</td>
<td>&quot;London&quot;</td>
</tr>
</tbody>
</table>
```

Query Execution Time = 4 seconds

This query searches for all resources which have a server name and an IP address. The value of the IP address and its associated subnet (network address) are accessed, the subnet value is then used to identify the physical site. ServerName, IPAddress and SiteName are then returned. This query uses management information from the DNS download and the Local Ontology, but no short-term performance information.

Question 2: Have any of the ping tests to site LANs had packet losses?

If so, where to and how bad?

Test Machine: Prawn

Query:

```
?a orga:hasName ?Site .
?a orga:hasLocalIP ?b .
?c rdf:type orga:pingTest .
?c orga:hasTargetIP ?b .
?c orga:hasLossPercent ?Loss .
?c orga:hasTestTime ?TestTime .
```

This query searches for all resources which have a server name and an IP address. The value of the IP address and its associated subnet (network address) are accessed, the subnet value is then used to identify the physical site. ServerName, IPAddress and SiteName are then returned. This query uses management information from the DNS download and the Local Ontology, but no short-term performance information.
This query uses performance information from the active probe measurements. It searches for any resource which is a member of the class orga:site, and has a local IP address (all sites are allocated a distinguishing IP address in the Local Ontology, actually the internal interface address of the Draytek router). Then a resource which is a member of the class pingTest is sought which has its target IP address the same as the identified site local IP address. The percentage of packets lost and the time of the test are returned along with the site name. To qualify for membership of the result set, the filter condition of having a loss percent greater than zero must be satisfied. The test time is returned in epoch time, but could easily be converted to a more readable format for presentation to a user. The results show that there was a problem with the connection from London to Manchester at around 10:30 am GMT, (11:30 am local time), with 100% packet loss. Using domain knowledge about how the VPN works lead to an interesting follow up to this query; to execute another which checks the ping test results between the London and Manchester external interfaces at the same time:

**Follow-up Question 3: How was the packet loss on the external interface at around the same time of the internal ping test which showed a problem?**

**Test Machine: Prawn**

**Query:**

```sparql
SELECT ?Site ?Loss ?TestTime WHERE {?
  a rdf:type orga:site .
  ?a orga:hasName ?Site .
  ?a orga:hasPublicIP ?b .
  ?c rdf:type orga:pingTest .
  ?c orga:hasTargetIP ?b .
  ?c orga:hasLossPercent ?Loss .
  ?c orga:hasTestTime ?TestTime .
  FILTER(?TestTime > 1313144701) .
  FILTER(?TestTime < 1313145301)}
```

**Query Results**

<table>
<thead>
<tr>
<th>Site</th>
<th>Loss</th>
<th>TestTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Glasgow&quot;</td>
<td>&quot;0.0&quot;</td>
<td>1313145001</td>
</tr>
<tr>
<td>&quot;York&quot;</td>
<td>&quot;0.0&quot;</td>
<td>1313145001</td>
</tr>
<tr>
<td>&quot;Bristol&quot;</td>
<td>&quot;0.0&quot;</td>
<td>1313145001</td>
</tr>
<tr>
<td>&quot;Manchester&quot;</td>
<td>&quot;100.0&quot;</td>
<td>1313145001</td>
</tr>
<tr>
<td>&quot;Dublin&quot;</td>
<td>&quot;0.0&quot;</td>
<td>1313145001</td>
</tr>
</tbody>
</table>

Query Execution Time = 1 seconds

This query adopts a similar approach to the former, but instead looks at the ping tests to the external IP addresses of the sites. There is no filter on packet loss, so all measured values are returned but on this occasion TestTime is filtered to select a five minute band around time of the observed problem at Manchester. These results elaborate on what was known before and show that not only was the
connection over the VPN itself not fully connected, but also the external Internet connection over which the VPN was built was not responding at Manchester. If this test had instead returned zero packet loss for the test to the Manchester external interface, a different fault would have been diagnosed; Internet connection working but tunneling inoperative.

Question 4: Who has been accessing the Old Web Server in the last hour (Assumes the question asked at 3:00pm on the 12th August 2011)?

Test Machine: Prawn

Query:

```
select ?ClientName where
  (?
  a rdf:type lboro:IPCommunication .
  ?a orga:hasDstIP ?b .
  ?a orga:hasTestTime ?f .
  ?c orga:hasServerName "Old_Web_Server" .
  ?c orga:hasIPAddr ?b .
  ?a orga:hasSrcIP ?d .
  ?e orga:hasIPAddr ?d .
  ?e orga:hasClientName ?ClientName .
  FILTER(?f > 1313154000) .
  FILTER(?f < 1313157600))
```

Query Results

<table>
<thead>
<tr>
<th>ClientName</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;helpdesk3&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk3&quot;</td>
</tr>
<tr>
<td>&quot;haricot&quot;</td>
</tr>
<tr>
<td>&quot;viceroy&quot;</td>
</tr>
<tr>
<td>&quot;viceroy&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk3&quot;</td>
</tr>
</tbody>
</table>

Query Execution Time = 6 seconds

This query uses performance information from the passive packet capture exercise, and also relies on inference provided by the Autonomous Ontology links. It effectively uses the sub-class hierarchy which the knowledgebase derives from the Traffic AO, which makes all types of flow a member of the class lboro:IPCommunication. The destination IP addresses for all IPCommunication packet captures are then compared against the IP address associated with the resource whose ServerName is Old_Web_Server. The source IP address in the same IPCommunication is then matched against any IP addresses of resources, and the property orga:hasClientName returns the name of that resource to the result-set, as long as the filter requirements are also met. A one hour time-slice is selected.

Question 5: Tell me the names of all the clients and their LANs who have initiated any communications in the last few minutes?

Test Machine: Prawn

OUT OF MEMORY message after 6min 30sec.

Test Machine: Lobster

select ?Client ?LAN where

```
(?a rdf:type orga:clientHW .
  ?b rdf:type lboro:LAN .
  ?b orga:hasName ?LAN .
  ?a orga:hasName ?Client .
  ?a lboro:dataLink ?b .
  ?a orga:hasIPAddr ?j .
  ?e rdf:type lboro:IPCommunication .
  ?e orga:hasTestTime ?f .
  FILTER(?f > 1313157500) .
```

CHAPTER 7. THE VPN CASE STUDY

FILTER(?t < 1313157600) .
?e orga:hasSrcIP ?j .

Query Results (Lobster)

<table>
<thead>
<tr>
<th>Client</th>
<th>LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;jollyroger&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk1&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk1&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk1&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk1&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;helpdesk1&quot;</td>
<td>&quot;Glasgow_LAN&quot;</td>
</tr>
<tr>
<td>&quot;vicery&quot;</td>
<td>&quot;Manchester_LAN&quot;</td>
</tr>
</tbody>
</table>

Query Execution Time = 307 seconds

This query involves properties derived from most of the larger data-sets in the knowledgebase and multiple inferences between the namespaces and the various ontologies. All IPCommunications are selected using the Traffic AO sub-class hierarchy. In addition, the lboro:DataLink property is used, which itself is a symmetric property and has a super-property which is transitive. It therefore placed a high demand on system resources, and Prawn was unable to complete and returned an “out of memory” message. The query was then run on Lobster, which returned the result-set in just over 5 minutes. Included with the listing is a snapshot of system resources being used by the query process, obtained using the linux top command. It shows 100% CPU usage, 27.9 GBytes of virtual memory reserved for the process and resident (physical) memory in use of 9.5 GBytes.

Question 6: Which network providers were carrying my data to Dublin at 11:30 this morning?

Test Machine: Prawn

Query:

