International Journal on Advances in Internet Technology

2008 vol. 1 nr. 1

The *International Journal On Advances in Internet Technology* is Published by IARIA.
ISSN: 1942-2652
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal On Advances in Internet Technology
Volume 1, Number 1, 2008

Editorial Board

First Issue Coordinators

- Shingo Ata, Osaka City University, Japan
- Eugen Borcoci, University "Politehnica" of Bucharest (UPB), Romania
- Michel Diaz, LAAS, France
- Michael Menth, University of Wuerzburg, Germany
- Michal Pioro, University of Warsaw, Poland
- Joel Rodriques, University of Beira Interior, Portugal
- Zary Segall, University of Maryland, USA

Ubiquitous Systems and Technologies

- Sergey Balandin, Nokia, Finland
- Matthias Bohmer, Munster University of Applied Sciences, Germany
- David Esteban Ines, Nara Institute of Science and Technology, Japan
- Dominic Greenwood, Whitestein Technologies AG, Switzerland
- Arthur Herzog, Technische Universitat Darmstadt, Germany
- Malohat Ibrohimova, Delft University of Technology, The Netherlands
- Reinhard Klemm, Avaya Labs Research-Basking Ridge, USA
- Joseph A. Meloche, University of Wollongong, Australia
- Ali Miri, University of Ottawa, Canada
- Said Tazi, LAAS-CNRS, Universite Toulouse 1, France

Systems and Network Communications

- Eugen Borcoci, University 'Politechncia' Bucharest, Romania
- Anne-Marie Bosneag, Ericsson Ireland Research Centre, Ireland
- Jan de Meer, smartspace®lab.eu GmbH, Germany
- Michel Diaz, LAAS, France
- Tarek El-Bawab, Jackson State University, USA
- Mario Freire, University of Beria Interior, Portugal / IEEE Portugal Chapter
- Sorin Georgescu, Ericsson Research - Montreal, Canada
- Huaqun Guo, Institute for Infocomm Research, A*STAR, Singapore
- Jong-Hyouk Lee, Sungkyunkwan University, Korea
- Wolfgang Leister, Norsk Regnesentral (Norwegian Computing Center), Norway
- Zoubir Mammeri, IRIT - Paul Sabatier University - Toulouse, France
- Sjouke Mauw, University of Luxembourg, Luxembourg
- Reijo Savola, VTT, Finland

Future Internet

- Thomas Michal Bohnert, SAP Research, Switzerland
- Fernando Boronat, Integrated Management Coastal Research Institute, Spain
- Chin-Chen Chang, Feng Chia University - Chiayi, Taiwan
- Bill Grosky, University of Michigan-Dearborn, USA
- Sethuraman (Panch) Panchanathan, Arizona State University - Tempe, USA
- Wei Qu, Siemens Medical Solutions - Hoffman Estates, USA

- Thomas C. Schmidt, University of Applied Sciences – Hamburg, Germany

Challenges in Internet

- Olivier Audouin, Alcatel-Lucent Bell Labs - Nozay, France
- Eugen Borcoci, University "Politehnica" Bucharest, Romania
- Evangelos Kranakis, Carleton University, Canada
- Shawn McKee, University of Michigan, USA
- Yong Man Ro, Information and Communication University - Daejon, South Korea
- Francis Rousseaux, IRCAM, France
- Zhichen Xu, Yahoo! Inc., USA

Advanced P2P Systems

- Nikos Antonopoulos, University of Surrey, UK
- Filip De Turck, Ghent University – IBBT, Belgium
- Anders Fongen, Norwegian Defence Research Establishment, Norway
- Stephen Jarvis, University of Warwick, UK
- Yevgeni Koucheryavy, Tampere University of Technology, Finland
- Maozhen Li, Brunel University, UK
- Jorge Sa Silva, University of Coimbra, Portugal
- Lisandro Zambenedetti Granville, Federal University of Rio Grande do Sul, Brazil

International Journal On Advances in Internet Technology
Volume 1, Number 1, 2008

Foreword

Finally, we did it! It was a long exercise to have this inaugural number of the journal featuring extended versions of selected papers from the IARIA conferences.

With this 2008, Vol. 1 No.1, we open a long series of hopefully interesting and useful articles on advanced topics covering both industrial tendencies and academic trends. The publication is by-invitation-only and implies a second round of reviews, following the first round of reviews during the paper selection for the conferences.

Starting with 2009, quarterly issues are scheduled, so the outstanding papers presented in IARIA conferences can be enhanced and presented to a large scientific community. Their content is freely distributed from the www.iariajournals.org and will be indefinitely hosted and accessible to everybody from anywhere, with no password, membership, or other restrictive access.

We are grateful to the members of the Editorial Board that will take full responsibility starting with the 2009, Vol 2, No1. We thank all volunteers that contributed to review and validate the contributions for the very first issue, while the Board was getting born. Starting with 2009 issues, the Editor-in Chief will take this editorial role and handle through the Editorial Board the process of publishing the best selected papers.

Some issues may cover specific areas across many IARIA conferences or dedicated to a particular conference. The target is to offer a chance that an extended version of outstanding papers to be published in the journal. Additional efforts are assumed from the authors, as invitation doesn't necessarily imply immediate acceptance.

This particular issue covers papers invited from those presented in 2007 and early 2008 conferences. The papers reflect the emerging Internet-related technologies. One aspect is related to real-time traffic monitoring and management, including tracking on P2P information sharing. Another trend concerns development design and development of system properties and system monitoring using the Web Services paradigm.

We hope in a successful launching and expect your contributions via our events.

First Issue Coordinators,
Jaime Lloret, Universidad Politécnica de Valencia, Spain
Pascal Lorenz, Université de Haute Alsace, France
Petre Dini, Cisco Systems, Inc., USA / Concordia University, Canada

International Journal On Advances in Internet Technology
Volume 1, Number 1, 2008

CONTENTS

A Case Study on Integrating Extra-Functional Properties in Web Service Model-Driven Development: from Model to Code

Guadalupe Ortiz
Quercus Software Engineering Group
Centro Universitario de Mérida, UEX
Mérida, Spain
gobellot@unex.es

Juan Hernández
Quercus Software Engineering Group
Escuela Politécnica, UEX
Cáceres, Spain
juanher@unex.es

Abstract— Being one of the most promising current technologies, Web Services are at the crossing of distributed computing and loosely coupled systems. Although vendors provide multiple platforms for service implementation, service integrators, developers and providers demand approaches for managing service-oriented applications at all stages of development. In this sense, approaches such as Model-Driven Development (MDD) and Service Component Architecture (SCA) can be used jointly for modeling and integrating services regardless of the underlying platform technology. Besides, WS-Policy provides a standard description for extra-functional properties, which remains independent of both the final implementation and the binding to the service in question. In this paper we show a case study in which the aforementioned MDD, SCA and WS-Policy are assembled in order to develop web services and their extra-functional properties from a platform independent model, which is later transformed into platform specific ones and then into code.

Keywords:Extra-functional property, Web service, model-driven development, aspect-oriented techniques, WS-policy.

I. INTRODUCTION

Web Services provide a successful way to communicate distributed applications, in a platform independent and loosely coupled manner, providing the systems with ample flexibility and more manageable maintenance. Although development middlewares provide a splendid environment for service implementation, methodologies for earlier stages of development, such as the modeling stage, are not provided in a cross-disciplinary scope, whereby, for instance, the automatic model-implementation transformation or the addition of extra-functional elements would be feasible.

Academy and industry are beginning to focus on the modeling stage, where it is also pursued to keep the loosely coupled notion and independence from the platform [13]. Some rising proposals focus on representing the service as a component and others base the model on WSDL elements; representative approaches are described below:

To start with, *Service Component Architecture* (SCA) and *Service Component Description Language* (SCDL) provide a way to define interfaces and references independently of the final implementation technology, which will be bound subsequently [3]. According to SCA, services are modeled as components. These components are linked to a given interface, which can be later specified in a particular one. Besides, the components will show the required references for their behavior to be completed. This proposal provides the following advantages: first of all, it defines a very high level model, allowing the developer to bind it to a specific technology at a later stage. Secondly, the model can be implemented by using different approaches such as Java, BPEL and States Machine, therefore permitting adaptability to the client's specific needs, or to the most suitable option for its integration in a specific environment. Thirdly, the model can be converted into XML, providing an intermediate language to integrate different party models into a unique system. However, this proposal does not face how to integrate this definition with other stages of development, such as implementation.

As a second trend, many proposals are emerging in the literature where a Model Driven Architecture (MDA) approach is being applied to web service development. MDA has been proposed to facilitate the programming task for developers by dividing system development into three different phases: a *Platform Independent Model* (PIM), a *Platform Specific Model* (PSM) and, finally, the code. Thus, MDA solves the integration of the different stages of development, as mechanisms are provided to model applications in a platform independent manner which may be later transformed into the specific required models and eventually into final code, but it does not provide a specific way to do so for service technology.

Let us consider now that we want to provide our modeled services with extra-functional properties, that is, with additional pieces of code which are not part of the main service functionality. It is suggested by the SCA specification that this type of property may be modeled at a different level; the way to do so and to include them in additional stages of development has not been approached as yet. Alternatively, the named MDA proposals do not consider how extra-functional properties may be included in modeled services. On the other hand, WS-Policies have emerged as a standardized way for describing extra-functional service capabilities by using the XML standard

[17]. This allows properties to remain completely decoupled when described and there is no need to establish dependences from the service description file (WSDL) to the policy ones; property description is not linked to a specific implementation, either, maintaining the platform's independent environment. However, WS-Policy does not determine how the properties are to be modeled or implemented, and an additional mechanism would be necessary so as to integrate property modeling and implementation with their description in service-based systems.

In this paper we show a case study in which a proposed model-driven methodology is applied in order to deal with extra-functional property integration in web service development, extending our previous work on the topic [9].

The rest of the paper is organized as follows: Section 2 gives an overview of the steps followed in this approach. Section 3 shows how the PIM should be implemented. Then, Section 4 explains the PSM stage, where Section 4.1 shows the specific metamodels; Section 4.2 explains the rules used for PIM to PSM transformation and, finally, Section 4.3 shows the specific models obtained from the case study PIM. Section 5 explains the rules used to obtain code from PSMs and the final generated code. Other related approaches are examined in Section 6, whereas discussion and conclusions are presented in Section 7.

II. MODEL-DRIVEN TRANSFORMATIONS

In this section we are going to provide a general overview of the presented approach, describing the order to be followed to face web services and extra-functional property development from platform independent model to code, which will be explained in detail in the next sections.

We will use one UML profile and an additional stereotype in order to define our case study platform independent model. Thus, UML will be our PIM metamodel and the developer will be able to design the system by using common standard modeling tools at this stage of development. UML is MOF compliant, and so will the PIM metamodel. The extra-functional property profile defines the abstract stereotype *extra-functional property*, which will extend *operation metaclass* or *interface metaclass*. The extra-functional property provides five attributes: the first one is *actionType*, which indicates whether the property functionality will be performed *before*, *after* or *instead of* the stereotyped operation's execution – or if no additional functionality is needed it will have the value *none,* only possible in the client side. Secondly, the attribute *optional* will allow us to indicate whether the property is performed optionally –the client may decide if it is to be applied or not– or compulsorily –it is applied whenever the operation is invoked. Then, a third attribute, *ack*, is included: when *true* it means that it is a well-known property and its functionality code can be generated at a later stage; it will have the value *false* when only the skeleton code can be generated. Finally, *PolicyId* contains the name of an existing policy or the name to be assigned to the new one in the service side and *priority* allows the developer to establish a priority in the execution of the functionality of those properties which affect the same operation. These are the necessary attributes to define the main characteristics in any property, which may be complemented with specific property attributes.Once we want to use the profile in a specific case study, we will extend it with the specific properties to be used or we can have a pool of predefined properties.

- Afterwards, the specific models have to be obtained: in this case we decided our models to be EMF-complaints (http://www.eclipse.org/emf/), which facilitates a graphical edition of the element attributes within the Eclipse environment, allowing easier consultation or modification, if necessary. Service models will be based on a JAX-RPC metamodel, and three additional specific metamodels are provided for properties: an aspect-oriented one, a policy-based one and a soap tag-based one.

Subsequently, the transformation from PIM to the PSMs has to be defined. Several tools can be found for model transformations and code generation. We used *ATL* (ATLAS Transformation Language – see http://www.eclipse.org/gmt/atl/), which provides an Eclipse plugin and has its own model transformation language, also MOF-compliant. The ATL transformation file will define the correspondence between the elements in the source metamodel (PIM) and the target ones (PSMs) and will be used to generate the target model based on the defined rules and the input model. When the transformation rules are applied to the case study PIM, its platform specific models are obtained.

Finally, code can be generated from the specific models by applying additional transformation rules. In this case no target metamodel is needed since these new rules will establish correspondences from the elements in the specific metamodels to *Strings*. On the one hand, JAX-RPC web service code, to be deployed with the Java Web Service Developer Pack, will be generated from the service specific model. On the other, AspectJ will be used for the implementation of the property functionality, thus maintaining properties well modularized and decoupled from the implemented services; Java will be used to implement the code necessary for optional property inclusion. With regard to description, WS-Policy documents are obtained for each property [1], which are integrated with the aspect-oriented implementation.

III. CASE STUDY

The case study presented in this paper consists of a set of services related to a university administrative service and a web client, created for their use.

The service side consists of a set of five web services:

- PreregistrationService: using the pre-registration web service, the user will be able to create a new preregistration application for any of the courses taught in the University Centre of Mérida (CUM), to check the preregistration status and to ask for a new copy of the preregistration application to be sent to him.

Figure 1. PIM with extra-functional properties.

- RegistrationService: by using the registration service, users will be able to formalize a registration at the University Centre of Mérida, by providing their personal details, the courses to register for and payment information.
- ExamOpportunityService: through the exam opportunity service the user can obtain a list of the different subjects in a specific qualification and can bring forward or cancel any exam opportunity from the registered subjects.
- AcademicResultsService: academic results can be consulted through this service.
- TeacherService: this service can be used to obtain a list of all the CUM teaching staff in a particular area and to obtain additional information on them.

Let us imagine that we want to include some extra-functional properties to the services' model. At this stage we can discern three types of property: properties which are always applied and do not imply changes or additional information in the client code; those which are optional, so they have to be somehow chosen by the client; and those which imply changes to client code. In this sense three examples are provided, one for each option:

- First of all, a *log* property, to be applied to all operations offered by the registration service to record received invocations.
- Secondly, a property called *detailedInfo*, which will be required discretionarily by the client when invoking *bringForwardExam* in *ExamOpportunityService*: exam dates and locations can be obtained when changing the semester in which the student is going to sit the exam; the change is regularly updated and no additional information is obtained.
- Additionally, invocations to *personalData* in *RegistrationService* must be encrypted. In order to enable this functionality the *desencyption* stereotype has to be applied to the offered operation.

Regarding the client in the case study, we have created a web client for students to make use of these services by a user-friendly interface. The main web page of the web client is shown in *Figure 1*, from which the different service clients can be accessed.

IV. PLATFORM INDEPENDENT MODEL

In order to create the platform-independent model we will make use of the profile defined in the previous section and motivated in [10], which allow us to models services and their extra-functional properties in a platform-independent way.

In order to integrate the properties described in the previous section in service models we have to extend extra-functional property stereotype as shown in *Figure 2*. This figure shows us the three mentioned property stereotypes:

- *DetailedInfo* property which provides the attribute *detailedInfoFunction* in order to invoke the method which will provide us with the new functionality.
- *Log* with two attributes. *logFile* and *myLogFunction*, the first one will be the file in which we will record all the log information and the second one the will be the method which may be required for the mentioned log.
- *Desencryption* shows the attributes *keyDoc* – its value is used as the reference of the private key in the parameters decryption- and *desencryptionFunction* – it indicates the method used for the decryption.

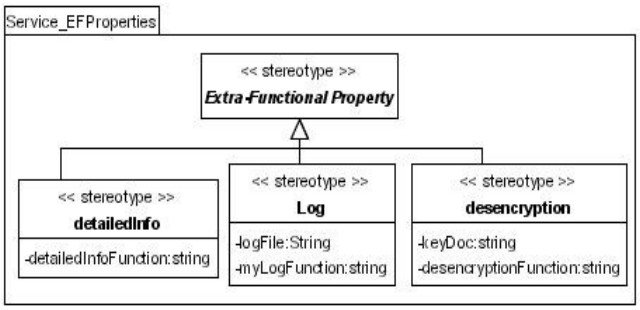

Figure 2. Extension of the extra-functional property profile with specific properties

Then we include the new property stereotypes in the service models as depicted in *Figure 3*, which are described in the following lines:

- In order to provide *bringForwardExam* in *ExamOpportunityService* with *detailedInfo* in the PIM, we have to stereotype the named operation with *<<detailedInfo>>*. Stereotype attributes are attached to models as tagged values, but they have also been included as comments in the illustration to show their values. In it the attributes for *detailedInfo* indicate that the property will be performed *optionally instead* of the execution of the named operation; it is not a *well-known* property; *policyID* is *DetailedInfo _ao4ws* and *policyDoc* is *null*.
- To provide *personalData* in *RegistrationService* with decryption in the PIM, we have to stereotype the named operation with *<<desencryption>>*. Stereotype attributes indicate that the property is not *optional* and it will be performed *instead* of the execution of the named operation – so the decryption will wrap *personalData* functionality. It is not a *well-known* property (*ack* is *true*) so that the functionality code will not be generated; *policyID* is *Desencryption _ao4ws* and *policyDoc* is http://ao4wDes.xml. Finally, for this property we can see that two specific parameters have been added: *KeyDoc* and *desencryptionFunction*, which contain the values *myPrivateKey* and *myDesencryptionFunction*, respectively and which are used as the key and function to decrypt the received message.
- Finally, log will be done for all the operations in the interface offered by *RegistrationService*. For this purpose, we have stereotyped the offered interface – *RegistrationServiceIF*– with *<<log>>* in the PIM. Stereotype attributes indicate that the application of the property will be mandatory (*optional* is *false*) and its functionality will be performed *after* the execution of the interface operations. Since *ack* has the value *true* it is a *well-known* property and therefore we will generate the complete functionality code for it. To end with, *policyID* is *log_ao4ws, policyDoc* is *null,* the method used for the logging is *myLogFunction* and the file in which the log will be recorded *myLogFile*.

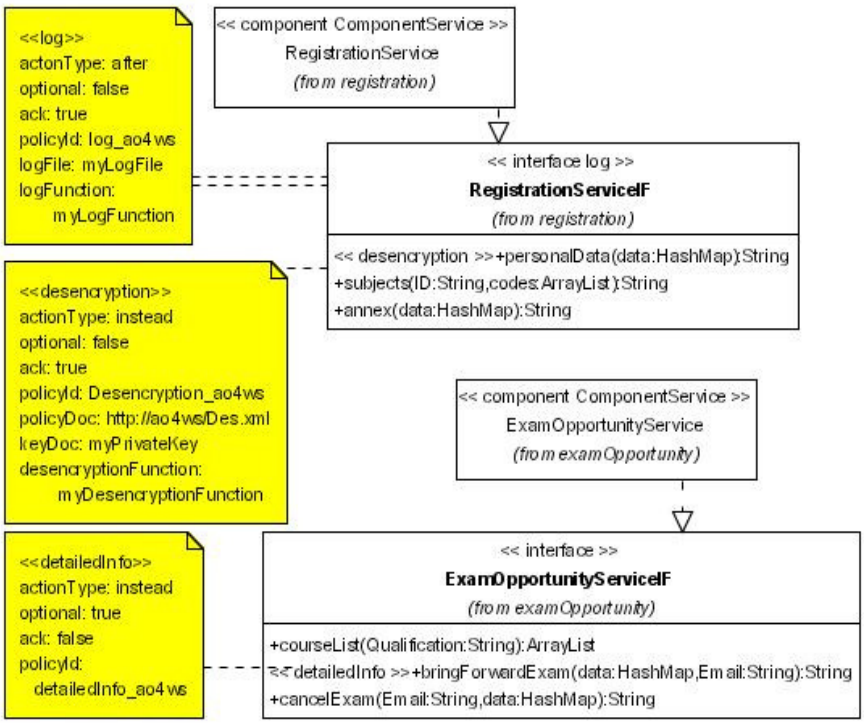

Figure 3. PIM with extra-functional properties.

It is important to remark that various stereotypes may be applied to the same operation, if necessary, thus different properties can be applied to the same element. Besides, different priorities can be assigned to those properties which are applied to the same element.

V. EXTRA-FUNCTIONAL PROPERTY PSMS

In this section we will show, first of all, the metamodels proposed for specific models, secondly the main rules used for transformations as well as the Eclipse environment configuration will be explained, and the specific models obtained for the case study will be discussed.

A. Proposed metamodels

We generate a specific model oriented to JAX-RPC services to be compiled and deployed with Java Web Service Developer Pack. In this regard the metamodel will be formed by the service Java interface and its implementation plus the necessary configuration files: *web*, *config-interface* and *jaxrpc-ri*; these elements are shown in the left-top part of *Figure 4* (properties of every element in the metamodel are not shown in the figure due to space restrictions). The metamodel, as shown in the said figure, is EMF-compliant instead of MOF-compliant, since it allows the developer to edit the generated EMF specific models to easily check and modify property attributes when necessary. The different elements shown in this part of the figure correspond to a simplified Java metamodel plus the three configuration files, which contain the main attributes necessary for their description.

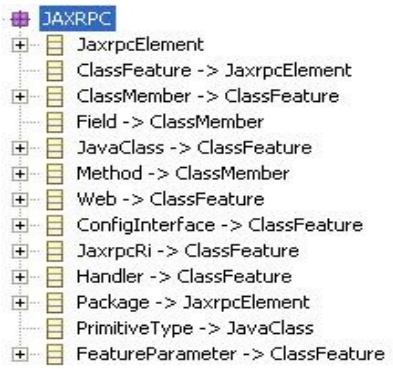

Figure 4. JAX-RPC metamodel.

As far as extra-functional properties are concerned, our specific models will be based, first of all, on an aspect-oriented approach to specify the property functionality, secondly on a *soap tags*-based approach, to lay down the necessary elements to be included or checked in the SOAP message header and, finally, a policy-based one for property description. EMF-compliant metamodels are depicted in *Figures 5, 6 and 7* and explained below:

- As shown in *Figure 5*, every *aspectClass* will have an attribute *target* which indicates the method for the property to be applied, a second attribute, *actionType,* which informs of when it has to be applied; *ack* indicates whether the property is well-known and, finally, an *action* may refer to the corresponding functionality. Besides, all additional particular property characteristics will be included as attributes. The metamodels, though represented in the EMF format, have been defined by using the KM3 syntax provided by ATL. For the better comprehension of the property-related metamodels, we have also included in this paper the KM3 definition. In this sense, in the following lines we can see the Aspect metamodel KM3 definition:

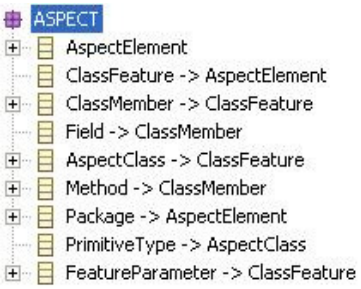

Figure 5. Aspect-based metamodel.

```
package ASPECT{

abstract class AspectElement {
     attribute name : String;       }


abstract class ClassMember extends AspectElement{
     reference type : AspectClass oppositeOf
            typedElements;
     reference owner : AspectClass oppositeOf
            members;       }

class Field extends ClassMember {
     attribute value: String;       }

class AspectClass extends AspectElement{
     reference typedElements[*] : ClassMember
            oppositeOf type;
     reference parameters[*] : FeatureParameter
            oppositeOf type;
     reference "package" : Package oppositeOf
            classes;
     reference members[*] container : ClassMember
            oppositeOf owner;
     attribute target : String;
     attribute ack: String;
     attribute actionType: String;
     attribute opt: String;
     attribute priority: String;    }

class Method extends ClassMember {
     reference parameters[*] ordered container :
            FeatureParameter oppositeOf method;}

class Package extends AspectElement {
     reference classes[*] container : AspectClass
            oppositeOf     "package";      }

class PrimitiveType extends AspectClass {}

class FeatureParameter extends AspectElement {
```

```
    reference type : AspectClass oppositeOf
            parameters;
    reference method : Method oppositeOf
            parameters;    }
}
```

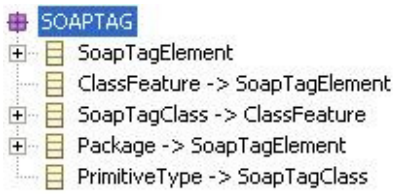

Figure 6. Soap tag-based metamodel.

- New tags are included in the SOAP Header to select –in the client side– or check –service side– relevant properties, when optional, or to deliver any other necessary information, as shown in *Figure 6*. Every *SoapTag* element will have an attribute *target* which instructs the method for the property to be applied, a second attribute, *value,* to show the tag to be included; finally, *side* indicates whether the tag is to be included by the client or checked by the service. The KM3 definition for the soap tag-based metamodel is included below:

```
package SOAPTAG {

abstract class SoapTagElement {
attribute name : String;    }


class SoapTagClass extends SoapTagElement {
    reference "package" : Package oppositeOf
            classes;
    attribute target: String;
    attribute value: String;
    attribute side: String;
    attribute providerName: String;       }

class Package extends SoapTagElement {
    reference classes[*] container : SoapTagClass
            oppositeOf "package";  }

class PrimitiveType extends SoapTagClass {}
}
```

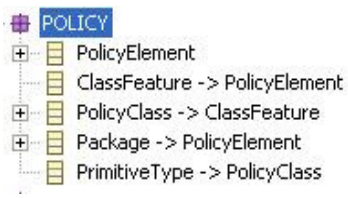

Figure 7. Policy-based metamodel.

- *Figure 7* shows that a policy will be generated for each property. The policy element will contain the policy *name*, whether the property is *optional*, well-known or domain-specific (*ack*); *targetType* indicates whether the policy is to be applied to a *portType* or an *operation* and *targetName* gives the name of the latter. For a further understanding, the KM3 description is shown in the following lines:

```
package POLICY {

abstract class PolicyElement {
attribute name : String;    }


class PolicyClass extends PolicyElement {
    reference "package" : Package oppositeOf
            classes;
    attribute opt: String;
    attribute acronym: String;
    attribute targetType: String;
    attribute targetName: String;
    attribute policyReference: String;
    attribute interface: String;
    attribute service: String;    }

class Package extends PolicyElement {
    reference classes[*] container : PolicyClass
            oppositeOf "package";  }

class PrimitiveType extends PolicyClass { }
}
```

B. *Transformation Rules*

In this section we comment briefly on one of the rules in the created transformation file in order to show how the syntax of the *ATL* declarative language is. As shown in *Figure 8* this rule applies to those properties whose *actionType* is other than *none* and which are applied to an operation. In the following lines we describe the different output results obtained in the transformation:

- The first output is an *aspectClass*; its *name* is formed by the UML package name added to the operation name and property one. Its *package* will be the one of the source element. Its *target* will be defined by the source namespace, its package name and its own name. The *actiontype* will be one in the source stereotype and its *ack* value will also be the one in the source stereotype.
- The second output –*out2*– is used for obtaining additional fields from the particular property to be included in the aspect; that is, its *name*, its *owner* and its *type* (the type will be *String* by default).
- The third output provides the aspect with the *action* and its corresponding parameters (*out3*).
- *Out4* will provide us with the *soaptag* elements to be checked to apply the property when optional. This is the reason why *type* has always the value *service* in this rule.
- Finally, policy information is found in *out5*. This information is composed of the *policy name* and *package*, the *target type* and *name*, the *ack* value and if the policy is optional or not.

The following step is to configure the Eclipse environment in order to fulfil the transformation. To start the process we will use the platform-independent model created in *Section IV* as the source for the PIM-PSM transformation. For this purpose, we have had to export the model to XMI (case tools have an

option to export to XMI). Once we have our PIM in XMI format we generate the specific models. For this purpose, as previously mentioned, we have used the Eclipse environment, in which the ATL plugin is installed. In the Eclipse environment we have created a project in which the XMI file is included. In this project the UML, JAXRPPC, ASPECT, POLICY and SOAPTAG metamodels are also present, together with the predefined set of transformation rules.

```
rule TV2AO {
from e: UML!TaggedValue (
        (e.taggedValueType() = 'actionType')  and
       (e.taggedValueDataValue()<>'none')and (
        e.modelElement.oclIsTypeOf(UML!Operation)
)       )
to out: ASPECT!AspectClass(
    name<-e.modelElement.owner.namespace.name+'_'
        +e.modelElement.name+'_'+
        e.type.owner.name,
   package <-e.modelElement.owner.namespace  ,
   target                                  <-'public
   '+e.modelElement.owner.namespace.name+ ' ' +
     e.modelElement.owner.name+'.'+
   e.modelElement.name+'(..)',
   actionType <- e.taggedValueDataValue(),
   ack<-e.getAck()     ),
out2 :distinct ASPECT!Field foreach(d in
e.getFields())(
   name <- d.type.name,
   owner <- out,
   type <- String      ),
out3 : ASPECT!Action (
   name <- 'action',
   owner <- out,
   type<-e.modelElement.parameter-
   >select(x|x.kind=#pdk_return)-
   >asSequence()first().type,
       parameters <-  e.modelElement.getP()->
        collect (p |thisModule.P2F(p))        ),
out4 :distinct SOAPTAGS!SoapTag foreach(d in
e.optional='true')(
   name <- d.type.name,
   type <- String,
                     target <-
   'public'+e.modelElement.owner.namespace.name+'
                          '+
     e.modelElement.owner.name +'.'+
   e.modelElement.name+ '(..)',
   value:<- true
   side <-service,
   package <-e.modelElement.owner.namespace  ),
out5: POLICIES!Policy(
    name<-e.modelElement.owner.namespace.name+'_'
        +e.modelElement.name+'_'+
        e.type.owner.name,
   package <-e.modelElement.owner.namespace,
   targetType<-'Operation',
   targetName <- 'public '+
   e.modelElement.owner.namespace.name       + '
   '+   e.modelElement.owner.name +'.'+
   e.modelElement.name+'(..)',
   ack<-e.getAck(),
   optional<-e.getOptional()  )             }
```

Figure 8. Transformation rules.

In order to execute the transformation we had to configure the Eclipse running environment: first of all the ATL file containing transformation rules is selected (see top part of *Figure 9*), then the source and target metamodels and source model has to be indicated, as well as the location where we desire the target generated model to be stored. An example is shown in the lower part of *Figure 9*; in it we can see the configuration for the UML2JAXRPC transformation, therefore we only have one target metamodel. When the transformations from UML to ASPECT, POLICY and SOAPTAG are configured, the three specific metamodels and output models have to be specified.