```sparql
select ?HopNumber ?AS ?ISP where{
?a rdf:type orga:traceTest .
?a orga:hasTestTime ?t .
?a orga:hasTargetIP ?b .
?a orga:hasPublicIP ?b .
?m orga:hasName ?n .
?b orga:hasIPValue ?z .
?a orga:hasHop ?c .
?c orga:hasHopNum ?HopNumber .
?c orga:hasHopIP ?e .
?g rdf:type orga:autonSystem .
?g orga:hasASNum ?h .
?h orga:hasASValue ?AS .
?g orga:ownedBy ?j .
?j orga:hasCoName ?ISP .
?g orga:hasHopIP ?e .
FILTER(?t = 1313148601) .
FILTER regex(?n,'^Dublin','i'))
```

Query Results

<table>
<thead>
<tr>
<th>HopNumber</th>
<th>AS</th>
<th>ISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;35228&quot;</td>
<td>&quot;BEUNLIMITED_Telefonica_UK_Limited&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;3549&quot;</td>
<td>&quot;GBLX_Global_Crossing_Ltd.&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;1299&quot;</td>
<td>&quot;TELIANET_TeliaSonera_International_Carrier&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;1299&quot;</td>
<td>&quot;TELIANET_TeliaSonera_International_Carrier&quot;</td>
</tr>
<tr>
<td>6</td>
<td>&quot;1299&quot;</td>
<td>&quot;TELIANET_TeliaSonera_International_Carrier&quot;</td>
</tr>
</tbody>
</table>

Query Execution Time = 0 seconds
This query searches for a resource which is a traceroute test, then derives the hop level IP addresses and then associates these with the IP address to AS mapping properties, which in turn were created using the public base level text file based data from the Team Cymru service. The datatype properties for the literal names and IP values are then used to present a human-readable answer to the original question.

### 7.10 Rules

The example user questions in Section 7.9 and their corresponding SPARQL queries demonstrate some of the ways in which the structure of the model and the inferences which are included can be used to combine data from many disparate sources in an RDF based knowledgebase to good effect. Question 5 and follow-up question 6 are examples where the queries deliver useful information, but still require a degree of domain-specific and individual network-specific knowledge to interpret the results returned. In situations such as this, it is possible to enhance the reasoning provided by RDFS and OWL by adding custom Rules into the reasoning mix. In this situation, it was desirable to build-in a way for senior engineers at TRML to incorporate some expertise into the knowledgebase to alert junior staff, not only that there was a problem, but also what steps to take next. The ping-loss problem provided a vehicle for that development. The method which was used can be described thus:

1. A *Jena GenericRuleReasoner* object was created in the *querygen* class, associated with a *rule* created separately in a text file://[0.5ex]

   ```java
   List rule = Rule.rulesFromURL("file:/home/ian/rule2.rul");
   Reasoner reasoner = new GenericRuleReasoner(rule);
   reasoner = reasoner.bindSchema(ontquerymodel);
   InfModel infmodel = ModelFactory.createInfModel(reasoner, ontquerymodel);
   ```

   The reasoner was bound to the ontology model, and a new memory resident inference model was created, which gave a view onto the original asserted statements (triples), the inferred statements created by the ontology model’s built in reasoner using OWL Lite reasoning, and now any further statements created when the associated rules are fired.

2. The rules were written into the text file using the *Jena Rules* syntax. Two rules were required, *extconnectionRule* fires if there exists an individual of the *org:pingTest* class, which has a packet loss percentage more than zero. The affected site was then identified by matching the target IP address of the ping test with the public IP address of one of the satellite branch offices. Each test which fires the rule generates four triples. The subject of each
is the ping test itself, the objects are all string literals and the properties linking them are orga:hasWarning, orga:warnAbout and two suggestion properties. The intconnectionRule fires on similar lines, but also looks checks for packet losses of less than 10% between external router interfaces and losses of more than 80% on between the LANs. A different set of literals are given.

```prefix lboro: http://nets.lboro.ac.uk/liol#
@prefix orga: http://nets.lboro.ac.uk/orga#
@include <RDFS>.

[extconnectionRule: (?a rdf:type orga:site) (?a orga:hasName ?Site) (?a orga:hasPublicIP ?b) (?c rdf:type orga:pingTest) (?c orga:hasTargetIP ?b) (?c orga:hasLossPercent ?Loss) (?c orga:hasTestTime ?TestTime) greaterThan(?Loss,0) greaterThan(?TestTime,1313144701) lessThan(?TestTime,1313145301) -> (?c orga:hasWarning "INTERNET_DOWN") (?c orga:hasWarnAbout ?Site) (?c orga:suggestion1 "REBOOT_ROUTER") (?c orga:suggestion2 "CALL_ISP")]

[intconnectionRule: (?a rdf:type orga:site) (?a orga:hasName ?Site) (?a orga:hasPublicIP ?b) (?c rdf:type orga:pingTest) (?c orga:hasTargetIP ?b) (?c orga:hasLossPercent ?Loss) (?c orga:hasTestTime ?TestTime) lessThan(?Loss,10) (?a orga:hasLocalIP ?d) (?e rdf:type orga:pingTest) (?e orga:hasTargetIP ?d) (?e orga:hasLossPercent ?locloss) (?e orga:hasTestTime ?TestTime) greaterThan(?locloss,80) -> (?c orga:hasWarning "INTERNET_OK,TUNNEL_DOWN") (?c orga:hasWarnAbout ?Site) (?c orga:suggestion1 "RESET_TUNNEL") (?c orga:suggestion2 "CALL_MARTYN") (?c orga:failsite ?Site)]

3. Having created the rules, a very simple query could then be used to check to see if they had been fired and to obtain the warnings and suggestions they contained:

```
```

<table>
<thead>
<tr>
<th>Site</th>
<th>ALERT</th>
<th>FIRST</th>
<th>SECOND</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Manchester&quot;</td>
<td>&quot;INTERNET_DOWN&quot;</td>
<td>&quot;REBOOT_ROUTER&quot;</td>
<td>&quot;CALL_ISP&quot;</td>
</tr>
<tr>
<td>&quot;York&quot;</td>
<td>&quot;NET_OK,TUNNEL_DOWN&quot;</td>
<td>&quot;RESET_TUNNEL&quot;</td>
<td>&quot;CALL_MARTYN&quot;</td>
</tr>
</tbody>
</table>

Query Execution Time = 1 seconds

The query returns two alerts, each with different suggestions for the next course of action to rectify the respective alerted problems. The Manchester alert was a genuine one from the field data, the York incident was artifi-
cially created by adjusting the real base data (the only such modification done throughout) to simulate the conditions which the rule was designed to find. This approach could be used by a network administrator to develop a comprehensive library of automatic diagnostics and is a practical way to embed network domain expertise.

7.11 Case Study Summary and Review

The VPN case study was a complete practical project in its own right, building on and utilising the more abstract ideas developed for the Loughborough Internetworking Ontology Library (LIOL) described in Chapter 6.

A data gathering system was developed within a live commercial network, bringing in a wide range of data-sets in different formats, including passive and active measurements made in-house and external data related to the public Internet. Java classes were produced to convert these disparate data-sets into standard RDF triples, all held in a single knowledgebase. In the packet-level conversion exercise, a disk-backed triplestore of over 124 million triples, occupying over 18.5 GBytes was created, and the practicalities of converting volumes of data at this scale, including conversion times on two very different machines were tested. In the case of the standard desk computer with 4 GBytes of memory, this was tested to complete failure by memory overload.

A new local ontology was created to define the important concepts in the local network and to be the “glue” holding together the instance data and higher-level reference ontologies. This ontology was possibly the most important part of the project, because many of the ontological constructs which made the knowledgebase more than just a simple database were created there. The ontological statements were used as one way of including domain expertise alongside the data in the same store, a core concept of the semantic web tools which were used.

The knowledgebase was then tested on a single-day slice of data using a number of queries, a representative sample of which are documented here. The queries drew on triples originally derived from all the different data sources and so demonstrated that such information, when properly converted, could simply be put together as a set of unordered RDF triples in a purely RDF-based model. The model could then be reasoned over to give answers to some real-life questions for which a network administrator may on occasion need rapid answers. The queries demonstrated that sensible answers could be obtained but in some particularly
involved cases they proved too much for the system resources of a basic desk-top computer and needed to be processed on lobster, a more powerful machine, where they were always successfully processed.

Another part of the semantic web toolkit, the rules system was then tested as a way to include very specific diagnostic information and solution suggestions in the knowledgebase. This was implemented for some real situations which were known to occur from time to time and a typical example is documented.