Thus, using Eclipse and the ATL plugin, we can perform PIM to PSMs transformation from the case study, whose result is shown in the next subsection.

C. Specific Models in our Case Study

Some of the specific models obtained from the case study PIM transformation are shown in *Figure 10, 11, 12 and 13*, where service and property models can be examined. Only some branches of structure have been deployed to make the illustration easier to understand. Specifically, we have chosen, for instance, the *detailedInfo* property as a characteristic example for the remainder of this paper.

Figure 10 shows the created web services with their corresponding generated elements, namely service Java interfaces and implementations and configuration files. For instance, *examOpportunity* service package is deployed in the figure, where we can see the Java interface *examOpportunityServiceIF* with its corresponding methods and parameters and its implementation. We can also see the properties corresponding to the three configuration files.

Figures 11, 12 and 13 show the property models obtained, *detailedInfo* property is explained as follows:

- An aspect, *examOpportunity_bringForwardExam_detailedInfo*, will be generated for *detailedInfo* in the service side. As we can see in *Figure 11* its attributes *target* will be the method *bringForwardExam*, for which we are aware they have two parameters –*data* and *Email*. For inspecting the remaining attribute values in the EMF environment, we would have to click on the aspect element so that the remaining values would be shown in the Eclipse property window. If we did so we would see that *actionType* has the value *instead*, and *ack false*.
- Regarding the policy element, as represented in *Figure 12*, its name will *be detailedInfo_ao4ws*. If we inspected the property window we would see that its *optional* value is *true,* for *policyAttachmen*t, *targetType* is *operation* and *targetName bringForwardExam*.
- Due to its optional nature, we ought to include code whose function is to check whether *detailedInfo* has been selected: the corresponding *SOAPTag target* will be *bringForwardExam*, its value *detailedInfo* and it will operate as a *side service* – depicted in *Figure 13*.

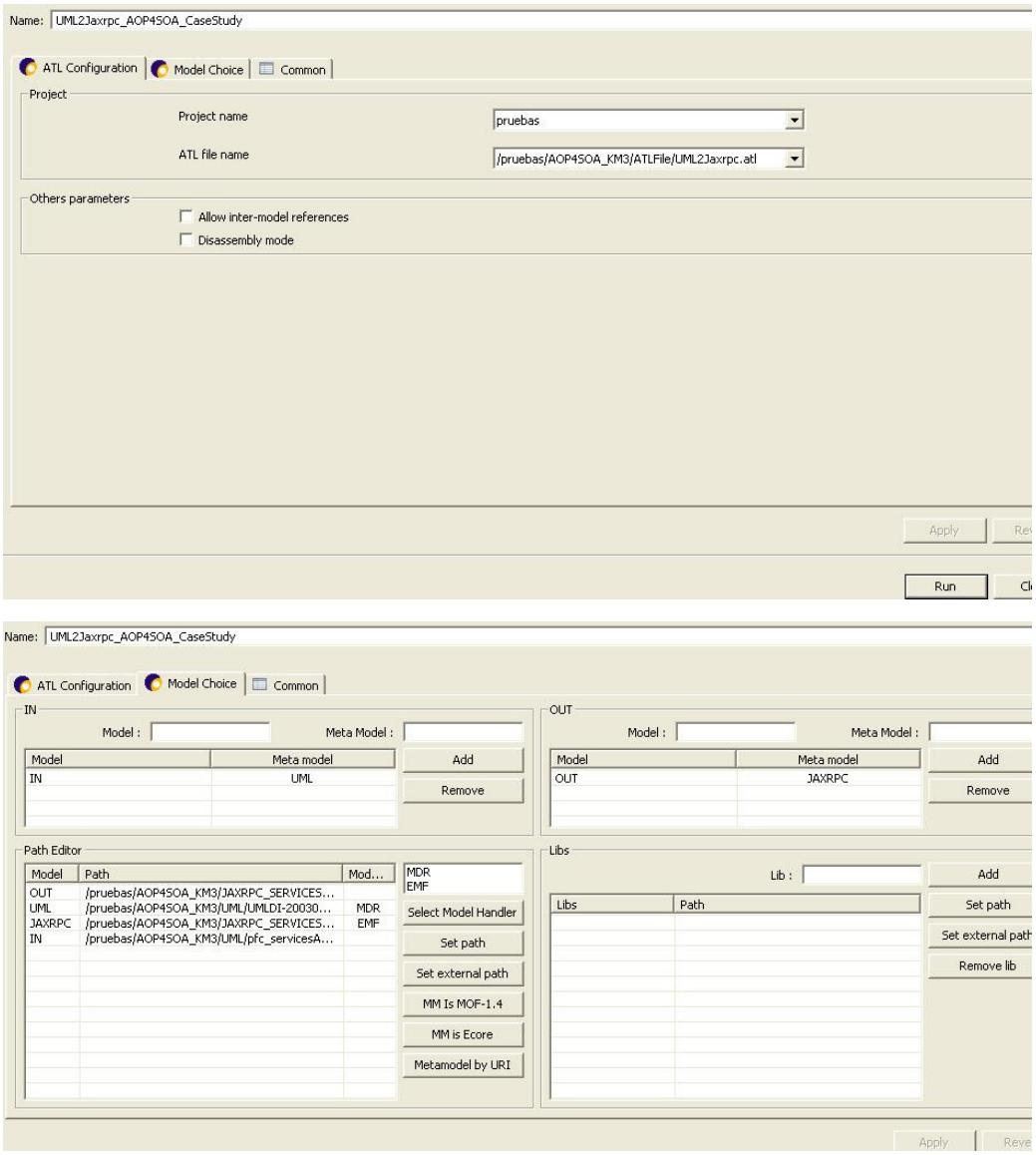

Figure 9. Eclipse configuration for model transformations.

VI. CODE LAYER: EXTRA-FUNCTIONAL PROPERTY GENERATED CODE

Once we have our models for the case study we may also apply additional rules to generate code from them. For this purpose we will make use of the platform-specific models obtained in previous section as the source for the PSM-code transformation. The developer may have modified any attribute value should they be necessary in the eclipse environment.

In order to generate the code, we again use the Eclipse environment. We have created a project in which the Ecore files obtained in previous subsections are included. In this project the UML, JAXRPPC, ASPECT, POLICY and SOAPTAG metamodels are also present, together with the set of transformation rules corresponding to this stage of development.

In order to execute the transformation we have to configure the running environment. First of all the ATL file invoking the transformation rules is selected, then the source metamodels and models are indicated, as well as the location of the libraries with the complete set of transformation rules, as shown in *Figure 14*. In this figure the ALL2STRING transformation is chosen, in this sense we have four source metamodels, JAXRPC, ASPECT, POLICY and SOAPTAG, and their respective models. We have also included six separate libraries with transformation rules: one library is included for each type of metamodel, one more in order to maintain the code generation for the compilation and deployment files separate from the implementation and description code, and one more in order to generate the full code of well-known properties.

From the service specific model, where additional attribute values can be added or modified (e.g. *deployment endpoint*),

Figure 10. Jax-rpc PSM model.

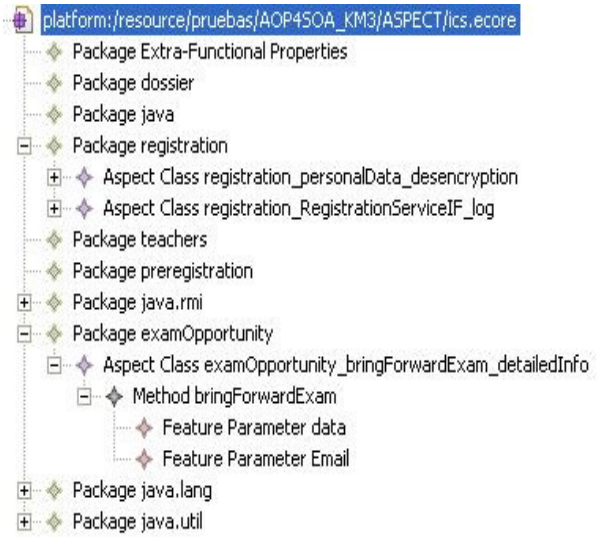

Figure 11. Aspect-based PSM model.

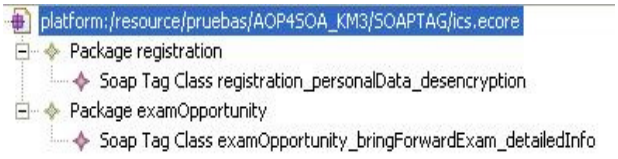

Figure 12. Soap tag-based PSM model.

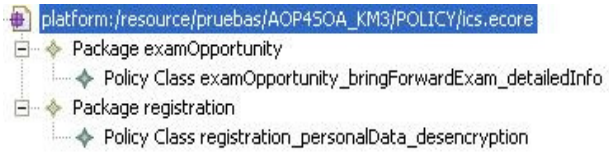

Figure 13. Policy-based PSM model.

the JAX-RPC service skeleton code for JWSDP compilation and deployment is generated. In this sense, the Java interface and implementation skeleton will be generated and complete configuration files created. *Figure 15* shows the Java interface generated for *examOpportunityService*.

From the property models, transformation rules will generate skeleton code for the three extra-functional property model elements: *Figure 15* shows code generated for *detailedInfo*. However, in the case of well-known or user-defined properties, a repository with specific code may be maintained to generate additional code for the three of them. In these cases, in which *ack* is *true*, it is possible to generate the advice functionality and further policy content.

Regarding property implementation, Java code will be generated to check if soap tags are included in the SOAP message and AspectJ has been chosen for the implementation of the property's functionality, consequently properties remaining well modularized and decoupled from implemented applications, as demonstrated in [11]. An AspectJ aspect will be generated for each aspect class in our model. AspectJ pointcuts will be determined by target element's execution. Concerning the advice, depending on the *actionType* attribute value, *before*, *after* or *instead*, the advice type will be *before*, *after* or *around*, respectively; its name will be the one in the *action* attribute. With regard to property description, it is proposed to generate the WS-Policy documents for each property, integrated with the aspect-oriented generated properties as explained in [12]. In this sense, an xml file based on the WS-Policy and WS-PolicyAttachment standards is generated. The policy is attached to the stereotyped element in the PIM, represented by the attribute *targetName* in the policy specific model.

VII. Related Work

As regards Web Service modeling proposals, such as [16] and [4], it can be noted that most of the literature in this area tries to find an appropriate way to model service compositions with UML. The research presented by J. Bezivin *et al.* [4] is worth a special mention; in it Web Service modeling is covered in different levels, using *Java* and *JWSDP* implementations in the end. It is also worth mentioning the paper from M. Smith *et al.* [15], where a model-driven development is proposed for grid applications based on the use of Web Services. Our work differs from these in the sense that ours provides the possibility of adding extra-functional properties to the services and is not oriented to the service modeling itself; therefore it could be considered as complementary to them. We can mention the ASG (*Adaptive Services Grid*) project, which takes into consideration some specific extra-functional properties in their WSDL-centric model-driven development [14]; however, services and properties have to be initially described by a semantic language, and, being a WSDL-centric approach from the very beginning, the possibilities of implementation for the services described are limited.

Concerning proposals which focus on extra-functional properties, we can especially mention two. To begin with, WSMF from D. Fensel *et al.* [6], where extra-functional properties are modeled as pre and post conditions in an ontology description. Secondly, L. Baresi *et al.* extend WS-

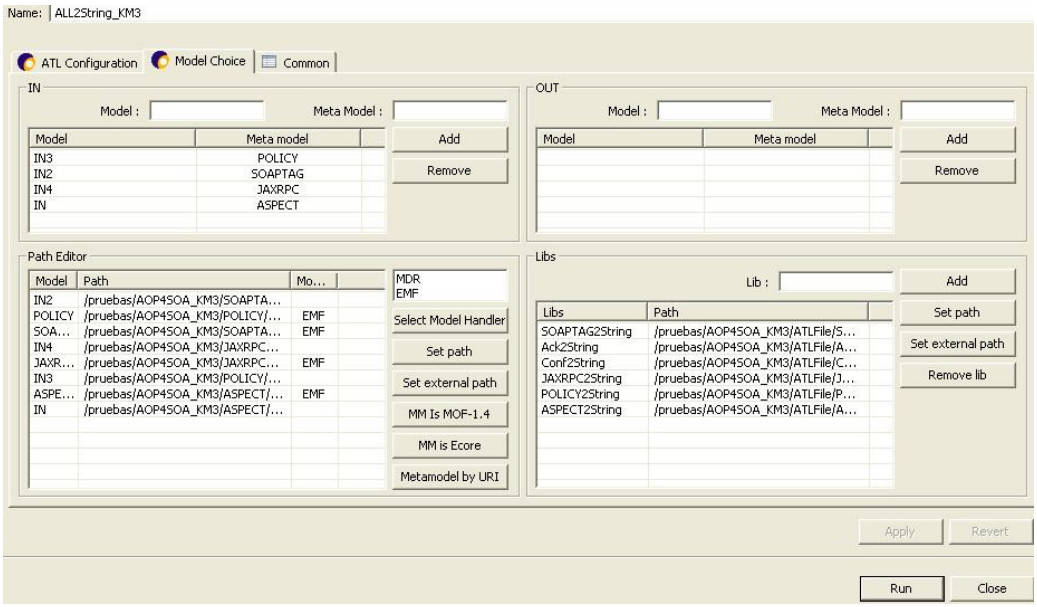

Figure 14. Eclipse configuration for code generation.

Policy by using a domain-independent assertion language in order to embed monitoring directives into policies [2]. Both are interesting proposals, however they do not follow the UML standard, which we consider essential for integrating properties in future service models.

Many policy-related contributions are emerging as policies are a very popular issue nowadays. Among them we can especially remark the contribution from T. Gleason *et al.*, which provides very interesting discussion on policy management [7]. On the other hand, plenty of literature can be found on model-driven development for web service compositions (for instance [5]); our proposal aims at providing support for extra-functional properties in isolated or composed services, and therefore could be complementary to the named proposals.

VIII. Discussion and Conclusion

This paper has shown a case study in which a model-driven approach to web service and their extra-functional properties development has been followed. Several issues arise for discussion:

- First of all, it is important to remind that the profiles provided for PIM level are motivated and further discussed in [10], as previously said. We wish to mention that both profiles attempt to keep the modularization and decouplessness of the different level elements in our model from this initial stage of development.
- Secondly, four different specific metamodels are used at PSM stage in order to maintain the service development independent from the property one on the one hand (Jax-rpc metamodel), and, on the other, in order to keep the properties decoupled from the implementation (aspect metamodel) and description (policy metamodel) perspectives. Besides, more versatile services are provided when the extra functionality is optional for the client, thus it ought to be possible for properties to be selected somehow in a transparent way for the service (soap tag metamodel). This last case is especially suitable for domain specific properties.
- Concerning the generated code, AspectJ has been used for the implementation of the property functionality, thus maintaining properties well modularized and decoupled from the services implemented as demonstrated in [14], where additional elements are also necessary for optional property inclusion. With regard to description, it is proposed to generate the WS-Policy [1] documents for each property, which are integrated with the aspect-oriented generated properties as explained in [15]. This allows properties to remain decoupled not only at modeling stage, but also during implementation. Besides, SOAP Tags will be used to select optional properties and transfer the additional data necessary due to the property inclusions in a transparent way.
- Moreover, having service and property metamodels separated, model-driven transformations remain simpler, but still complementary.
- Besides, traceability is maintained from the very independent model to code, so properties are easily eliminated or added from any stage of development at any level of abstraction, without damaging the remainder of the system.
- Finally, regarding performance, it is important to mention that no payload is included due to the aspect-oriented implementation as it is a static approach.

```
**********EXAMOPPORTUNITY SERVICE INTERFACE*******
package examOpportunity;
public interface ExamOpportunityServiceIF extends
Remote {
    public ArrayList courseList (String
        Qualification) throws RemoteException;
    public String bringForwardExam(HashMap data,
        String EMail) throws RemoteException;
    public String cancelExam(HashMap data, String
        Email) throws RemoteException;}
```

```
***************DETAILED INFO  ASPECT**************
public aspect
pportunityExam_bringForwardExam_detailedInfo {
pointcut bringForwardExam_detailedInfoP
   (data: hashMap, Email:String): execution
   (public     *.opportunityExam_bringForwardExam
   (HashMap, String)) && args(data, Email);

String around ((data: hashMap, Email:String):
bringForwardExam_detailedInfoP (data, Email){
  if ((((String)opportunityHandlerHandler.
  operDetailedInfo.get("operationName")).compareTo
  ("bringForwardExam") ==0) &&(((String)
  opportunityHandler. operdetailedInfo.get
 ("propertyName")).compareTo("detailedInfo")==0))
   { [...]
       [functionality to be completed]    [...]}
    else result=proceed(data, Email)        […]}
```

```
***************** DETAILED INFO POLICY***********
<wsp:PolicyAttachment >
<wsp:AppliesTo>[…]
<wsp:Operation  Name= bringForwardExam/>[…]
</wsp:AppliesTo>
<wsp:Policy name=detailedInfo_ao4ws [...] ">
        <[to be completed]/>
</wsp:Policy></wsp:PolicyAttachment>
```

```
***************SERVICE SIDE SOAP CODE************
if element.getElementName().getLocalName().
 equals ("operationName")){
  String operationName = element.getValue();
  operdDetailedInfo.put("opName",operationName);
  Iterator iter2= element.getAllAttributes() ;[…]
  if (name.getLocalName().equals("propertyName")){
    String propertyName=
        Element.getAttributeValue(name);
    operDetailedInfo.put("propertyName",
        propertyName); }  }
```

Figure 15. Generated Code.

In regard with our present and future work, we are extending our approach in order to cover Quality of Service monitoring. Further information on this topic can be found in [8].

REFERENCES

[1] Bajaj, S., Box, D., Chappeli, D., et al.. Web Services Policy Framework (WS-Policy), ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf, September 2004

[2] Baresi, L. Guinea, S. Plebani, P. WS-Policy for Service Monitoring. VLDB Workshop on Technologies for E-Services, Trondheim, Norway, September 2005

[3] Beisiegel, M., Blohm, H., Booz, D.,et al. Service Component Architecture. Building Systems using a Service Oriented Architecture. http://download.boulder.ibm.com/ibmdl/ pub/software/dw/specs/ws-sca/SCA_White_Paper1_09.pdf, November 2005

[4] Bézivin, J., Hammoudi, S., Lopes, D. et al. An Experiment in Mapping Web Services to Implementation Platforms. N. R. I. o. Computers**:** 26, 2004

[5] Castro, V. Marcos, E. Lopez, M. A model driven method for service composition modelling: a case study, Int. Journal in Web Engineering and Technology, V. 2, I. 4, 2006.

[6] Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. http://informatik.uibk.ac.at/users/c70385/ wese/wsmf.bis2002.pdf

[7] Gleason, T., Minder, K., Pavlik, G. Policy Management and Web Services, Proc. Policy Management for the Web Workshop at IWWW Conf., Chiba, Japan, May 2005.

[8] Ortiz G., Bordbar B. Model-driven Quality of Service for Web Services: an Aspect-Oriented Approach. Proc. Int. Conf. on Web Services, Beijing, China, September 2008.

[9] Ortiz G., Hernandez J. A Case Study on Integrating Extra-Functional Properties in Web Service Model-Driven Development. Proceedings International Conference on Internet and Web Applications and Services, 2007. Digital Identifier: 10.1109/ICIW.2007.2

[10] Ortiz G., Hernández J., Toward UML Profiles for Web Services and their Extra-Functional Properties, Proc. Int. Conf. on Web Services, Chicago, EEUU, September 2006.

[11] Ortiz G., Hernández J., Clemente, P.J. How to Deal with Non-functional Properties in Web Service Development, Proc. Int. Conf. on Web Engineering, Sydney, Australia, July 2005

[12] Ortiz, G., Leymann, F. Combining WS-Policy and Aspect-Oriented Programming. Proc. of the Int. Conference on Internet and Web Applications and Services, Guadeloupe, French Caribbean, February 2006

[13] Papazoglou, M. Van Den Heuvel, W. Service-oriented design and development methodology, International Journal in Web Engineering and Technology, V.2, Issue 4, 2006.

[14] Roman, D et al. Requirements Analysis on the ASG Service Specification Language. Deliverable D1.1-1, DERI Innsbruck, 2005.

[15] Smith, M., Friese, T. Freisbelen, B. Model Driven Development of Service-Oriented Grid Applications. Proc. of the Int. Conference on Internet and Web Applications and Services, Guadeloupe, French Caribbean, February 2006

[16] Thöne, S. Depke, R, Engels, G.. Process-Oriented, Flexible Composition of Web Services with UML. Int. Workshop on Conceptual Modeling Approaches for e-business: A Web Service Perspective, Tampere, Finland, 2002

[17] Weerawarana, S. Curbera, F. Leymann, F., et al. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Ed. Prentice Hall, ISBN 0-13-148874-0, March 2005

This work has been developed thanks to the support of MEC under contract TIN2005-09405-C02-02.

Real-time Network Traffic Management using the Modified BPTraSha Algorithm

Karim Mohammed Rezaul
Centre for Applied Internet Research (CAIR)
Glyndwr University, Wrexham, Wales, UK
karim@cair-uk.org

Vic Grout
Centre for Applied Internet Research (CAIR)
Glyndwr University, Wrexham, Wales, UK
v.grout@glyndwr.ac.uk

Abstract- **Various researchers have reported that traffic measurements demonstrate considerable burstiness on several time scales, with properties of self-similarity. Also, the rapid development of technologies has widened the scope of network and Internet applications and, in turn, increased traffic. The self-similar nature of this data traffic may exhibit spikiness and burstiness on large scales with such behaviour being caused by strong dependence characteristics in data: that is, large values tend to come in clusters and clusters of clusters and so on. Several studies have shown that TCP, the dominant network (Internet) transport protocol, contributes to the propagation of self-similarity. Bursty traffic can affect the Quality of Service of all traffic on the network by introducing inconsistent latency. It is easier to manage the workloads under less bursty (i.e. smoother) conditions. This paper continues the work published in [1], which introduced a novel algorithm for traffic shaping to smooth out the traffic burstiness. It was named as the Bursty Packet Traffic Shaper (BPTraSha). Experimental results show that this approach allows significant traffic control by smoothing the incoming traffic. BPTraSha can be implemented on the distribution router buffer so that the traffic's bursty nature can be modified before it is transmitted over the core network (e.g., Internet). A modified BPTraSha algorithm is proposed in this research, which can be shown to be more dynamic, and therefore responsive, than the previous one. In this case, the dynamic variation of link speed can lead to further reducing the long-range dependence of network traffic.**

Keywords:
Self-similarity, LRD, ACF, QoS, Shaping, BPTraSha.

I. INTRODUCTION

A number of factors, such as a slow start phase of the congestion window, packet losses, ack-compression of TCP traffic and multiplexing of packets at the bottleneck rate, can cause either short- or long-term burstiness in the behaviour of TCP flow [2]. The research in [3] investigates how various versions of TCP congestion control affect network performance when traffic is bursty. It shows a significant adverse impact on network performance attributable to traffic self-similarity and, while throughput declines gradually as self-similarity increases, queueing delay increases more drastically. Self-similarity is closely related to the phenomenon of heavy-tailed distributions, where the tail index of the distribution declines as a power law with small index (less than 2). TCP represents the dominant transport protocol of the network (e.g., Internet), which contributes to the propagation of self-similarity [4]. It was shown in [4] that TCP itself inherits self-similarity when it is combined with self-similar background traffic in a bottleneck buffer through the transform function of the linear system.

The research in [5] investigated the relationship between TCP's congestion control mechanism and traffic self-similarity under certain network conditions. It demonstrates that, when a TCP connection is going through a highly-lossy channel - and the loss condition is not affected by this single TCP connection's behaviour, TCP starts to produce packet trains that show pseudo-self-similarity [6] (i.e. traffic is self-similar over limited range of time scales). In fact, when the loss rate is relatively high, TCP's adaptive congestion control mechanism generates traffic with heavy-tailed off or idle periods (i.e. inter-arrival time), which in turn introduces long-range dependence into the overall traffic. The researchers in [7] analysed the traces of actual TCP transfers over the Internet and reported that individual TCP flows, isolated from the aggregated flow on the link, also have a self-similar nature. Also, the loss rate experienced by TCP flow is an important indicator of the degree of self-similarity in the network traffic. A natural construction of the extremely bursty nature of TCP traffic comes from timeouts (representing 'silent' periods) that lead to losses and, consequently, losses increase the burstiness - and higher loss rates thus lead to a higher degree of self-similarity (i.e. higher values of Hurst parameter) [7]. It has been shown [8] that if packets were to arrive according to the well-behaved Poisson process, simple retransmission mechanisms can make traffic appear self-similar over time scales and be a possible source of long-range dependence. Retransmission mechanisms can make a network congestible, because these mechanisms often cause network inefficiencies which cause throughput to degrade specifically in periods when load is already high.

One of the major drawbacks of TCP/IP is the lack of true Quality of Service (QoS) functionality. QoS in networks, in simple terms, is the ability to guarantee and limit bandwidth appropriately for certain services and users. Traffic shaping is the term used for any system by which traffic is constrained to a specific speed. Traffic shaping is an attempt to control network traffic in order to optimize, attempt to optimize or guarantee performance, low-latency and bandwidth. Traffic shaping deals with concepts of classification, queue disciplines, enforcing policies, congestion management, QoS and fairness. Shaping is the mechanism by which packets are delayed before transmission in an output queue to meet a desired output rate. This is one of the most common requirements of users seeking bandwidth control solutions. The basic principle of traffic shaping is based on the fact that the outgoing traffic from the FireBrick or router is scheduled. The FireBrick is a network appliance with a rich feature set, including a stateful firewall, router, managed switch, traffic shaping, tunneling, multilink handling, and much more.

Each packet has a time stamp, stating when it is to be sent, and all traffic is normally sent in order and not before its time. This method is used to deliberately slow responses

from reject and bounce filters, as well as for speed lanes. When sending a packet, its length is considered and the transmission time added to the time for the next packet to be sent. This ensures packets can only leave at the designated rate and no faster. Shapers can smooth out bursty traffic and attempt to limit or ration traffic to meet, but not exceed, a configured rate (e.g. packets per second or bits/bytes per second). However earlier research [9, 10] reports that the strong robustness of self-similarity properties existing in traffic cannot be removed by shaping.

This paper is organised as follows. Section II highlights research related to shaping traffic. Section III describes the definitions of self-similarity, long-range dependence and the autocorrelation function. Section IV introduces the algorithm BPTraSha and its purpose. Section V discusses the performance and complexity of BPTraSha by experimental analysis. Finally we draw conclusions and suggest future work in section VI.

II. Related research

Several researchers show how to control the network in situations where the distribution tail of the traffic flow process cannot be altered. In [11] it is claimed that, by incorporating shapers and policers at the edges of the networks, huge buffers are needed that result in large delays and may thus be unacceptable in practice. In [12] a Burst Shaping Queueing (BSQ) algorithm is presented, which can minimize the burstiness of traffic on packet switched routers by interleaving packets that are going to follow different links on next hops. The research in [13] discusses issues of shaping and simulated queueing performance of ATM traffic. In this work, a leaky bucket shaping method is used and the shaping effect surprisingly results in higher values for the estimated Hurst parameter (the degree of self-similarity) - that is, the estimated Hurst parameter is increased due to shaping. It is also noted that the interpretation of the estimated Hurst parameter is problematic in practice.

In [14] an optical packet assembly mechanism is proposed to function as a traffic shaper and its impact on self-similar traffic characteristics at the edge router are investigated. Simulation results demonstrate that the optical packet assembly mechanism can reduce traffic correlation and the degree of self-similarity. In [15], the three different traffic shaping techniques are presented: thinning, striping and shuffling, which can improve the queueing characteristics of data by decreasing the short-term burstiness and diminishing short-term correlations. However, none of these processes are shown to decrease the degree of Long-Range Dependence (LRD) in data. The research in [16] proposes a dual leaky bucket technique for shaping the web traffic, reducing the intensity of the long duration traffic bursts, which, in turn, reduces the Hurst parameter. The 'leaky bucket' procedure [17] is also employed in [18] to examine the effectiveness of shaping in the case of α-stable fractal traffic and it is found that shaping and policing mechanisms do not eliminate self-similarity.

III. Self-similarity, Long-range dependence and Autocorrelation Function

It is especially important to understand the link between self-similarity and long-range dependence of network traffic and performance of the networks because such characterization can be potentially applied for control purposes such as traffic shaping, load balancing, etc. In general two or more objects having the same characteristics are called self-similarity. A phenomenon that is self-similar looks the same or behaves the same when viewed at different degrees of magnification or different scales on a dimension and bursty over all time scales. Self-similarity is the property of a series of data points to retain a pattern or appearance regardless of the level of granularity used and is the result of long-range dependence in the data series. If a self-similar process is bursty at a wide range of timescales, it may exhibit long-range- dependence. In general lagged autocorrelations are used in time series analysis for empirical stationary tests. Self-similarity manifests itself as long-range dependence (i.e., long memory) in the time series of arrivals. The evidence of very slow, linear decay in the sample lag autocorrelation function (ACF) indicates the nonstationary behaviour [19]. Long-range-dependence means that all the values at any time are correlated in a positive and non-negligible way with values at all future instants. For a continuous time process $Y = \{Y(t), t \geq 0\}$ is self-similar if it satisfies the following condition [20]:

$$Y(t) \overset{d}{=} a^{-H} Y(at), \quad \forall a > 0, \quad and \quad 0 < H < 1 \qquad (3.1)$$

where H is the index of self-similarity, called Hurst parameter and the equality is in the sense of finite-dimensional distributions.

The stationary process X is said to be a long-range dependent process if its autocorrelation function (ACF) is non-summable [21] meaning that

$$\sum_{k=-\infty}^{\infty} \rho_k = \infty \qquad (3.2)$$

The details of how ACF decays with k are of interest because the behaviour of the tail of ACF completely determines its summability. According to [22], X is said to exhibit long-range dependence if

$$\rho_k \sim L(t) k^{-(2-2H)}, \ as \ k \to \infty \qquad (3.3)$$

where $\frac{1}{2} < H < 1$ and $L(.)$ slowly varies at infinity, i.e.,

$$\lim_{t \to \infty} \frac{L(xt)}{L(t)} = 1, \quad \text{for all } x > 0 \qquad (3.4)$$

Equation (3.3) implies that the LRD is characterized by an autocorrelation function that decays hyperbolically rather than exponentially fast.