In terms of the interpretation problems laid out in Section 5.8, the ontological approach used here has been demonstrated to be one practical way of addressing them. When analysing how well this has been done, it becomes clearer that the syntactical, semantic and expertise problems are closely related and to some extent they overlap. The syntactical problem was addressed by bringing together very differently formatted base data and representing it in RDF triples using the clear and well understood standards laid out by the W3C. This worked very well. The semantic problem and was also partly addressed by the design of the data structure into which the resulting triples were placed and by the ontological statements made about it, both implemented purely in RDF. The expertise problem was partly addressed again by the ontology statements which were used by the standard reasoners to enrich the information from the instance data. A more direct approach using rules was also implemented to good effect, demonstrating the possibility for a senior administrator to make some of his/her expertise available in absentia through a knowledgebase.
Chapter 8

The AS Topology Case Study

8.1 Chapter Introduction

This chapter describes in detail the practical research work carried out in the AS Level Topology Case Study. Section 8.2 introduces the problem in the context of other research initiatives and explains the motivation for choosing this particular case. Section 8.3 goes on to describe the methodology employed and the potentially conflicting base datasets which were used. The conversion process used to create standard RDF triples from these datasets is further explained and analysed in Section 8.4. Section 8.5 gives a detailed description of the ontology design for the study, emphasising why the design route taken was not necessarily the most obvious one but was judged to be more useful in this context. The final data conversion process and viability testing of the implementation is described in some detail in Section 8.6. Section 8.7 summarises and reviews this case study.

8.2 Background and Motivation

The motivation for monitoring and measuring the Internet and some of the barriers to doing so are discussed in general terms in Chapter 2. Some of the problems encountered arise from the fact that although some aspects are regulated, for example IP addressing and domain names, there is no overall central control. This is particularly true of the topology of the Internet infrastructure. The autonomous systems (ASes) are free to peer (interconnect) with each other for practical and commercial reasons of their own and to change these arrangements as they see fit. In Chapter 1, official UK government statistics are presented to demonstrate the importance of the Internet to society and the reliance on it for commercial and social reasons. The topology at the AS level is one important aspect which needs to be understood in order to predict any problems which may be on the horizon.
CHAPTER 8. THE AS TOPOLOGY CASE STUDY

and to enable governments and other stakeholders to influence the direction in which the Internet is headed.

Because no one organisation can know with authority the exact structure of the Internet connections at any one time and because that structure is potentially important to us all, topology discovery has been an active research area for many years. CAIDA (Section 2.6) is one group which is active in this area having produced many research papers on and around the subject of topology discovery, particularly at the AS level. In a CAIDA technical report of 2012 [25], Huffaker et al follow up on their earlier work and give a description of the data sources which are available to enable graphs of the Internet topology to be derived. They explain that these graphs can be produced at the IP-level, at which every router interface is a node or at the router-level, at which every router itself is a node. One of the difficulties of operating at these levels is that some ASes have large numbers of routers within their own networks (Huffaker et al [25] report that one anonymous ISP had 2420 routers in their network, for example), and not all would be prepared or required to release such internal potentially sensitive information. The alternative is the AS-level abstraction, in which each AS is a node and the peering arrangements between them are the edges. At the AS-level three fundamentally different types of data can be used to derive the Internet graph. They are:

- traceroute data, derived from sending active probes between nodes and recording the responses of the intervening nodes. This operates at the router-level or IP-level, although AS-level information can be mapped to it. (see Section 2.5).

- BGP path data, derived from the Routing Information Base (RIB) of BGP peers and showing paths already resolved to the AS level.[58].

- whois administrative data, gathered from one of the regional Internet Registries (RIRs). The European RIPE registry is favoured by researchers as the most reliable but because it relies on voluntary returns from ISPs it can still be lagging behind the “ground-truth” situation in time and incomplete.

In 2011 CAIDA hosted a workshop to discuss BGP and traceroute topology data. The summary report [9] concludes with a number of open questions, one of which is “Can we integrate traceroute data into inferences of AS routing relationships to further improve their accuracy?” The AS Topology Case Study was an attempt to develop an ontological approach to bringing these two types of data sources together in a logical framework, using semantic web technology and provide a new way for combining them which may lead to a more accurate model of the
real Internet topology than is possible by either alone. It may be considered as a proof-of-concept exercise.

8.3 Methodology and Data Sources

The General Methodology described in Section 5.7 was used as the basis for the AS Topology Case Study, adapted to meet the specific requirements of this part of the research. The case-specific methodology is illustrated in figure 8.1. In this case, only two different types of data were utilised; traceroute responses and dumps from a BGP peering router. The aim was to convert these so that they were in the same format within an ontology model so that ontological constructs could be used to combine them in different ways.

The traceroute data were obtained from an ongoing project conducted by other researchers at Loughborough University. That project aims to send traceroute tests to at least one subnet within every AS in the Internet, and refine these to obtain the maximum number of responses. The file supplied had traces to over 300,000 remote IP address targets, with varying degrees of success in reaching the target and illiciting responses from hops along the path. The aim with this data-set was to convert it from a list of sequential IP addresses in a path, to a corresponding list of AS numbers in the same path, so that the AS level graph edges could be represented and combined with the BGP path data. The Team Cymru service\(^1\), discussed in Section 2.6 was again used as a way to obtain the mappings. The steps in the data-conversion process are illustrated in figure refasmethod, which shows the Java classes used at each stage. The steps were to:

1. reduce the original file, which contained over 6 million IP address references to a list of unique IP addresses for processing by Team Cymru. There was much duplication in the original data, as would be expected because the origin of each trace was the same machine and therefore the first few hops were the same in all traces. Each of the original IP addresses was read into the program in turn and compared against each existing element in a string array. If that IP address was not already in the array, it was entered as the next element in the array and so on. In this way, an array of just over 450,000 unique IP addresses was built up and exported as a one-address-per-line file.

2. use the Team Cymru look-up service to process the unique addresses and where possible obtain a mapping to an AS number, then return a file with one-to-one mappings.

\(^1\)https://www.team-cymru.org/ Last viewed September 2013
Figure 8.1: The AS Topology Study Methodology (See Appendix F for Key to Symbols)
3. run through each trace and replace the IP address with the corresponding AS number. This involved creating a pair of string arrays, in which the IP addresses and AS mappings respectively were stored in a pair-wise fashion. Then each of the 6 million IP address references in the traces were read in *in the correct order* and each in turn compared against the IP address array. When a match was found, the element from the corresponding location in the AS array was output, again *in the correct order*. In this way each IP address in the original file was replaced with an AS number. Due to the large number of comparisons needed, this program took just over 37 hours to run.

### 8.4 Data Conversion Analysis

The end results of these steps was a set of AS paths, one for each trace in the original traceroute file. There were some exceptional situations:

- When the hop in the traceroute had failed to return an IP address, the customary asterix was there in its place. In this situation, an asterix was also placed in the corresponding position in the newly created AS path.

- When the trace *did* have an IP address in a particular hop position but the look-up failed to find a match in the array because there was no match in the Team Cymru return, a letter “n” was placed as a marker in that position in the new AS path. This situation would arise when private IP addresses were being used within as AS, which would have no unique value and therefore no frame of reference outside that AS.

- When a chain of IP addresses in a trace were from the same AS, consecutive hops in the new AS path obviously had the same value. These were retained as hops in the AS path at this stage.

Table 8.1 shows the numbers of IP address references, traces, AS number references and AS paths where appropriate at each stage. Note that these are references; at this point there were still a large number of asterix markers and “n” markers which are included in the table numbers.

Loughborough University, in common with other academic institutions in the UK, is connected to the *Joint Academic NETwork (JANET)*, and it was from one of the BGP peering routers in JANET that the BGP dump file was obtained by Loughborough researchers. The BGP dump file was very simple to process; each line from the file containing an AS path was extracted in one step, and an AS path file in the
Table 8.1: Traceroute and BGP dump to AS path Conversion

<table>
<thead>
<tr>
<th></th>
<th>Traces</th>
<th>IP addresses</th>
<th>AS Numbers</th>
<th>AS Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace File</td>
<td>311,565</td>
<td>6,236,748</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Cymru request</td>
<td></td>
<td>459,783</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Cymru match</td>
<td>N/A</td>
<td>450,348</td>
<td>450,348</td>
<td>N/A</td>
</tr>
<tr>
<td>BGP Dump</td>
<td>N/A</td>
<td>N/A</td>
<td>1,974,591</td>
<td>453,809</td>
</tr>
<tr>
<td>Trace derived Path File</td>
<td>N/A</td>
<td>N/A</td>
<td>6,236,748</td>
<td>311,565</td>
</tr>
<tr>
<td>BGP derived Path File</td>
<td>N/A</td>
<td>N/A</td>
<td>1,974,591</td>
<td>453,809</td>
</tr>
</tbody>
</table>

The same format as that derived from the traceroute was created. The final stage of data integration into RDF for inclusion in the knowledgebase is more appropriate to the ontology, and so is discussed in Section 8.6.

8.5 The AS-Level Ontology

There are several different ways in which the ontological constructs of the semantic web could be used to model the AS level Internet graph. Just as in the VPN case study in Chapter 7, it was important to consider the end use for the model at the start of the design process. Possibly the most intuitive approach would have been to make all ASes members of the class extension of the class `lboro:autonSystem`, as created in the `infrastructure AO` described in Section 6.6 and then use the object property `lboro:ASLinkTo` also created there with ASes as its subject and object respectively for each link discovered. This puts the emphasis on the ASes as resources and makes them available as individuals in a class hierarchy, leaving the links as properties. In this study though, the existence of the autonomous systems themselves is not in doubt because the AS topology problem described by CAIDA [25] concerns the existence (or not) of the links between them, which are mapped differently depending on which discovery method is used. Therefore for this exercise it was decided not only to make the ASes individuals in their own class extension, but also to create new classes for the links themselves, thus extending the infrastructure AO to suit the local needs of the study. If at a future time the concepts used in this study needed to be aligned with the higher level ontologies, all these new link classes would be sub-classes of the more generic `lboro:link` class.

The existence of a link between any two ASes at any point in time cannot be
known for certain by remote testing because any of the discovery methods can give a “false positive” or fail to discover a link which does actually exist. The premise adopted in this study is that it is possible to be more certain about the existence of a link if it is discovered more often, either by different methods or by the same method on more than one occasion. For example, the problems with traceroute described in Section 2.5 can lead to false link discovery because routing may have changed between the launch of one individual packet forming the test and the next one, or by different forwarding decisions in the two packet flights. If however, two or more traceroute tests to the same target both discover the same link, it is more likely that such a link exists. Likewise, if a link is discovered by both BGP and by traceroute, more certainty can again be attached to that link.

A sliding scale of certainty exists, with the least conservative estimate requiring only a single discovery by any method for inclusion of a link in the graph. At the most conservative end of the scale, the link would only be acknowledged if and only if it is found on every test available by any method. This is the basic set theory idea of unions and intersections. In between these extremes, there can be compromise solutions, the choice depending on the view of the domain expert doing the analysis. In the ontology, this idea is modelled using OWL and RDFS constructs and tested using the real datasets whose partial conversion has already been described in Section 8.3.

The AS-level ontology is illustrated in Figure 8.2 and a full RDF/XML listing is in Appendix E. It consists mainly of a hierarchy of OWL classes and intersections, also represented as classes in the diagram. The four classes shown with shaded-background symbols across the centre row in the figure are those classes whose class extensions are populated with individual resources, one per link discovered in the field data. Four are shown, but the concept is for each dataset to have its own class with an individual ID within the namespace, such as those illustrated. In this sense, a dataset is considered as a batch of traces or a BGP dump file containing information about many potential links, such as the two datasets used in this study. These “data-classes” then form the foundation for the ontology, which aims to make different views of the combined data available depending on which class membership is specified as a requirement for links returned in the result-set of a query. The naming convention for the links themselves as individual resources follows a consistent pattern regardless of the dataset from which they are derived. In each case the name includes the AS numbers which the link connects from and to in that order. The super-classes and extension classes (with the prefix omitted) are:

**TRASLink101 and TRASLink102** These are the classes whose class exten-
Figure 8.2: The AS Level Ontology (See Appendix G for key to symbols)

sions are intended to include the link information derived from batches of traceroute results. This is by naming convention only, in fact link resources from any source could be declared as members of these classes.

**BGPASLink1 and BGPASLink2** These are the classes whose class extensions are intended to include the link information derived from BGP dumps. This is by naming convention only, in fact link resources from any source could be declared as members of these classes.

**TRASInter** This is an *intersection*. All link resources (and only those link resources) who are members of *both* of the listed classes in the intersection are inferred as members of this class. In other words links have to have been discovered by a trace in *both* traceroute batches to be included.

**BGPASInter** This is an *intersection*. All link resources (and only those link resources) who are members of *both* of the listed classes in the intersection are inferred as members of this class. In other words links have to have been discovered by *both* BGP dumps.

**hardClass** This is the most conservative intersection, requiring membership of both TRASInter and BGPASInter for links to be included. Because of
Listing 8.1: as-level Ontology Code

```java
1  String defaultNameSpace = "http://nets.lboro.ac.uk/liol#";
2  String prefix = "lboro";
3  //Model and Prefix set up
4  OntModel asOnt = ModelFactory.createOntologyModel(OntModelSpec.OWL_LITE_MEM);
5  asOnt.setNsPrefix(prefix, defaultNameSpace);
6  //Create the Ontology Classes
7  OntClass TRASLink101 = asOnt.createClass(defaultNameSpace + "TRASLink101");
8  OntClass TRASLink102 = asOnt.createClass(defaultNameSpace + "TRASLink102");
9  OntClass BGPASLink1 = asOnt.createClass(defaultNameSpace + "BGPASLink1");
10 OntClass BGPASLink2 = asOnt.createClass(defaultNameSpace + "BGPASLink2");
11 OntClass easyClass = asOnt.createClass(defaultNameSpace + "easyClass");
12 OntClass allTRASLinks = asOnt.createClass(defaultNameSpace + "allTRASLinks");
13 OntClass allBGPASLinks = asOnt.createClass(defaultNameSpace + "allBGPASLinks");
14 //Subclasses for UNION option
15   allTRASLinks.addSubClass(TRASLink101);
16   allTRASLinks.addSubClass(TRASLink102);
17   allBGPASLinks.addSubClass(BGPASLink1);
18   allBGPASLinks.addSubClass(BGPASLink2);
19   easyClass.addSubClass(allBGPASLinks);
20   easyClass.addSubClass(allTRASLinks);
21   easyClass.addSubClass(TRASLink101);
22   easyClass.addSubClass(TRASLink102);
23 //Intersection classes for INTERSECTION approach
24 RDFList cs = asOnt.createList( new RDFNode[] {TRASLink101, TRASLink102} );
25 IntersectionClass TRASInter = asOnt.createIntersectionClass(defaultNameSpace + "TRASInter", cs);
26 RDFList cs1 = asOnt.createList( new RDFNode[] {BGPASLink1, BGPASLink2} );
27 IntersectionClass BGPASInter = asOnt.createIntersectionClass(defaultNameSpace + "BGPASInter", cs1);
28 RDFList cs2 = asOnt.createList( new RDFNode[] {TRASInter, BGPASInter} );
29 IntersectionClass hardClass = asOnt.createIntersectionClass(defaultNameSpace + "hardClass", cs2);
30 RDFList cs3 = asOnt.createList( new RDFNode[] {BGPASLink1, TRASLink101} );
31 IntersectionClass realClass = asOnt.createIntersectionClass(defaultNameSpace + "realClass", cs3);
```

the class hierarchy, this means that any link resource which is a member of this class must have been discovered in *every test* in the study.

**allTRASLinks** This is a straightforward super-class of the two traceroute-discovered link classes. Any link discovered in *either* TRASLink101 or TRASLink102 (or both) is inferred as a member of this class.

**allBGPASLinks** This is the super-class of the two BGP-discovered link classes. Any link discovered in *either* BGPASLink1 or BGPASLink2 (or both) is inferred as a member of this class.

**easyClass** This is the top of the sub-class hierarchy. *Any* link discovered in *any* dataset in the ontology is inferred as a member of this class, which is therefore the most liberal in its membership criteria.

**realClass** This intersection class has been defined specifically for the analysis of actual real field data in this study. The full links datasets derived from the traceroute file and the BGP dump are stored in TRASLink101 and BGPASLink1 respectively. **realClass** is the *intersection* of the two.

The main Java code used to create the ontology is shown in Listing 8.1.
8.6 Final Conversion and Testing

The initial stages of converting the original traceroute and BGP dump files as described in Section 8.3 were not summarised at all in terms of AS level information. Every individual IP address in the original traces was replaced with the mapped AS number if available, or an asterix or “n” marker if not. Also, where adjacent IP addresses were in the same AS, the AS number was simply repeated each time. The final stage not only converted the data into resources and statements in the RDF model, it also summarised the data according to certain assumptions and rules, as explained:

- Where a series of consecutive entries in the path were for the same AS number, only one entry was carried over.

- Where an “n” was present in the path (IP address there in the trace but no corresponding AS number found on look-up), it was assumed that this had been the position of a private address within the AS and so it was simply removed from the path at the AS level.

- Where an asterix, or a consecutive series of asterixes was present (no IP address(es) at that position in the original trace), it was assumed that the AS numbers on either side were in fact linked, and so the markers were simply removed.