LRD processes are characterised by a slowly decaying covariance function that is no more summable. When the network performance is affected by LRD the data are correlated over an unlimited range of time lags and this property results in a scale invariance phenomenon. Then no characteristic time scale can be identified in the process, they are all equivalent for describing its statistics, i.e., the part resembles the whole and vice versa. This is why LRD is also called Self-Similarity [23].

IV. BPTraSha: an algorithm for controlling bursty traffic

Let us assume that the client networks (such as C_1, C_2, C_3,….., C_n) are connected to the main router of Internet service provider (ISP). The packet sequences (i.e. packet size in byte) from different sources are queued at the router

buffer. The packet sequences arrive at the router buffer with timestamp in second (or millisecond). Therefore we have packet size in byte for corresponding timestamp. For the experimental analysis we used Lawrence Berkeley Laboratory (LBL) TCP data which are publicly available in [24]. The bursty nature of packet sequences arrive at the router will be shaped with the fixed rate by the shaper algorithm BPTraSha. Here we mean the link speed as desired fixed rate (i.e. capacity, C) at which the packets would be transmitted. In other words, a bursty traffic in the input will be regulated with the fixed rate before they pass through the network. The algorithm is described in Figure 1. For the user's convenience, the algorithm is implemented both in Java and Matlab programming language.

```
T = timestamp
B = Packet size in bytes
TT = transmission time
bps = Bit per second
Delt = Delay in second
Tmod = Modified time
Tmod_cng = change in modified time
bps_mod = Modified bit per second
Ld = Longest delay
Sd = Shortest delay
S = sample count (e.g. number of packet sequences)
C = link speed

1. Capture B for corresponding T (i.e. T and B)
2. Count S
3. For k = 0 to (S-1)
   a)  if (k = 0)
         bps[k] = B[k] *8 / (T[k]+TT[k])
           where TT[k] = B[k] *8 / C
       else bps[k] = B[k]*8 / (T[k]-T[k-1]+TT[k-1])
   b)  if (k = 0)
         Delt[k] = 0
         Tmod[k] = T[k]
       else
           i)  Delt[k] = T[k]-(Tmod[k-1]+TT[k-1])
          ii)  if (Delt[k] >= 0)
                 Tmod[k] = T[k]
               else
                 Tmod[k] = T[k]-Delt[k]
4. For k = 0 to (S-2)
  i)  if (k = 0)
        Tmod_cng[k] = Tmod[k]
      else
        Tmod_cng[k] = Tmod[k+1]-Tmod[k]
  ii)  set  bps_mod[k] = B[k]*8 / Tmod_cng[k]
  iii)  if (Delt[k] <0)
        find out Ld     // Longest delay
        find out Sd     // Shortest delay
5. Exit
```

Fig. 1. The algorithm, BPTraSha

The performance of algorithm has been depicted in Figure 2 to Figure 10. The Figures show how the bursty nature of traffic are smoothed out by the algorithm, i.e., bursty traffic have been shaped by the desired fixed rate. The length of packet sequences used for these experiments is N =65536. We used various types of TCP data for the experiment, but due to space limitations we provide here results from using LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet data. The link capacity (i.e. desired rate) applied here are C = 5 Mbps, C = 10 Mbps and C =15 Mbps. Figure 11 illustrates the expected longest delay observed for different link speed with the variation of length of packet sequences. It is clear from the Figure that higher capacity yields less delay and thereby provides better quality of service. Figure 12 shows the expected shortest and longest delay for different link speed while length of sequences is varied. The shortest delay found to be from 0.000001 second to 0.000004 second. The longest delay here observed to be from 0.00056 second to 0.15927 second depending on the link speed (C) and length (N) of packet sequences. The higher the link speed the shorter the observed delay. Table 1 illustrates a sample of trace files that the BPTraSha algorithm uses.

TABLE 1: SAMPLE OF A TRACE FILE

Length of samples	Timestamp (T_i)	Packet size in byte (B_i)
1	0.008185	41
2	0.010445	42
3	0.023775	42
4	0.026558	41
5	0.029002	82
6	0.032439	55
7	0.049618	41
8	0.052431	42
9	0.056457	42
10	0.057815	454
11	0.072126	40
12	0.098415	95
13	0.104465	55
14	0.122345	40
15	0.12449	40
16	0.125228	41
17	0.138935	41
18	0.13995	104
19	0.14093	41
20	0.146912	72
:	:	:
N	T_n	B_n

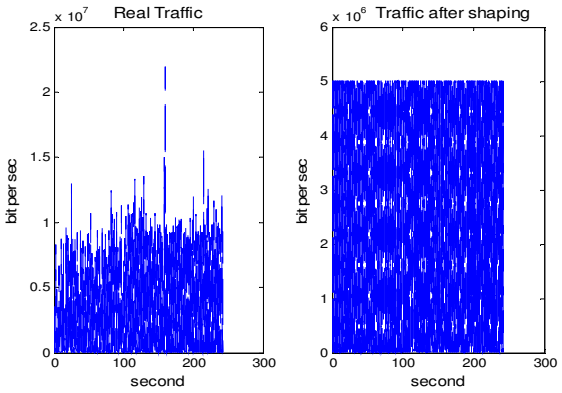

Fig. 2. LBL-tcp3-pkt, C = 5 Mbps, N = 65536

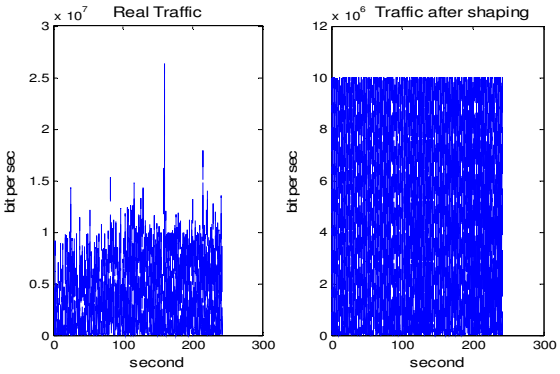

Fig. 3. LBL-tcp3-pkt, C = 10 Mbps, N = 65536

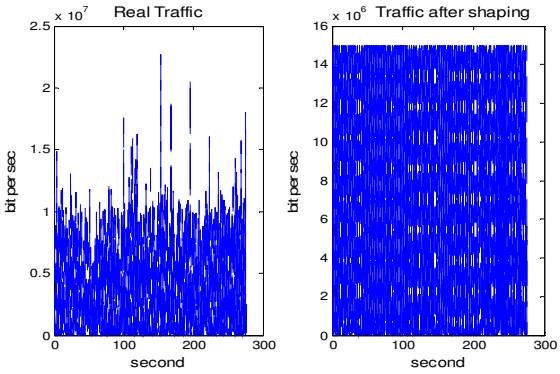

Fig. 7. LBL-pkt-4_tcp, C = 15 Mbps, N = 65536

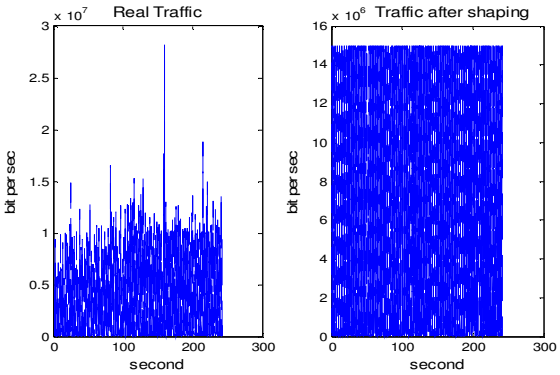

Fig. 4. LBL-tcp3-pkt, C = 15 Mbps, N = 65536

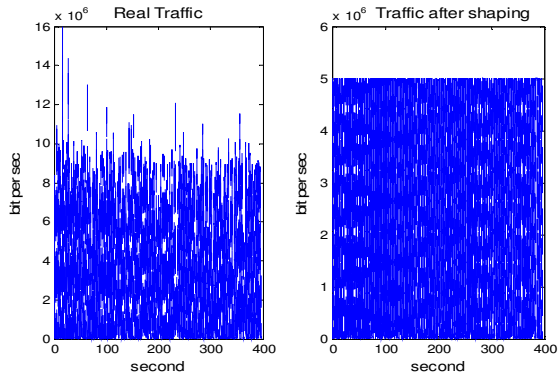

Fig. 8. LBL-pkt-5_tcp, C = 5 Mbps, N = 65536

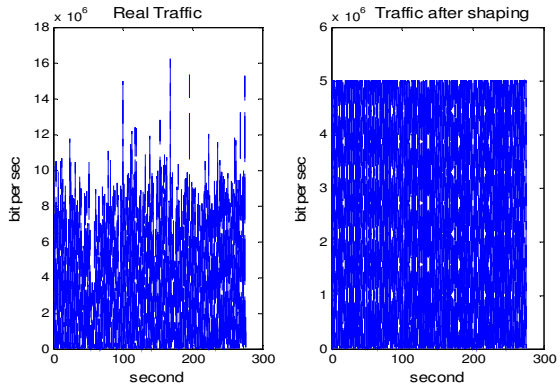

Fig. 5. LBL-pkt-4_tcp, C = 5 Mbps, N = 65536

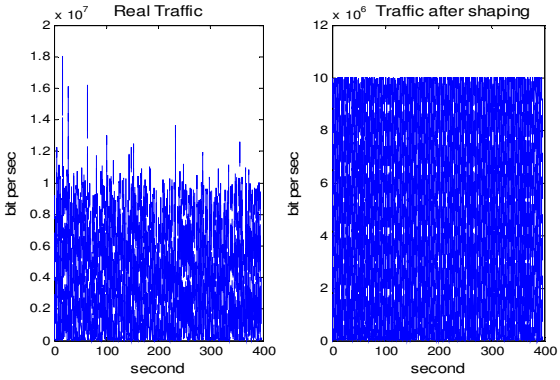

Fig. 9. LBL-pkt-5_tcp, C = 10 Mbps, N = 65536

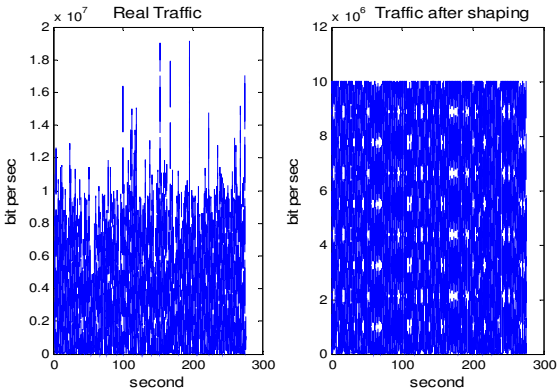

Fig. 6 . LBL-pkt-4_tcp, C = 10 Mbps, N = 65536

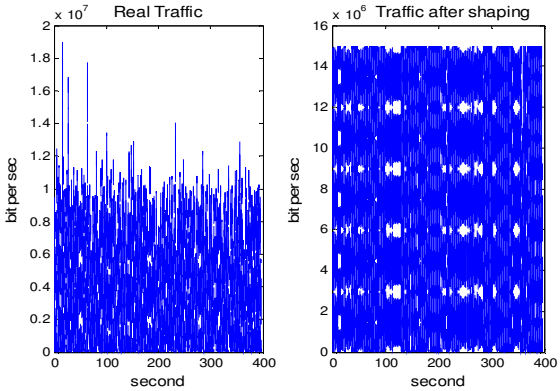

Fig. 10. LBL-pkt-5_tcp, C = 15 Mbps, N = 65536

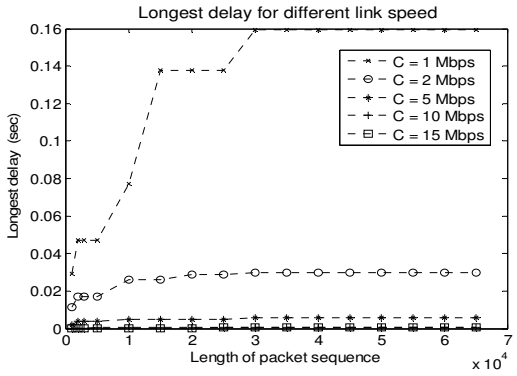

Fig. 11. Performance of BPTrasha algorithm: Observation of longest delay. Variation of link speed with different length of packet sequences.

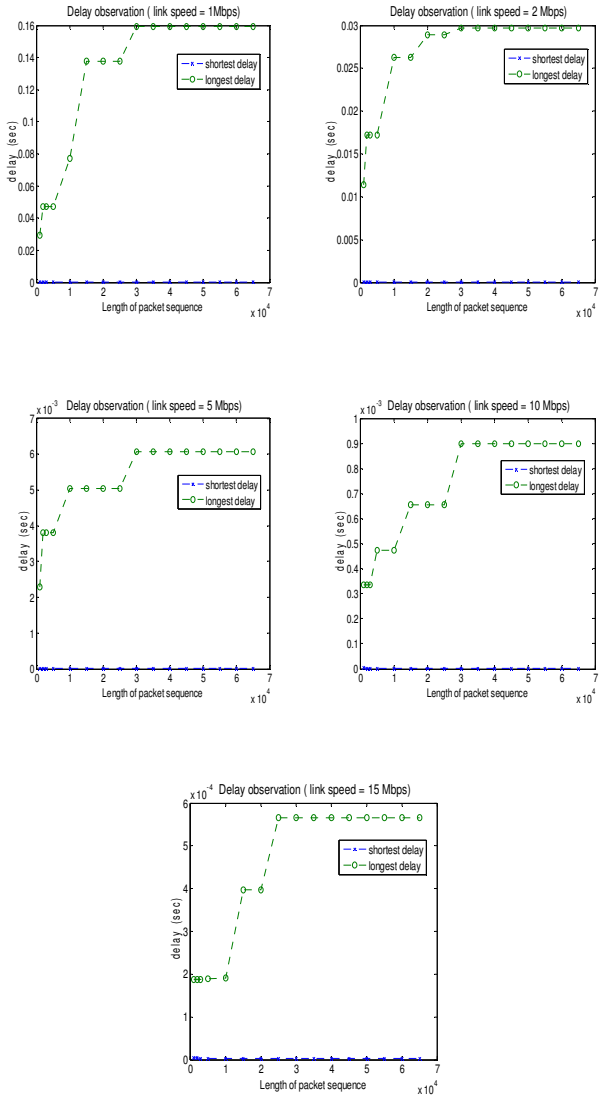

Fig. 12. Performance of BPTrasha algorithm: Observation of shortest and longest delay, for different length of packet sequences.

V. COMPLEXITY OF THE ALGORITHM, BPTRASHA

To explore the complexity of BPTraSha we chose six workstations with different specifications, which are represented in Table II. We investigated several lengths of packet sequences such as N = 1000, N = 2000, N = 3000, N = 5000, N = 10000, N = 15000, N = 20000, N = 25000, N = 30000, N = 35000, N = 40000, N = 45000, N = 50000, N = 55000, N = 60000 and N = 65000. In our research we mainly emphasise the time (as opposed to space) complexity of the algorithm.

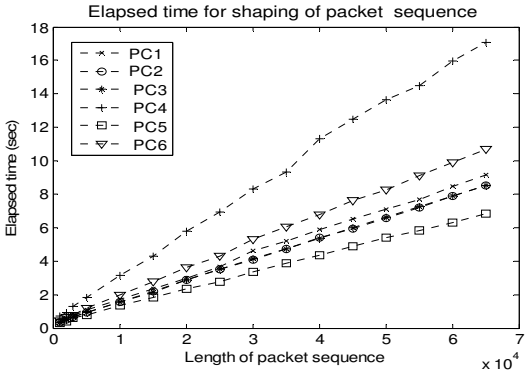

Fig. 13. Observation of elapsed time for different length of packet sequences with different PC's

Figure 13 depicts the observation of elapsed (execute) time for different lengths of packet sequences with different PCs. It is obvious that PC5 yields better performance as it possesses higher specifications. Figure 14 shows a percentage of affected packets due to delay for different lengths of packet sequences. Here higher capacity (C) signifies better performance due to less affected packets. However, the elapsed time for executing the algorithm does not significantly vary for different link speeds with the variation in lengths of packet sequences, a feature that can be observed in Figure 15.