The conversion to RDF then effectively changed the hop-by-hop path format into a model in which every AS and every link between them is a single resource, with properties \( \text{lboro:connectsFrom} \) and \( \text{lboro:connectsTo} \) linking them as appropriate. This further reduces the duplication, while losing the individual paths but retaining the who-links-to-who information. Applying these criteria led to a drastic reduction in the number of AS number references in the final RDF triplestore from those in the base data. Table 8.2 shows this reduction. The BGP source data, being already at the AS level and having no missing hop information markers, was not much reduced during the summarisation. The reductions which were made came about from duplicate AS numbers in adjacent positions in some of the paths. The traceroute derived data was reduced by a factor of more than 3, because of the “missing information marker” removal and the greater number of identical adjacent AS numbers. As would be expected, the number of unique AS numbers created in the model was very much smaller than the path data in both cases as the repetition in different paths was no longer there.

To test the functionality of the ontology, two very small subsets of the path data
were extracted from the main path files to artificially populate the class extensions of the other classes which were created, so that the intersection classes could become slightly populated. lboro:TRASLink102 received five link individuals and BGPASLink2 received only one of the five.

To test the functionality of the ontology and to evaluate the unions and intersections of the real field data, a query was created requesting the object of all statements with a `connectsFrom` and `connectsTo` property, whose subject (an AS level link) was a member of the class extension of the class of interest. The query was modified to specify each class in turn, to check the inferencing. In this case, the standard Jena ResultSetFormatter was not used, but rather the result-set of each run of the query was stepped through and the AS numbers associated with the link output. In this way, the number of links presenting as members of each individual class could be counted in turn. The query is listed once only, with a dummy class-of-interest marker rather than repeat for each test:

```sparql
PREFIX lboro: <http://nets.lboro.ac.uk/liol#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

select ?connectsFrom ?connectsTo
where{?a rdf:type lboro:***CLASS-OF-INTEREST***.
    ?a lboro:connectsFrom ?b.
    ?b lboro:hasASNum ?c.
    ?c lboro:hasASValue ?connectsFrom.
    ?a lboro:connectsTo ?e.
    ?e lboro:hasASNum ?f.
    ?f lboro:hasASValue ?connectsTo}
```

Table 8.3 shows the number of links found and the query execution time for each class viewpoint. All queries were executed on the Prawn standard desktop test machine described in Section 7.7. A sample query output, actually for the TRAS-Inter class test is shown:

<table>
<thead>
<tr>
<th>TRASInter</th>
<th>Query Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link Discovered From 786 to 3257</td>
<td></td>
</tr>
<tr>
<td>Link Discovered From 2516 to 10000</td>
<td></td>
</tr>
<tr>
<td>Link Discovered From 2519 to 10000</td>
<td></td>
</tr>
<tr>
<td>Link Discovered From 2516 to 2519</td>
<td></td>
</tr>
<tr>
<td>Link Discovered From 3257 to 2516</td>
<td></td>
</tr>
<tr>
<td>Total Links Discovered = 5</td>
<td></td>
</tr>
<tr>
<td>Query Execution Time = 146 seconds</td>
<td></td>
</tr>
</tbody>
</table>

The results show that the inferencing was working correctly. The nested intersection classes allowed different views to be taken of the links which were observed, with increasingly conservative acceptance criteria within the view. This went right up to hardClass, which would only accept links as part of the class extension if they were present in every test. In the other direction, the sub-class hierarchy built up as expected, with easyClass having the full union of all unique links found in either test. The RDF 3-D modeling principle described in Section 3.5 once again ensured that there were no duplicates.
In terms of the actual field data, a total of 61,982 links were discovered using the BGP dump and only 24,872 by traceroute. The intersection class realClass shows that 14,517 links were common to both data-sets. If the liberal interpretation is used, the traceroute contribution was 10,355 links which would have been unknown if only the BGP dump file were used. If the conservative approach were taken on the real data, only 14,517 links would be believed to exist in this study.

Another use for the populated knowledgebase is as an easy look-up for individual connections between ASes. For example, the following query could be used to show the destination of all the known links which go out from a particular AS (in this example (AS22388):

```
select ?connectsTo where {
  ?a lboro:connectsFrom lboro:AS22388 .
  ?a lboro:connectsTo ?e .
  ?e lboro:hasASNum ?f .
  ?f lboro:hasASValue ?connectsTo}
```

Query Results
Link Discovered From AS22388 to AS7660
Link Discovered From AS22388 to AS4641
Link Discovered From AS22388 to AS2500
Link Discovered From AS22388 to AS3662
Total Links Discovered = 4
Query Execution Time = 148 seconds

### 8.7 Case Study Summary and Review

The AS topology case study was carried out as an investigation into the use of ontological techniques to combine data from different viewpoints and mediate between them to resolve any differences. The vehicle chosen for this was the AS topology problem, an active research area in which different discovery techniques are known to yield different results. In this study, real Internet monitoring data from a research project at Loughborough University were used as the base. Traceroutes to over 300,000 target addresses and a BGP dump from a major ISP were obtained and converted in stages to individual RDF resources in a knowledgebase. The work addressed the existence or otherwise of links between ASes in particular; each individual observed link and each individual AS number were stored as instances of their respective ontological classes and related to each other by object properties.

An ontology was created with OWL classes and intersections, using basic set-theory to allow different combinations of the information from the two sources to be viewed. It was then possible to take a very conservative view about the level of observation needed to prove a link, or to relax this at different levels to the other extreme, where any link detected once was considered to be real. The inferences implicit in this ontology were tested with the field data and found to be operating correctly, demonstrating that the semantic web tools can be used to good effect.
with this type of network monitoring data. In addition, the knowledgebase has been demonstrated as a useful who-connects-to-who look-up system at the AS level.

The conflict resolution problem has therefore been partially addressed, with some success for this particular issue. The class hierarchy could be developed to accommodate other more subtle views into the various data-sets if needed; a framework has been established. This has been principally a proof-of-concept exercise and a practical method for data conflict resolution has been demonstrated using an ontological approach and semantic web technology on genuine field data from the Internet.
Table 8.2: AS Number to RDF Conversion

<table>
<thead>
<tr>
<th></th>
<th>Traceroute Discovery Source</th>
<th>BGP Discovery Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Path data</td>
<td>6,236,748</td>
<td>1,974,591</td>
</tr>
<tr>
<td>Summarised Path Data</td>
<td>1,087,823</td>
<td>1,854,496</td>
</tr>
<tr>
<td>Unique ASes to the Model</td>
<td>14,180</td>
<td>43,778</td>
</tr>
</tbody>
</table>

Table 8.3: AS Links by Ontology Class View

<table>
<thead>
<tr>
<th>Class View</th>
<th>Number of Links</th>
<th>Query Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRASLink101</td>
<td>24,872</td>
<td>183 sec</td>
</tr>
<tr>
<td>BGPASLink1</td>
<td>61,982</td>
<td>197 sec</td>
</tr>
<tr>
<td>TRASLink102</td>
<td>5</td>
<td>165 sec</td>
</tr>
<tr>
<td>BGPASLink2</td>
<td>1</td>
<td>172 sec</td>
</tr>
<tr>
<td>TRASInter</td>
<td>5</td>
<td>146 sec</td>
</tr>
<tr>
<td>BGPASInter</td>
<td>1</td>
<td>178 sec</td>
</tr>
<tr>
<td>hardClass</td>
<td>1</td>
<td>141 sec</td>
</tr>
<tr>
<td>allTRASLinks</td>
<td>24,872</td>
<td>306 sec</td>
</tr>
<tr>
<td>allBGPASLinks</td>
<td>61,982</td>
<td>224 sec</td>
</tr>
<tr>
<td>easyClass</td>
<td>72,337</td>
<td>366 sec</td>
</tr>
<tr>
<td>realClass</td>
<td>14,517</td>
<td>278 sec</td>
</tr>
</tbody>
</table>
Chapter 9

Conclusions, Contribution and Future Direction

The early research in this work identified that interpretation of network monitoring and measurement data is a difficult task. A gap was identified between the basic data, which are either available already or can be gathered locally, and the level of information which is actually of practical use to stakeholders for the management of networks. The tools and techniques which have been developed and standardised for the Semantic Web were identified as potentially providing a method for tackling this interpretation issue, using an ontological approach to bridge the gap. These were proposed as offering a solution and the practical work set out to test that concept.