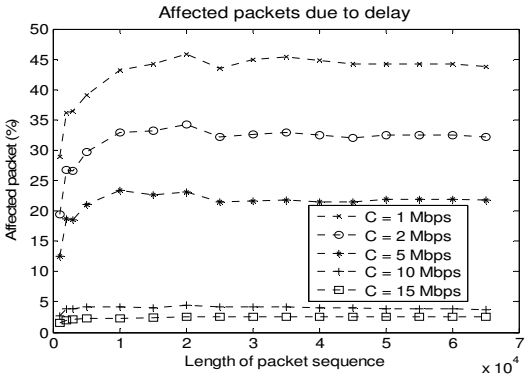

Fig. 14. Affected packets due to delay for different length of packet sequences

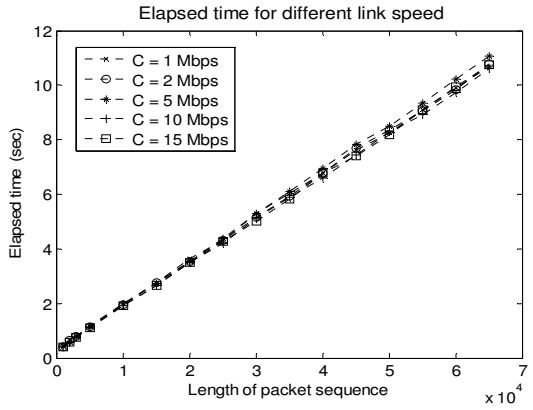

Fig. 15. Elapsed time for different length of packet sequences with the variation of link speed.

TABLE II. WORKSTATIONS WITH DIFFERENT SPECIFICATION

Work station	Specification
PC1	Intel Pentium (R) 4, CPU 2.4 GHz, 512 MB of RAM
PC2	Intel Pentium (R) 4, CPU 3.0 GHz, 0.99 GB of RAM
PC3	Intel Pentium (R) 4, CPU 3.0 GHz, 504 MB of RAM
PC4	Intel Pentium (R) 3, CPU 866 MHz, 384 MB of RAM
PC5	Intel Centrino Duo Core, CPU T2250 @ 1.73 GHz, 1024 MB of RAM
PC6	Intel Pentium (R) 4, CPU 1.80 GHz, 256 MB of RAM

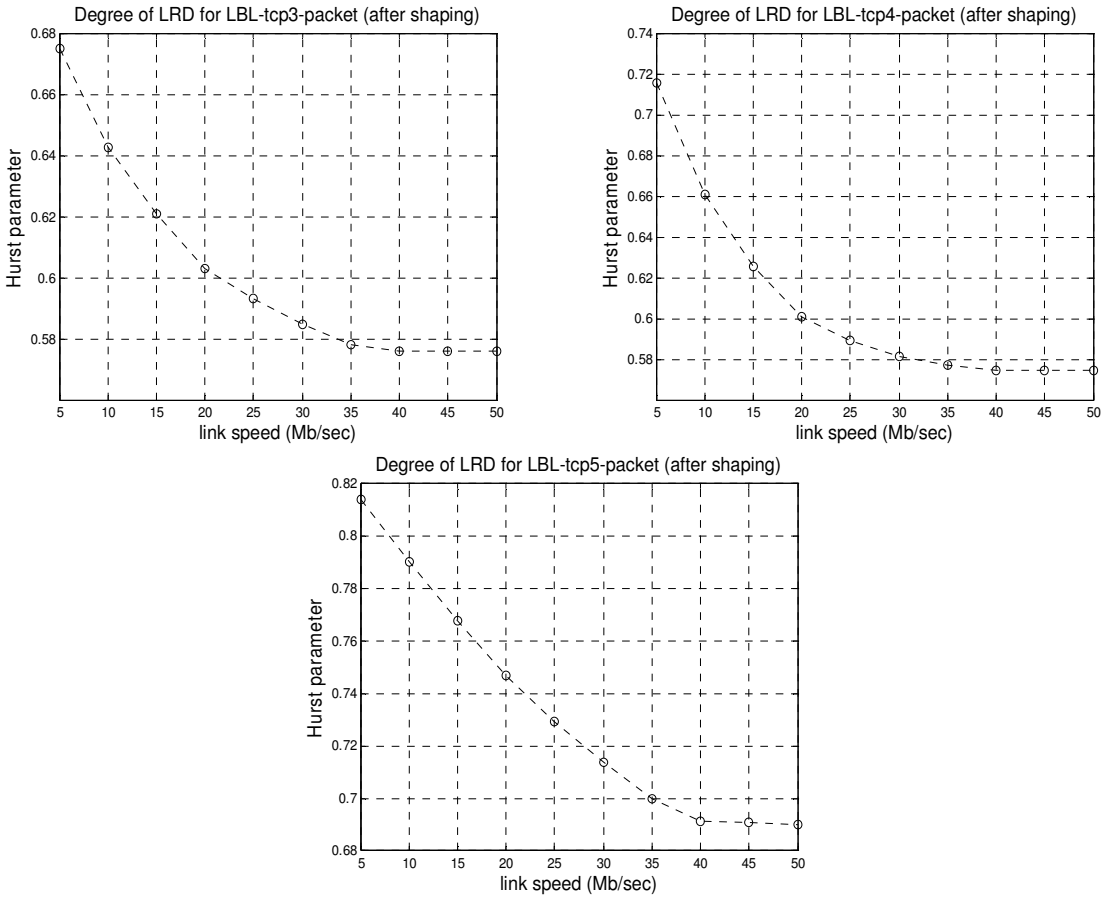

Fig 16: Variation of the degree of LRD with different link speed (C). Before shaping, the estimated H = 0.66, H = 0.68 and H = 0.7968 for LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet respectively.

Figure 16 depicts the variation of the degree of LRD with different link speeds (C) while shaping the traffic. Before shaping, the estimates are H = 0.66, H = 0.68 and H = 0.7968 for the LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet respectively. Clearly the Hurst parameter (H) decreases with increasing link speed (C), meaning that long-range dependent traffic can be reduced by the BPTraSha algorithm. As link speed is inversely proportional to link utilisation, a higher C designates lower utilisation and thereby reduces traffic bit rate, which is consistent with our findings. In Figure 2 to Figure 10 it is clear that the traffic burstiness (i.e., large variation of traffic bit rate) is reduced with the desired rate (i.e., C) by the shaping algorithm. Note that LRD cannot be reduced by simply increasing the link speed but it (C) plays an important role when using BPTraSha algorithm. Also note that at a certain limit (C = 40, 45 and 50 Mbps) the Hurst parameter remains unchanged: that is, no reduction of LRD is possible any more. The Hurst parameter is estimated here by HEAF(2) [25, 26]. Since there is an obvious correlation existing between link speed (C) and Hurst parameter (H), the algorithm has been modified as shown in Figure 17. In

Figure 17, the modified part of BPTraSha is evident in step 3 (i.e., Estimate LRD by H). The link speed is chosen according to the intensity of existing LRD traffic, estimated by H. It is clear from step 3 that the value of link speed falls into different ranges of H.

VI. Conclusion and Future Work

In this research, we present a modified version of the algorithm, BPTraSha, to control the bursty nature of network traffic. Experimental results show that the BPTraSha algorithm is capable of smoothing out the bursty nature of traffic packets received at the router buffer before they are transmitted to the core network (Internet). According to complexity and delay analyses, it is clear that the algorithm is not dependent on the size of the network or the amplitude of the spike. We naturally found that the higher the link speed, the shorter the observed delay.

Also, it is clear from Figure 16, that LRD can be reduced by BPTraSha with increasing link speeds. In the modified algorithm, the dynamic variation of link speed (C) has been shown, depending on the intensity of LRD, which is measured by Hurst parameter (H). As the main function of BPTraSha is to shape the bursty packet traffic, it can contribute to reducing the network load and lead to the improvement of QoS in future network (e.g., Internet) performance. Future work will include an evaluation of the applicability of the modified BPTraSha algorithm to real-time implementation at the FireBrick or router.

```
T = timestamp
B = Packet size in bytes
TT = transmission time
bps = Bit per second
Delt = Delay in second
Tmod = Modified time
Tmod_cng = change in modified time
bps_mod = Modified bit per second
Ld = Longest delay
Sd = Shortest delay
S = sample count (e.g. number of packet sequences)
C = link speed

1. Capture B for corresponding T (i.e. T and B)
2. Count S
3. Estimate LRD by Hurst parameter (H)
   if  H <= 0.5
      C = 1+Math.random( )*3  // to generate random C
                                between 1 and 3
   elseif  0.5 < H <= 0.6
      C = 4+Math.random( )*15
   elseif  0.6 < H <= 0.7
      C = 15+Math.random( )*35
   else  0.7 < H <= 1
      C = 35+Math.random( )*50
4. For k = 0 to (S-1)
 (a)   if (k = 0)
         bps[k] = B[k] *8 / (T[k]+TT[k])
           where TT[k] = B[k] *8 / C
       else bps[k] = B[k]*8 / (T[k]-T[k-1]+TT[k-1])
 (b)   if (k = 0)
         Delt[k] = 0
         Tmod[k] = T[k]
       else
           i)   Delt[k] = T[k]-(Tmod[k-1]+TT[k-1])
           ii)  if (Delt[k] >= 0)
                    Tmod[k] = T[k]
                else
                    Tmod[k] = T[k]-Delt[k]
5. For k = 0 to (S-2)
   i)  if (k = 0)
         Tmod_cng[k] = Tmod[k]
       else
         Tmod_cng[k] = Tmod[k+1]-Tmod[k]
   ii)  set   bps_mod[k] = B[k]*8 / Tmod_cng[k]
  iii) if (Delt[k] <0)
         find out Ld      // Longest delay
         find out Sd      // Shortest delay
6. Exit
```

Fig. 17. Modified BPTraSha algorithm

REFERENCES

[1] Karim M. Rezaul & Grout V., BPTraSha: A Novel Algorithm for Shaping Bursty Nature of Internet Traffic, Proceedings of the 3rd IARIA/IEEE Advanced International Conference on Telecommunications (AICT 2007), May 13-19, 2007, Mauritius.

[2] Amit Aggarwal, Stefan Savage and Thomas Anderson, Understanding the Performance of TCP Pacing. *Proc. of the IEEE INFOCOM 2000 Conference on Computer Communications*, March 2000, pp. 1157 - 1165.

[3] K. Park, G. Kim, and M. Crovella, On the effect of self-similarity on network performance, *In Proceedings of the SPIE International Conference on Performance and Control of Network System*, November 1997, pp. 296-310.

[4] A. Veres, Zs. Kenesi, S. Molnár, G. Vattay, TCP's Role in the Propagation of Self-Similarity in the Internet, *Computer Communications*, Special issue on Performance Evaluation of IP Networks and Services, Vol. 26, Issue 8, May 2003, pp. 899-913.

[5] L. Guo, M. Crovella, and I. Matta, TCP congestion control and heavy tails, *Technical Report*: BUCSTR -2000-017, Computer Science Dept - Boston University, 2000.

[6] Liang Guo, Mark Crovella and Ibrahim Matta, How does TCP Generate Pseudo-Self-Similarity? *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS~'01)*. Cincinnati, Ohio, pp. 215-223.

[7] Biplab Sikdar and Kenneth S. Vastola, On the Contribution of TCP to the Self-Similarity of Network Traffic, *Lecture Notes In Computer Science, Proceedings of the Thyrrhenian International Workshop on Digital Communications: Evolutionary Trends of the Internet*, Vol. 2170, 2001, Springer-Verlag, London, UK, pp. 596 – 613.

[8] J. M. Peha, Protocols can make traffic appear self-similar, *In Proceedings of the 1997 IEEE/ACM/SCS Comm. Networks and Distributed System. Modeling and Simulation Conference*, Jan 1997, pp. 47-52.

[9] A. L. Neidhardt and A. Erramilli, Shaping and policing of fractal traffic, *In 10th ITC Specialists Seminar on Control in Communications*, 1996, pp. 253-264.

[10] S. Vamvakos and V. Anantharam, On the departure process of a leaky bucket system with long-range dependent input traffic, *Queuing Systems: Theory and Applications*, vol. 28, 1998, pp. 191-214.

[11] Pruthi, P. and Popescu, A., Effect of Controls on Self-Similar Traffic, *In Proceedings of the 5th IFIP ATM Workshop*, Bradford, UK, July 1997.

[12] Vasilios Darlagiannis, Martin Karsten, and Ralf Steinmetz. Burst Shaping Queueing. In *Computer Networks and Distributed Systems (WMC)*, SCS, January 2003, pp. 65-70.

[13] S. Molnár and A. Vidács, On Modeling and Shaping Self-Similar ATM Traffic, *Proc. in 15th International Teletraffic Congress (ITC15)*, Washington, USA, July, 1997.

[14] Fei Xue, S. J. Ben Yoo, Self-similar traffic shaping at the edge router in optical packet-switched networks, *Proc. IEEE International Communication Conference (ICC 2002)*, vol.4, April 28- May 2, New York, 2002, pp.2449-2453.

[15] Dennis Bushmitch, S. S. Panwar, and A. Pal, Thinning, striping and shuffling: Traffic shaping and transport techniques for variable bit rate video, *In proceedings of the IEEE Globecom 2003*, vo.2, November 17-21, Taipei, pp.1485-1491.

[16] K. Christensen, V. Ballingam, Reduction of Self-Similarity by Application-level Traffic Shaping, *In Proc. IEEE 22nd Annual Conference on Local Computer Networks*, November 1997, pp. 511 – 518.

[17] J. Turner, New directions in communications, or which way to the information age?, *IEEE Communications Magazine*, vol. 24, 1986, pp. 8-15.

[18] Harmantzis F.C. , Hatzinakos D. and Katzela I. , Shaping and policing of fractal α-stable broadband traffic, *Proc. Canadian Conf. on Elec. and Comp. Engineering (CCECE)*, Toronto, Canada, May 2001, pp. 697-202.

[19] Brocklebank J. and D. Dickey. SAS System for Forecasting Time Series. *SAS Institute Inc*. Cary NC. 1986.

[20] Walter Willinger, Vern Paxson, and Murad Taqqu, Self-similarity and Heavy Tails: Structural Modeling of Network Traffic, Adler, R., Feldman, R., and Taqqu, M.S., (editors), *In A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, Birkhauser, 1998.

[21] Cox D., Long-Range Dependence: a Review. H. A. David and H. T. David (eds.), *In Statistics: An Appraisal, Iowa State Statistical Library*, The Iowa State University Press, 1984, pp.55-74.

[22] Leland Will E. Taqqu M. S., Willinger W. and Wilson D. V., On the Self-similar nature of Ethernet Traffic (Extended version), *IEEE/ACM Transactions on Networking*, February 1994, Vol. 2, No. 1, pp. 1-15.

[23] Antoine Scherrer, Antoine Fraboulet, Tanguy Risset, Multi-phase On-chip Traffic Generation Environment, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, *INRIA, Research Report No. 2006-22*, June 2006.