The *Loughborough Internetworking Ontology Library* (LIOL) was created, by applying knowledge of the networking domain developed by research during the first part of the project combined with existing experience-based expertise of the researcher and by the use of a substantial range of the semantic web technologies with Jena semantic web framework. The LIOL is a set of autonomous reference ontologies which are intended to give an ontological framework of the computer networking domain, to be used on a mix-and-match basis as a starting point to underpin more detailed case-level work or future projects. This is a new approach in this field. LIOL represents a free-standing contribution of this research in its own right and is to be made available to any researchers engaged in similar work. It gives a framework to model the complete domain of Computer Networking, being extensible in any new direction but at the same time covering the current basic concepts. It has been thoroughly tested for inferencing and some of the autonomous ontologies were implemented in practice as the foundation for the practical work which was then conducted as part of this specific project.
The interpretation issue was first broken down into four specific problems for consideration, syntactical, semantic, expertise and conflict resolution. These turned out not to be quite so separate, but in practice they were found to overlap considerably with each other. Two case studies involving data gathered from real production networks were conducted, as vehicles to evaluate the concept. Between them they cover the four problems and are each reviewed at the end of their respective chapters, this chapter presents an overall assessment and addresses the success criteria originally set out in introductory Section 1.5.

1. *The syntactical problem* concerns formatting, presentation and the way data are made available. It can be a real barrier to making practical use of monitoring data. RDF offers a structured and highly standardised yet very flexible approach to data storage and has been used here as the single storage platform for data of every type used in the two studies. This was achieved by developing converters, in the form of Java classes. It could be argued that much of the solution to the syntactical problem lies in the converters themselves, and that the storage medium could have been something other than RDF triples, for example a relational database. However, the advantage of RDF which became very clear in the these studies is that all the triples, whether data instances or ontological statements, can be simply brought together in no particular order in a triple-store. This provides a great deal of flexibility, as new sources of any type which came along could easily be added to the store without going “back to the drawing board” each time as would be the case if using linked tables. This not only makes the storage solution easier, but allows the converters to be written in a reusable way, meaning that the syntactical problem is tackled once, not multiple times. One of the downsides to RDF in this comparison is query speed, research [3] suggests that triple-stores are considerably slower than an equivalent relational database. However, in the VPN case study, practical solutions were developed in a medium-sized organisational network without access speed problems in most cases, so this was not a barrier.

2. *The semantic problem* is about meaning. Even if the syntactical problem can be overcome, the problems of determining whether two concepts are the same or different, how concepts relate to each and so on remain. The ontological approach using the semantic web languages was used in both studies to define such relationships in a very practical way. For example, it was an easy task to assert that a destination IP address in a TCP flow was the same concept as an address of a server interface, both being members of sub-classes of the
CHAPTER 9. CONCLUSIONS, CONTRIBUTION AND FUTURE DIRECTION

IP address class. Sub-class and sub-property assertions were used along with other RDFS and OWL constructs to link networking ideas together, which made developing queries to answer real questions much easier than would be possible with more “flat” data storage. The other side to this is that when dealing with such powerful inferencing mechanisms, it was found to be very easy to get it wrong, particularly when using an existing property for a new use. Unexpected incorrect consequences further down the line can easily be created if care is not taken in the design of the these ontologies. It is a highly skilled business to get right but overall the approach did have a significant impact on the semantic problem.

3. The expertise problem was found to overlap with the semantic problem to a large extent. In a sense, knowledge about the domain is embedded at every stage. For example, taking captured packet headers and using them to construct a model of network flows in the correct form required significant domain knowledge, which was effectively then embedded. More specifically, the use of reasoners and rules in the VPN study has clearly demonstrated that this approach goes beyond the conversion exercise and it can be used successfully in the networking domain to embed expertise about real issues of interest to network administrators. It has been shown to facilitate a situation where problems need to be solved once, not multiple times.

4. The conflict resolution problem is addressed in the AS topology study, which uses a system using basic set-theory of intersections and unions is used as a way of bringing together conflicting data, facilitating different views into the conjoined data, which can be selected depending on the viewers’ expert opinion. The technique worked very well in this example and so does demonstrate the concept and show that at least a partial solution to the conflict resolution problem can be given by these techniques. Other approaches could be adopted in different situations, with possibly the use of rules being appropriate in some cases.

Some scaling issues were encountered while using the basic desk-top machine. In each case these caused out-of-memory issues. Converting and storing large amounts of data in a single triple-store and queries with large result-sets were the two situations when this problem arose. The practical solutions to these issues were to store triples in several smaller stores rather than one large one, and to design more focussed queries.

The two case studies involving real network data have been successfully completed.
They have involved a significant amount of development work, constructing converters, knowledgebases and tools which have been used to address some typical questions for which answers are needed in the field. This has proven the concept that an ontological approach can be used successfully in practice to tackle interpretation problems and provide useful information in this domain. Demonstrating this represents a further contribution of the work. Some potential problems have also been identified, including the risks of incorrectly specifying ontological relationships and producing unwanted and incorrect inferences. Despite these, the concept has been substantially proven.

Using an RDF triplestore (in this work Jena TDB) has been shown to be a viable and very attractive method for storing both active measurements and passive monitoring data of the sort needed by a monitoring system for small commercial computer networks. The best results were obtained when some intelligent data compression or summarisation was done in the stage before full conversion of base data to standard RDF. Particularly because the main purpose was to get some useful, high-level information at the end, RDF seems to lend itself best to this thoughtful, pre-processing, lossy approach to data storage. Simple storage of large, complete sets of base data in a lossless manner, with no idea of the ultimate use for the data is probably best left to traditional database methods rather than the current generation of RDF triplestores. The real power of storing information in an RDF triplestore is the standardisation discussed above and the ability to link these triples to ontological statements by importing them into a model. Reasoning over such a model yields inferred triples, or entailments, which can provide valuable network management information from a well-designed ontology. Possibly the most surprising result of this work is that a great deal of the reasoning needed to create an ontological network monitoring system can be obtained from RDFS alone, with only a small number of the more powerful OWL constructs needed. The RDFS domain and range properties proved to be particularly useful and powerful inferencing methods, although they needed care in use to avoid unwanted and incorrect consequences.

The AS level topology study was used as a vehicle for testing the use of ontologies to resolve data conflicts in the networking domain. The basic set-theory approach used in the ontology allowed different views to be adopted depending on confidence in the base data. This approach was very successful and could actually be used with very little change to many other, non-networking situations. It therefore offers a potentially very powerful solution to practically modeling and automating data conflict resolution generally. The ontology uses more power-
ful reasoning constructs that those used in the other case study, particularly the OWL intersection class. The complimentary set theory idea of a union was more straightforward in this case, being modeled using the RDFS subclassOf property.

The use of semantic rules, (in this research JenaRules) proved to be a particularly successful method of embedding expertise into the ontologies. Rules have been demonstrated to be a straightforward and useful way to create new entailments under particular network monitored conditions. These entailments are easy to search for using simple queries and have been used to not only identify network pathologies from the combinations of disparate observed data, but also to suggest actions to resolve those pathologies.

Future work could go in a number of different directions. On the development side, the solution developed for the VPN case study is already the heart of a practical monitoring system for use on a real network. However further development is needed to build a user interface and to refine the code, make it more robust and more modular. This could potentially lead to a practical ontological network monitoring system as a product in its own right.

In terms of network research, the notion of embedding expertise at a more and more sophisticated level in network monitoring ontologies is a fascinating direction in which this work could continue. A starting point would be to further leverage the inferencing possibilities of semantic web rules, in collaboration with network managers in some specific network to build up a library of pathology detection rules and solution suggestions. Also of great interest would be developing more advanced techniques for data conflict resolution which has much potential using these techniques.

The conflict resolution ontology developed for the AS level topology study has the potential to be more widely used. A future research direction could be to test and further develop these techniques in completely different domains of interest. Areas such as resolving conflicting accounts of events in history or even combining key aspects of different religious belief systems could be investigated, for example. The conflict resolution problem is basically the same whenever two or more sources of incomplete or inaccurate information about the same subject are brought together. This work has contributed one possible solution.
References


REFERENCES


REFERENCES


REFERENCES


Appendix A

Industrial Partner Meeting Notes

A.1 TRML Meeting

Notes from a meeting held at the TRM Ltd offices, Elmesthorpe, Leics

Meeting Date: 14th October 2009

Present:
Mr Martyn Bright (MB), MD of TRM Ltd
Mr Andy Stockton (AS), Technical Director of TRM Ltd
Mr Ian Napier (IN), Research Student of Loughborough University

IN explained the nature of the research he was undertaking at Loughborough University, and that he hoped to work with Industrial partners wherever possible for testing the ideas about the use of semantic web languages in networking in a suitable testbed. Ideally a research server would be given access to some part of a live commercial network for packet capture, and if acceptable some active probe testing.

MB explained that one of their customers (Organisation A) had a fairly complex network of VPN tunnels interconnecting branches, and that problems do occur. They had a good relationship with them and he thought they would be willing to cooperate.

He and AS and other senior staff are often out of the office and when things go wrong sometimes the junior staff, although very good on desktop support don’t know how to fix networking problems, especially when related to the Internet. Any system that could help with the expertise problems and provide any information to use when wrangling with ISPs on the phone would also be good. He offered to propose to the customer that IN should be allowed to access their network for this research on certain conditions:

• No disruption at all to operations and no time overhead for TRML or Organ-
isation A staff.

- No security risks to be introduced to the network.

- No application layer information to be captured or even observed at all during the project.

- Any access to the Organisation A premises to be strictly with TRML staff only.

- No significant network capacity to be used during working hours, and only with agreement out of hours.

- IN to write a brief document explaining what was to be done, why and any potential benefits which may be had later on.

IN agreed to this, and MB kindly offered to arrange the first visit during the following few weeks and to set up a VPN tunnel on a loaned Draytek router which could be housed at IN’s home and would give access to the Organisation-A network. This was not to be made available at the University or other public place. This was to be for IN’s use only.
Appendix B

Programming Framework Selection

B.1 Assessment Criteria

It was envisaged that this project would involve building a number of quite varied types of knowledgebases and applications in achieve the research aims. There were therefore quite demanding requirements which needed to be met by the programming framework (or frameworks) to be utilised. These were condensed into the five criteria listed:

**Triple Store.** Persistent storage of triples, both instance data and ontological constructs, is an essential requirement.

**Reasoning.** It was known early on in the process that this research would involve a great deal of experimentation with the application of different RDFS and OWL constructs to networking ontologies and instance data. This was essential to assess the practicality and scalability, in terms of semantic richness, of this ontological approach. Therefore it was important to select a programming framework which was not limited in its scope to apply a full range of properties, so that any applications subsequently created could be tested to their limit in this regard. In practice, this means having good support for OWL.

**Query and Rules Functionality** As described earlier, the ability to reason across a knowledgebase and correctly apply the chosen logical constructs from RDFS and OWL is essential. The ability to create queries to extract data for external applications is equally vital. It was considered that in the network management area, particularly for diagnostic use, the rules slice of the semantic web cake could be of particular value, so also had to be supported.
Usability. The premise for this research is that the semantic web languages, tools and frameworks should have reached a sufficient level of maturity and reliability to make their application to network management and monitoring a practical proposition. Therefore the requirement here is for a stable, well documented and supported framework which can be applied, possibly in novel ways with some minor modification, to that domain. The emphasis was not on major coding exercises to create new prototype frameworks, but rather to concentrate coding efforts on creating and testing network-specific ontologies.

User Community. A major advantage in any such research is the existence of a thriving on-line user community and discussion groups for the tools of choice. Accessing an active user forum can be an excellent way of getting help with the low level, detailed coding issues and of keeping up to date with developments. Tips and advice may be gleaned which go beyond the more formal tutorials and documentation, so this was also one of the criteria for framework selection.

Four possible frameworks were shortlisted from an initial web search for assessment against these criteria. They were each assessed in terms of the claimed functionality, and then downloaded and installed for some basic tests. These evaluations, particularly of usability, are inherently subjective and reflect the experience of researcher when trying out these tools in practice.

B.2 Redland

Redland is described on the application home page\(^1\) as \textit{A set of free software C libraries that provide support for the Resource Description Framework (RDF)}. Although C based, it has bindings for Perl, PHP, Python and Ruby via a separate Redland bindings package. Further add-on packages are available for parsing and serialising RDF (Raptor) and for initiating queries with either SPARQL or RDQ (Rasqal). Persistent storage is possible using a range of database options, including MySQL and Berkeley DB. There is no native persistent storage mechanism. Support for reasoning, inference and rules is not provided directly and would need to be realised using other tools if Redland were the main frameset used.

The core Redland API package (version 1.0.8.1) was downloaded compiled and installed. In a 2002 paper by Beckett [2], the creator of Redland, he describes the advantages and disadvantages of implementing Redland in C. The code is described as \textit{rather low level and tricky}, which proved to be the case in this trial.

\(^1\url{http://librdf.org}, \text{last viewed 1st August 2013}\)
A simple RDF graph was eventually created, but along the way there were many problems with compilation errors and apparently missing libraries. Unfortunately there was no tutorial available on the Redland website, just a blank page. Overall, usability was not considered good. Searching for help from user fora was also a fruitless exercise. It may be that Redland would be an excellent choice for a very experienced C programmer, who could possibly benefit from the flexibility such a low level set of libraries provides but it did not perform well “out of the box” for a moderately experienced user.

B.3 RDFLib

RDFLib is a pure Python based package for working with RDF.\(^2\) It includes parsers and serialisers, a graph interface and has the ability to implement persistent storage using Berkeley DB. RDFLib has full support for SPARQL queries. There is no direct inferencing support, but a companion package, FuXi\(^3\) can be added to give RDFS and OWL reasoning. The overview states that FuXi aims to be the engine for contemporary expert systems based on the semantic web technologies.

The RDFLib package (Version 3.1) was successfully downloaded and installed with no problems. The tutorial and user guide takes the new user right though from getting started to implementing a model. A simple network-based model was created fairly easily. It took a little while to create the first query, the SPARQL syntax appeared very awkward and unforgiving of errors but after overcoming initial problems the process worked well. Overall, usability was very good and answers to sample problems could generally be found either on the website or from other users posting on-line. A slight concern was that there is much less help available for the FuXi package, whose functionality would be an essential requirement.

B.4 Sesame

The openrdf website\(^4\) describes Sesame as an open source Java framework for storage and querying of RDF data. It has a disk-backed triple store and support for SPARQL queries. Inference for RDFS is provided in the core framework. OWL reasoning can be added when linking to the OWLIM RDF database management system\(^5\).

\(^2\)https://rdflib.readthedocs.org, last viewed 1st August 2013
\(^3\)http://github.com/RDFLib/FuXi, last viewed 1st August 2013
\(^4\)http://www.openrdf.org/doc/sesame2/users/ch01.html, last viewed August 2013
\(^5\)www.ontotext.com/owlim last viewed August 2013
Sesame has comprehensive documentation, start-up guide and tutorials. Getting started was particularly easy by downloading the single Java Archive file onejar.jar, which contains all the core Sesame libraries. This was easily entered into the classpath, for use in the Netbeans IDE. A simple rdf model was created and eventually queried, following the Sesame user guide. Overall the usability was very good. Easy to understand documentation is available and few problems came up when following the examples. The OWLIM DMS was not tested, but is described on the W3C semantic web wiki\(^6\) as the most scalable semantic repository.

### B.5 Jena

Jena is the framework most mentioned in texts on semantic web referenced in this research, and is the tool set of choice as a teaching aid by Hebeler et al \([52]\). It is a complete package, including support for RDFS and OWL reasoning using the modelFactory libraries, and its own rule system, JenaRules. It also supports SPARQL queries and an easy plug-in approach for other reasoners, such as pellet\(^7\).

Just like Sesame, Jena is used as a Java API, and is downloaded as a group of Java Archive (.jar) files which can easily be entered into the classpath. The Netbeans IDE was used to build an RDF model and run some simple queries. The tools to produce RDFS or OWL ontology models are included and are fairly easy to use. There are two ways to implement persistent storage included in the core package, SDB and TDB. The Jena documentation and tutorials are very comprehensive and the on-line user community is vast.

### B.6 Conclusions

Redland surely has a niche role and a loyal following, but falls short of the specified criteria on grounds of both usability and reasoning functionality. RDFLib is a very user friendly python package, which meets all the criteria in principle. The slight concern is with the FuXi companion package, which seems to be less well known and supported and so is judged to be a risk in terms of exercising the full functionality required. Sesame and Jena are both very professionally presented and well established. They both meet all the criteria, although Sesame relies on a separate product (OWLIM) for persistent storage and crucially for OWL inferencing. Jena has built-in OWL reasoning and also has TDB as a more integrated persistent storage option but can also connect to external database management systems.