[24] *Internet Traffic Archive*: http://ita.ee.lbl.gov/html/traces.html

[25] Karim M. Rezaul and Grout V., Exploring the Reliability and Robustness of HEAF(2) for Quantifying the Intensity of Long-Range Dependent Network Traffic, International Journal of Computer Science and Network Security, Vol. 7, No. 2, February 2007, pp. 221-229.

[26] Karim M. Rezaul, Pakštas A., Gilchrist R. and Chen T.M., HEAF: A Novel Estimator for Long-Range Dependent Self-similar Network Traffic, Y. Koucheryavy, J. Harju, and V.B. Iversen (Eds.): Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN), May 29 - June 2, 2006, LNCS 4003, pp. 34 – 45.

Design of Web services filtering and clustering system

Witold Abramowicz, Konstanty Haniewicz, Monika Kaczmarek, Dominik Zyskowski
Poznan University of Economics, Department of Information Systems,
Al. Niepodległości 10, 60-967 Poznań, Poland
{w.abramowicz, k.haniewicz, m.kaczmarek, d.zyskowski} @kie.ae.poznan.pl

Abstract

The need for filtering of services results from the ever growing number of available Web services that may be used to compose various business applications. Some of the available Web services offer similar functionalities, thus the need to differentiate between them occurs. The Semantic Web services filtering process is therefore based not only on the ontological description of functional aspects of services (i.e. what a service does), but also on a description of non-functional ones (i.e. how it performs its functionality). Within the filtering process, both functional and non-functional aspects of a service expressed using ontology are confronted with the preferences a user specified and the description of a composite application the service may become a part of. One of the weaknesses of the described filtering process is lack of high efficiency and its complexity as processing ontological descriptions and reasoning on them is time-consuming. In order to speed-up the filtering process, clustering techniques narrowing down the set of potential services to be considered by the filtering mechanism may be applied. In this article the architecture for Semantic Web services filtering and clustering system is briefly discussed.

Keywords**: **Semantic Web services, filtering, clustering, architecture

1. Introduction

Service Oriented Architecture (SOA) systems may be implemented using Web services technology, which allows for easy creation of reusable components. These components serve as building blocks to create composite structures i.e. business applications that should have high level of quality and consist of the best-of-breed (i.e. the best available) components. However, the market of services is not static and the number and properties of services are changing constantly. There were over 20.000 publicly available services in 2005 [1], whereas about 1200 in 2004 [24]. According to the latest research of Al-Masri and Mahmoud the number of publicly available Web services between October 2006 and October 2007 increased by 131% [36]. With the augmented appearance of service substitutes, a need emerges not only to identify the functionally relevant services but also to distinguish the best-fitting ones to be used within the composition. Once relevant components are identified, e.g. better in terms of non-functional aspects than the already used ones, the replacement of components in the application may follow.

In order to efficiently perform the process described above, the need for service selection, discovery and filtering arises. Due to the overwhelming number of Web services, which will exceed human cognitive capabilities, automation of these processes is strongly recommended. It may be achieved by using semantics and Semantic Web technologies [2] - in the consequence by the exploitation of Semantic Web services (SWS) paradigm. However, although using semantic allow for automation, most of the processes based on the ontology are time and resource consuming.

In this article, which is related to our ICIW 2007 publication [38] we propose architecture and algorithms for filtering and clustering to support identification of relevant services and selection of the best-fitting Semantic Web services to be used by a business system using external Web services. The proposed system is an extension of the F-WebS project [4] and may be used also in the context of Semantic Web services e-marketplaces [37].

The structure of the article is as follows. First, in the section 2 the related work is discussed. Then, we present a motivating scenario that will justify the application of service filtering system. In the next section the basic definitions relevant to the concept of service filtering and clustering are presented. In the following section the architecture of the implemented system is shown. Finally, the future work and conclusions are given.

2. Related work

Semantic Web services and their applications are one of the most popular research topics these days. Some researchers focus on creating the adequate semantic description of a Web service that would make this idea possible – e.g. OWL-S [3], WSMO [5], WSDL-S [6] or SAWSDL [28] while others concentrate more on mechanisms and algorithms used within the Semantic Web services description based interactions [7]. Many of the publications on service interactions tend to put more emphasis on certain aspects of reasoning [8, 9] rather than on focusing on current constraints and foreseeable evolvement of service interactions.

The ultimate challenge in the SWS world is still an issue of expressive description, reasoning mechanisms and their efficiency [14, 15]. Dealing with the ontologized description of a service implies the necessity to use the appropriate reasoning engines. Researches in AI and knowledge representation emphasize the fact that a choice between expressiveness of the notation and efficiency has to be made (due to feasibility of the task). Taking this issue into account most of the initiatives in the SWS field decides to use description expressed in the terms DL [12]. Nevertheless, the efficiency of the performed processes is still an open problem.

There are many publications describing the architecture of Semantic Web services systems performing various interactions among others also discovery and matchmaking of services in various domains [9, 10, 11, 12, and 13]. What is more, there is tremendous research effort in several EU-funded projects (some still ongoing) that deal with Semantic Web services and their applications in business context (e.g. ASG [12], METEOR-S [13], SUPER [25]). To our best knowledge there is none among them using the algorithms described in this paper.

The filtering of Web services and then Semantic Web services was first proposed by Abramowicz et al. [4]. It takes advantage of the achievements in SWS description and discovery area [9, 16, 17, 18 and 19] as semantic-based Web services filtering uses a variant of matching algorithms similar to ones used in Semantic Web services discovery process.

The idea of Web services clustering is not a novel one. First attempts were made based on the WSDL service description [20]. However, the effective and precise SWS clustering is still an ongoing research topic. The majority of researchers have left illusions of any reasonable results based on adoption of standard methods derived from the information retrieval field. At the moment, the only feasible solutions base on the employment of semantics and creation of similarity measures that take advantage of the underlying ontologies [20, 29]. It also seems that the most important issue associated with Web services clustering is the similarity measure, which has to fully map the relationships between various, differently formulated however similar Semantic Web services.

3. Motivating scenario

One of the reasons to build system according to the SOA paradigm and use the Web services technology is to easily and rapidly compose applications out of available services. Even though, the current state of publicly available Web services is far from the envisioned one, a user has an access to a variety of simple services that may be used to create a piece of software of real utility to business users.

The aim of this example is to sketch up a real world situation where not only a practical Web services based application is created but also it is maintained with the support of Web services filtering and clustering system.

The domain of application has been selected based on the following criteria:

- current availability of services,
- possibility of application,
- perks from application,
- relative ease of services description.

In our opinion the best domain for the practical illustration of our research is a financial area which has two following advantages: existing variety of services, wide spectrum of potential applications also for non-enterprise end-users.

An exemplary application built out of services is designed to manage personal finance, having an access to bank accounts, and authority to transfer money among accounts, buy or sell. This ideal example is based on general assumptions that the application can represent its user having all his rights. Discussion of soundness of this statement is beyond the scope of this article.

Table 1 enumerates necessary services (to be specific - service types, not the exact Web services) which have to be encapsulated to form desired application.

The mentioned elements may form an application that invests any superfluous money on bank accounts in one of the possible ventures. For example, by analyzing the exchange course between any pair of currencies, a user can decide (basing on a suggestion made by the discussed application) to play arbitration games by exchanging money from one currency to other. The user can choose alternatively to make a deposit in a

bank that has a higher interest rate that the one that possesses additional funds. Examples can be easily multiplied.

Table 1. Financial services

Service	Functionality description
financial situation	service should enlist all the liquid assets and on this basis one may undertake decision
money transfer	an application to be useful has to be able to transfer money from one account to other, thus the need of transfer functionality
exchange course	exchange course informs user of the ratio between currencies
trend	trend functionality should return tendency of some input data. E.g. introducing average price of some commodity, the functionality should return whether there is steady rise, fall, stagnation or some seasonal fluctuation
risk	risk service in the simplest mode should describe the deviation of prices of any commodity throughout some time
invest-ment situation	investment situation should list the actual state of all investments made. It should be based on their history.
invest-ment history	investment history apart from delivering data to the investment state should provide some manner of report creation for introspective reasons
interest monitor	interest monitor should be able to provide maximum information of interest rates of different commodities, such as stock options, raw materials, precious metals etc.
transfer cost	informs about financial viability of transaction (whether there should be a gain that is lesser than the cost).

The sheer power of the proposed solution lays in the constant monitoring of elements and providing suggestions for any possible tweaks and exchanges in the orchestrated workflow. Imagine that new kind of Web service that provides more accurate approximation of trend or risk evaluation should appear. The system will filter out any Web service that can be an upgrade for any of the components used in the application taking into account both functional and non-functional parameters that were defined by the user in his profile.

This stage is crucial for the system as not only users may enhance their application but also they may be sure that the elements they use are the best ones fitting their needs and preferences.

4. Web services filtering and clustering

This section is divided into three subparts. The first one provides general information on the process of both clustering and filtering. The second describes various considerations of the clustering task. Finally, the third subsection presents details of the filtering process.

4.1 General information

In general, information filtering can be described as specifying which objects from a given stream are relevant to a given profile. According to [21], a profile is a representation of regular information interests that may change slowly over time, as conditions, goals and knowledge change. This representation is used in filtering systems to provide users with information with the highest relevance.

In the Fig. 1 the process flow in the Semantic Web services filtering and clustering system defined based on the general architecture of the filtering systems [21] is presented. However, the filtering process has been enhanced with few additional activities aiming at increasing the effectiveness of the system (i.e. clustering).

In this model three main functional sections can be distinguished: service description creation (A), profile creation (B), and filtering and refinement process (C) (here comes into play clustering algorithm which not only saves execution time but also improves refinement of stored data). Each section consists of several subprocesses.

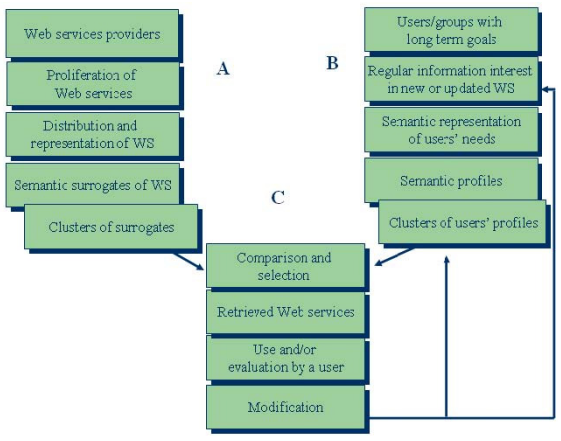

Figure 1. Semantic Web Services Filtering and Clustering

The filtering begins with clients of the filtering system having relatively stable and long-term goals (to consist of the best-of-breed components). Attaining such objectives is connected with acquiring information about new or updated services on the e-marketplace. This entails a demand for information, which is subjected to continuous changes mainly due to natural change of users' goals, conditions (e.g. change in components that are part of a composite application) or changes on the e-marketplace itself. The users' or applications' information needs can be represented in the form of profiles placed in the filtering system. In our system, the profiles are represented by enhanced OWL-S description [4] (OWL-S with extended list of non-functional properties, so that larger number of quality aspects would be taken into account in the filtering process). Additionally, user preferences assigned to the specific component of a business application are also included in his profile (e.g. the credit card payment service needs to be secure).

Simultaneously, service providers attempt to distribute their services, so that users become aware of them. Such services are represented as semantic artefacts that are amenable to computer processing. In order to make service descriptions and user profiles comparable, both are reduced to sets of attributes that map objects to a common representation. That is why as the service description within our system again enhanced OWL-S was used.

Moreover, in order to increase the effectiveness of the filtering process, the clustering analysis on profiles and services is performed. Main goal of the clustering algorithm is to shorten the duration of the matchmaking between the user's profile and the profiles of services stored in the repository. The efficiency paradigm of this task limits the range of feasible algorithms to those which can update their clusters within the online transaction and make use of medioids as the common denominator for the whole cluster.

4.2 Clustering details

Before we delve into the matter of medioids, their definition and selection has to be introduced into general set of assumptions and representations used for the need of the clustering task.

By enhancing Web services with semantic annotation a new set of additional information is introduced. This information set is represented by already mentioned domain ontology and should be used within the clustering process.

Traditional approach to representation of entities for clustering needs [32] postulates a vector-like representation of every service. Naive implementation would follow this idea by mapping ontology to a vector of data where every consecutive field would denote either absence or presence of some trait taken from the abovementioned domain ontology and checked with a Web service in scrutiny.

In addition to fact that this approach seems to be an excess in terms of meaningfulness (one has to bear in mind that ontologies grow and evolve, one vector cannot suffice all Web services without assumption of changelessness [33]), one should also consider the sheer amount of data that has to be loaded into computer's memory for the sake of computation.

Every Web service description takes into account four most important aspects: inputs, outputs, preconditions and effects (the same assumption is used within the filtering phase that is discussed in more detail later on in this section). Due to the fact that preconditions and effects are hard to define without considering every usage scenario, practice has dictated to annotate only inputs and outputs.

When these two are taken into consideration, a Web service can be described as pair of two vectors, first for all input parameters and second one for output parameters.

$$ws = (i, o)$$
$$i - vectorofinputs$$
$$o - vectorofoutputs$$

Formula 1 - Web service description as a pair of input and output vectors

If a Web service is to be perceived as an abstraction for function, vector of outputs can be replaced by a single value. Nevertheless, in a general usage scenario there are two vectors which size depends on the buoyant environment as any domain ontology which is used for description of every parameter is prone to change.

The key element of clustering task is a representation of distances among all Web services stored in a system. However, a general distance function to be used for Web services clustering is hard to define due to varying number of parameters and interactions occurring among services. Here, one has to consider a usage of distance matrix, where a distance among services is presented as a pair of two values where, in accord to representation chosen before, first value represents distance between input parameters of two Web services and the other distance of output parameters.

$$\begin{bmatrix} (dws_{1,1}i, dws_{1,1}o) & \cdots & (dws_{1,n}i, dws_{1,n}o) \\ \vdots & \ddots & \vdots \\ (dws_{n,1}i, dws_{n,1}o) & \cdots & (dws_{n,n}i, dws_{n,n}o) \end{bmatrix}$$

Formula 2 - Distance matrix for semantically annotated Web services

Where $dws_{x,y}i$ denotes distance between Web service's x inputs and Web service's inputs y and $dws_{x,y}o$ between output vectors.

Of utmost importance is to emphasize once more that we deal with varied entities. Varied to the point where one has to compare sets of information describing different number of parameters.

Thus, a need for heuristic functions to bring different Web services to a common denominator in terms of inputs and outputs occurs.

In the general case, the distance is calculated with use of reasoner, yet it can be delegated to other software if one would decide to ignore other than hierarchical relationships in domain ontology. It is a common practice due to difficulties with evaluation of the impact that these other types of relationships impose to the problem domain – in some cases they introduce ambiguity in others they are irrelevant.

The complexity of distance computation is of O(n2) class due to the need of cross-examination of every parameter of a first service against every parameter of a second one. The same situation appears within the first stage of general filtering phase. It is to be remembered that this procedure is performed for each pair of services in the system and that the distance matrix is not symmetric due to the nature of relationships and their meaning for distance computation (simple inversion of values is not a valid value of inverse distance measure).