---

\(^6\)www.w3.org/2001/sw/wiki/OWLIM, last viewed August 2013
\(^7\)http://clarkparsia.com/pellet, last viewed August 2013
(including OWLIM if required). Considering all this, Jena was selected as the core programming framework. These conclusions are summarised in table 5.1
Appendix C

LIOL RDF/XML Listings

C.1 The Organisation AO RDF/XML Listing

Listing C.1: The Organisation AO RDF/XML Listing

```xml
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:lboro="http://nets.lboro.ac.uk/liol#"
>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#userOrg">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#networkOrg"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#overseenBy">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#rirOrg">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#intGroupOrg"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#endUserCo">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#userOrg"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#lacnic">
    <lboro:overseenBy rdf:resource="http://nets.lboro.ac.uk/liol#ICANN"/>
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#rirOrg"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#ISOC">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#intGroupOrg"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#govOrg">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#userOrg"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#regulatorOrg"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#apnic">
    <lboro:overseenBy rdf:resource="http://nets.lboro.ac.uk/liol#ICANN"/>
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#rirOrg"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#IETF">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#intGroupOrg"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#IETF">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#intGroupOrg"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#ICANN">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#intGroupOrg"/>
  </rdf:Description>
</rdf:RDF>
```

147
C.2 The Protocols AO RDF/XML Listing

Listing C.2: The Protocols AO RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#">
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#userProt">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#appLayerProt"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#UDPProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#tranLayerProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#RIPProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#routingProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#netLayerProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#networkProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#DHCPProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#userProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#physLayerProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#networkProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#transportProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#networkProt"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#linkedProt">
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/liol#networkProt"/>
  </rdf:Description>
</rdf:RDF>
```
APPENDIX C. LIOL RDF/XML LISTINGS

C.3 The Traffic ao AO RDF/XML Listing

Listing C.3: The Traffic AO RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#" >

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasSrcIP">
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#IPAddr"/>
    <rdfs:subPropertyOf rdf:resource="http://nets.lboro.ac.uk/liol#hasIP"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasUDstIP">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#UDPFlow"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#IPAddr"/>
    <rdfs:subPropertyOf rdf:resource="http://nets.lboro.ac.uk/liol#hasDstIP"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#TCPFlow">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#IPCommunication"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#ICMPComm">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#IPCommunication"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasPort">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasTSrcPort">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#TCPFlow"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#IPAddr"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasDStPort">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#UDPFlow"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#IPAddr"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>

</rdf:RDF>
```
C.4 The Infrastructure AO RDF/XML Listing

Listing C.4: The Infrastructure AO RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#">
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#ASNumber">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
</rdf:RDF>
```
C.5 The Applications AO RDF/XML Listing

Listing C.5: The Applications AO RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#">
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hostedOn">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#serverHW"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#implAppProt">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appLayerProt"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#appType">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#implPort">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#appImpl">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#implTranProt">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#tranLayerProt"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#serverHW">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasAppType">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appType"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#appLayerProt">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#tranLayerProt">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
</rdf:RDF>
```

C.6 The Monitoring AO RDF/XML Listing

Listing C.6: The Monitoring AO RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#">
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#metricUsedBy">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#metrics"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#m3AppImpl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>  
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#processTool">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#tool"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#implAppType">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appType"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasAppType">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appType"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#metricUsedBy">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#metrics"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#m3AppImpl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>  
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#metricUsedBy">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#metrics"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#m3AppImpl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>  
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#processTool">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#tool"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#implAppType">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appType"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hasAppType">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#appType"/>  
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
</rdf:RDF>
```
<?xml version="1.0" encoding="UTF-8"?>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#appImpl">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#m3AppImpl">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#appImpl"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#tool">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#locDatSource">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#dataSource"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#getsM3DataFrom">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#processTool"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#dataSource"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#producesMetric">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#processTool"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#metrics"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#sendsLocDataTo">
    <rdfs:domain rdf:resource="http://nets.lboro.ac.uk/liol#monitorTool"/>
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/liol#locDatSource"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#extDatSource">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#dataSource"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#monitorTool">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#tool"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#metrics">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
</rdf:RDF>
Appendix D
Local Ontology Listing

D.1 VPN Study Local Ontology RDF/XML Listing

Listing D.1: The VPN Study Local Linking Ontology RDF/XML Listing

```xml
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:orga="http://nets.lboro.ac.uk/orga#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:lboro="http://nets.lboro.ac.uk/liol#" >
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#Glasgow_Branch">
    <orga:hasRouter rdf:resource="http://nets.lboro.ac.uk/orga#glRouter"/>
    <orga:hasSiteLAN rdf:resource="http://nets.lboro.ac.uk/orga#glLAN"/>
    <orga:hasPublicIP rdf:resource="http://nets.lboro.ac.uk/orga#217.206.141.218"/>
    <orga:hasLocalIP rdf:resource="http://nets.lboro.ac.uk/orga#192.168.20.254"/>
    <orga:hasName>Glasgow</orga:hasName>
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/orga#site"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#192.168.60.254">
    <orga:hasIPSubnet>192.168.60.0</orga:hasIPSubnet>
    <orga:hasIPValue>192.168.60.254</orga:hasIPValue>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#Glasgow_Tunnel">
    <orga:tunnelsTo rdf:resource="http://nets.lboro.ac.uk/orga#Glasgow_Branch"/>
    <orga:hasName>Glasgow-Tunnel</orga:hasName>
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/orga#vpnTunnel"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#hasSrcIP">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#hasPublicIP">
    <rdfs:range rdf:resource="http://nets.lboro.ac.uk/orga#IPAddr"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#82.110.55.2">
    <orga:hasIPValue>82.110.55.2</orga:hasIPValue>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#83.70.138.249">
    <orga:hasIPValue>83.70.138.249</orga:hasIPValue>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/orga#Bristol_Tunnel">
    <orga:tunnelsTo rdf:resource="http://nets.lboro.ac.uk/orga#Bristol_Branch"/>
    <orga:hasName>Bristol-Tunnel</orga:hasName>
    <rdf:type rdf:resource="http://nets.lboro.ac.uk/orga#vpnTunnel"/>
  </rdf:Description>
</rdf:RDF>
```
APPENDIX D. LOCAL ONTOLOGY LISTING
APPENDIX D. LOCAL ONTOLOGY LISTING
Appendix E

AS Level Ontology Listing

E.1 AS Level Ontology RDF/XML Listing

Listing E.1: The AS Case Study Ontology RDF/XML Listing

```xml
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:lboro="http://nets.lboro.ac.uk/liol#">
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#allTRASLinks">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <rdf:rest rdf:nodeID="A1"/>
    <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#BGPASInter"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#BGPASInter">
    <owl:intersectionOf rdf:nodeID="A2"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#TRASInter"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#TRASLink101">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allTRASLinks"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#realClass">
    <owl:intersectionOf rdf:nodeID="A3"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allTRASLinks"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#hardClass">
    <owl:intersectionOf rdf:nodeID="A0"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allTRASLinks"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#allBGPASLinks">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allTRASLinks"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A4">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#TRASLink102"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#BGPASLink2">
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#easyClass"/>
    <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allBGPASLinks"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
</rdf:RDF>
```
<rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#BGPASLink1">
  <rdfs:subClassOf rdf:resource="http://nets.lboro.ac.uk/liol#allBGPASLinks"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>

<rdf:Description rdf:nodeID="A5">
  <rdf:rest rdf:nodeID="A4"/>
  <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#TRASLink101"/>
</rdf:Description>

<rdf:Description rdf:nodeID="A2">
  <rdf:rest rdf:nodeID="A6"/>
  <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#BGPASLink1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="A7">
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#TRASLink101"/>
</rdf:Description>

<rdf:Description rdf:nodeID="A3">
  <rdf:rest rdf:nodeID="A7"/>
  <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#BGPASLink1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="A6">
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  <rdf:first rdf:resource="http://nets.lboro.ac.uk/liol#BGPASLink2"/>
</rdf:Description>

<rdf:Description rdf:about="http://nets.lboro.ac.uk/liol#easyClass">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>

</rdf:RDF>
Appendix F

Key to Methodology Diagrams

Figure F.1: A key to the Methodology Process Diagrams
Appendix G

Key to Ontology Diagrams

Figure G.1: A Key to the Ontology Diagrams