For the sake of discussion, consider following assumptions as to the values representing relations among concepts:

- when two parameters in question annotated with concepts from a domain ontology are subsuming one another a default value of 0.75 is used,
- default value for subsuming concepts is being modified depending on number of levels between them, when an ontology is treated as a taxonomy – i.e. how much more general one concept is from the other (default 0.75 is reserved for case of simple derivation – no other concepts lay on the path from the more general one to the less general one),
- when dealing with inverted subsuming concepts a default value of 0.25 is introduced that can be modified in analogical manner to the above-described one,
- when two concepts match a value of 1 is used,
- when there is no relation between parameters 0 is used to denote the state.

When one is presented with three different semantically annotated Web services i.e. ws_1, ws_2 and ws_3 which for the presentation's sake have the same number of input and output parameters, one can observe how distance is represented in general.

$$dws_{1,2} = ((1,0.75,0),0.5)$$
$$dws_{1,3} = ((0.25,0.5,0.25),0.25)$$

Formula 3 - An example of distances between Web services

Where $dws_{x,y}$ denotes distance between a Web service x and a Web service y.

As mentioned before, the situation in the example is rather simple one. Nevertheless, it can be used to demonstrate that average which could be easily applied to answer which Web service is closer to the first one is not applicable when the assumption of equal number of parameters is removed.

A first step to forging out of good heuristic function which would serve us in distance measurement is to answer a question of variable parameter number in Web services to be compared.

It is known that when a Web service has less input parameters in comparison to another one the situation is far from being optimal for the algorithm. We cannot assume that a Web service is worse (in terms of effectiveness) due to a fact that it does not use of all information in our possession. This statement is derived from observation of extra parameters that are treated as default values thus bearing no interest to the user in terms of his preferences. The possible composition of parameters (inputs or outputs) has not been proved to give satisfactory results in general use cases [34].

We can make an assumption that a parameter described by a more general concept is more desirable than a parameter described by less general one when we deal with input parameters and in reverse manner when we deal with output parameters (the more specific, the better).

Lack of relation between parameters is to be penalized in a manner similar to the situation when the number of parameters is different for both Web services.

When we employ a reasoner to answer whether two concepts are in relation (one subsumes the other one or they are identical) or no relation is present and the assumption of taking into account hierarchical relations holds we face a problem of meaningful information loss. Pray consider two concepts that are siblings (thus share a common parent). We can speculate that they can be in strong relation despite the fact that reasoner returns no relation value. To enable algorithm not to miss this kind of data it has been enhanced by a routine that check which concepts are instantiated and which serve only as classification ones.

If one is to consider a concept of currency that has only two subconcepts, euro and dollar concepts it is easy to weed out all input and output parameters that are of these types. Highly probable is that no service would employ a concept of currency but its specialization, either euro or dollar. Thus, we gain knowledge that the two are in strong relation and are possibly interchangeable.

All these is gathered and presented along with example of computation of distance measures among Web services in every cluster represented by medioids in further part of the section. Nevertheless, one can easily see that all presented steps were introduced to show how original distance matrix is to be transformed into a one that can be used by one of the well described clustering algorithms ($dws_{x,y}$ – transformed distance measure between two web services).

$$\begin{bmatrix} dws_{1,1} & \cdots & dws_{1,n} \\ \vdots & \ddots & \vdots \\ dws_{n,1} & \cdots & dws_{n,n} \end{bmatrix}$$

Formula 4 – Transformed distance matrix for semantically annotated Web services

We have proposed to represent the medioid as a most general Web service's profile i.e. the profile which has the factor of generality of the highest rank (the factor of generality is the weighted average of inputs, outputs and the existence of necessary non-functional parameters). This approach has been used by [29]. The issue with this factor arises from the fact that some Web services can take more inputs as others yet provide the same functionality (as depicted above in the section). It is easy to depict such an example. Let us assume that in the domain ontology the notion of amount of money is defined as an actual amount and the currency which applies to it. One Web service may take only one input with the stated amount of money already with the currency denominator, other may take two separate inputs one for the number, another for the currency. Controlling the domain ontology gives the ability to state that the inputs from the second Web service are encompassed by the one from the first. Therefore for such a simple case in which we have the two mentioned services put together in the cluster the first Web service is chosen for medioid. In current implementation the output has greater weight in the generality factor as obtaining what we want neglecting all what we do not need, has a greater value to the potential user.

The arising questions of possible multiple inputs out of which some do fit into the pattern and others do not, is generally well handled as the stored Web services provide only one functionality (one Web service does one thing, thus resembles a function in programming language). Furthermore, if one is to decide whether the Web service fits into a cluster by examining whether the output suits the not neglected inputs thus satisfying the goal of algorithm.

When medioids are chosen, one has to decide on the number of clusters in the repository. If we are to consider only the most general concepts (as F-WebS uses OWL, most general concepts are those derived directly from owl:Thing) we have to put certain amount of trust into domain ontology architect's skills in the matter of granularity choice of the main concepts.

Alternatively, one can use expert's approach to amend possible shortcomings of granularity induced by the ontology architect and come up with a number that is more desirable.

Natural algorithm for clustering when medioids are present is Partitioning around medioids [35]. There are four main steps in the algorithm:

- Initialization. Setting desired number of clusters (k). Domain is not covered by any of clusters. We initialise k medioids.
- Algorithm checks for the closest element to one of k medioids.
- Test for stop criterion is performed. Is the whole domain covered by the target number of clusters? If the answer is affirmative, algorithm ends its work, else it goes on.
- Medioids are updated with freshly found closest elements and thus a need for their recalculation arises. When finished, the algorithm returns to the second step.

Equipped with all the necessary information it is time to review an example. Let us consider the situation with three Web services:

- ws_1 has four input parameters: begin, end, stock-exchange and country. It has one output parameter of euro type.
- ws_2 has two input parameters: time-period and market, it has one output parameter of dollar type
- ws_3 has one input parameter of country type and one output parameter of timestamp type.

Services are described with the following example domain ontology.

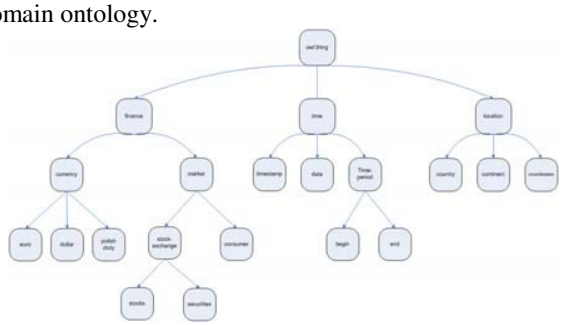

Figure 2. Example domain ontology

First phase takes into account number of parameters (whether an over or underflow is present) and it pairs best fitting parameters along with evaluation of their numerical relation. The phase is shown in Table 2.

Table 2. First phase of distance measure transformation

	parametres					flow	
	input				output	-	+
ws_1	begin	end	stock-exchange	country	**euro**	2	0
ws_2	time-period		marke		**dolar**		
result	*0,25*		*0,25*		*0,4*		
ws_1	begin	end	stock-exchange	country	**euro**	3	0
ws_3				country	**timestamp**		
result				*1*	*0*		
ws_2	time-period	market			**dolar**	0	2
ws_1	begin	end	stock-exchange	country	**euro**		
result	*0,75*	*0,75*			*0,4*		
ws_2	time-period	market			**dolar**	1	0
ws_3	country				**timestamp**		
result	0				0		
ws_3	country				**timestamp**	0	3
ws_1	begin	end	stock-exchange	country	**euro**		
result	*1*				*0*		
ws_3	country				**timestamp**	0	1
ws_2	time-period	market			**dolar**		
result	*0*				*0*		

First part of the table 1 informs us of which services were analyzed. Second part gives information of paired parameters, their relation (notice use of 0.4 for parameters that would have no relation in standard reasoning yet were classified as instantiations of general concept) and underflows (column -) and overflows (column +) in the number of parameters.

The second phase is started with computation of distance of input parameters using following formula:

$$d_i = \mu(1 - 0.1l - 0.05e)$$

Formula 5 – Distance measure for input parametres

Where μ stands for relations average in the analyzed parameters, l for underflow of parameters and e for overflow of parameters.

For our example results are presented in table 3.

Table 3. Input distances matrix

services	*ws_1*	*ws_2*	*ws_3*
ws_1	1	0.2	0.35
ws_2	0.675	1	0
ws_3	0.85	0	1

Due to the fact that every service has only one output calculations of output distance are obvious and presented in the table 1.

The final step is to combine transformed inputs and outputs to come up with transformed distance matrix. This is achieved by applying the formula 6:

$$d_{ws} = (0.45d_i + 0.55d_o) - h$$

Formula 6 – Final transformation of distance measures

Where di is a distance measure of inputs, do is a distance measure of outputs and h is penalty applied when the absolute difference between is greater than 0.4 and is greater than 60% of value expressed by first part of formula 6.

Final distance matrix is presented in table 4.

Table 4. Final distance matrix

services	*ws_1*	*ws_2*	*ws_3*
ws_1	1	0.31	0.1575
ws_2	0.52375	1	0
ws_3	0.135	0	1

4.3 Filtering

The filtering algorithm works in two stages. The first stage, ontology-based filtering aims at detection of service substitutes. It is quite similar to the typical matchmaking process. There are a few algorithms that match functionalities of provided and requested services [7]. Some of them are divided into stages, while some do everything in one step. We decided to take advantage of the method proposed in [19]. For more details regarding the filtering process see [4].

Analyzed elements of OWL-S service description are following: inputs, outputs, and service category. The algorithm starts whenever new service appears on the market. It checks whether the functionality of the service is relevant to any user profile. In order to shorten the time it is compared not to each service separately but to the medioid representing every cluster of services. If the new service turns out to be relevant to the given medioid, it is then compared to each and every service from the cluster in question. The four levels of matching between two properties/parameters were distinguished

- Equivalence - concepts have the same meaning;
- Subconcept - one concept is a subconcept of the other concept;
- Unclassified - one of two concepts is not classified;
- No relation - in other cases

Functions that determine levels of match use the ontology reasoner Pellet [30].

The final result is an aggregation of results from all partial comparisons. The new service is relevant to the profile when the final result of ontology-based filtering is higher than the threshold defined in the system. In other cases new services are turned down and not passed to further analysis.

The second stage of procedure is the constraint-based filtering. Its objective is to identify the best service from the set of relevant services according to the user preferences. Some propositions to compute utility function over the given parameters can be found in [22, 23], it is also possible to compute a distance measure, but it does not take preferences into account. That is why we have decided to take advantage of a multiple criteria analysis (MCA). Using this method services can be compared according to their characteristics, e.g. price, response time, accessibility. To each characteristic a weight is assigned, reflecting an application's preferences [4]. The exact method used to compare phenomena is computation of the synthetic indicator. For example, if two services are given, together with some statistics concerning their characteristics e.g. response time, reliability etc., it may occur that one of the services performs better according to reliability, while the other is more accurate and less expensive. Additionally, one service is paid by credit card and another by wire transfer. The synthetic indicator allows for comparison of such services, given the vector of user preferences. For details concerning this stage see [27]. The highest value is chosen as the indicator of the best service. If the best service is the incoming service then the user is notified.

5. Architecture of the system

The architecture of the system described in the previous section should consist of at least few components connected according to the SOA paradigm. The conceptual architecture model of Filtering and clustering system is presented in the fig. 3.

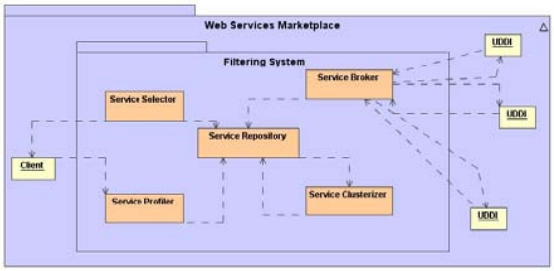

Figure 3. The conceptual model of the Filtering and Clustering System

In order to perform the filtering of newly appearing services, the system needs to store service profiles in the repository. To be able to interact with service providers and collect the information about new services the system should have the broker functionality (active search for new services, being passive source on information for the filtering system, performing all the necessary interactions with providers in order to create system-processable service descriptions and acquiring information about service quality parameters). The other important component of the system is the filterer. Its task is to perform the analysis of QoS constraints and semantic matching between two, potentially similar, Web services. Service Clusterizer creates groups of similar profiles. The last important element of the filtering system is a Service Profiler. This component is used by clients to create profiles of composite applications that use Web services.

The idea behind the prototyping was to use as many open source tools to increase the popularity of the solution. The prototype of implemented system was written in Java with use of XPath and MySQL as the database server. In its earliest version the system had Web-browser interface, as this way of communication is extremely user-friendly (implemented in Java servlets technology).

The appropriate experiment on the scenario described in the section 3 was conducted. The results were promising. The usage of the clustering algorithm has no impact on the precision and relevance of the system but it speeds up the process of processing the single service. As it works independently of the service discovery and filtering functionality, it may work in the background continuously updating the created clusters taking into account new services in the profiles. A brief summary of all system components is given in following subsections.

5.1 Service Repository

The service repository has two main functionalities: it stores service profiles created by the system clients and stores the information about all services on the market. Initially we considered the division of these two functions into two components. But some factors convinced us to keep everything in the one repository

- the service registered by a user can be also a new service on the market,
- clustering performed on the bigger sample of services gives better results.

Service repository is a database. It also triggers Service Clusterizer when new service appears.

5.2 Service Broker

The broker plays the role of intermediary between providers and the system itself. Broker can actively seek new services or be just passive receiver of notifications from UDDIs. When new WSDL file comes to the system it is automatically converted to OWL-S format. Afterwards, the broker asks provider for giving the information about non-functional properties' values of the service. When it is done, the service description (profile) is stored in the repository. Broker should give an interface that helps providers complete descriptions of their services. This functionality may be also provided as a stand-alone application that converts WSDL services to OWL-S format enhanced with parameters required by the filtering.

5.3 Service Profiler

The Service Profiler's goal is to help client express his needs. A client, through specialized interface defines the properties of Web services of which his application consists. The OWL-S descriptions of every atomic service are stored in the repository. Moreover, a client can define desired values of QoS parameters. Additionally, it is possible to put weights on these parameters, because for example, one client prefers cheap, but less reliable service, whereas other one is able to pay more for more reliable Web service. Every OWL-S file has accompanying vector of preferences. Altogether, they create a profile of an ideal atomic service. Such a profile is later clusterized. This profile is also matched against new services registered in the system.

5.4 Service Clusterizer

This component handles the task of clustering of atomic services stored in the repository. It is worth noting that the criterion of the clustering process is the functionality of a service. Thus, services of different providers can be grouped in one cluster. The granularity of clusters has several levels. The highest level relates to service category, lower ones are created according to the level of semantic equivalence between services in the same cluster. Effects of the clustering process are later taken into account during the filtering. New services are compared only to corresponding cluster. In the effect the number of comparisons is dramatically lower, because for example new payment service is not semantically matched to weather service.

5.5 Service Selector

Service selection is performed in two stages. The first phase, called the ontology-based filtering, is responsible for semantic matching of services functionalities. In the next phase, the non-functional properties are analyzed. When the overall level of match between the new service and the service in profile exceeds threshold value the client gets the notification that new, better service was filtered by the system.

6. Summary and future work

The presented architecture of the Semantic Web services filtering and clustering system may solve some of the problems of the SOA paradigm. The system consists of several components dealing with one aspect

of the task. The elements are chained in a workflow that reflexes the step-by-step solution. The results of the filtering process as presented in Abramowicz et al [4] are promising but not as precise as one could wish for, mainly due to the ontology related problems. In our opinion, one of the main problems is to define a precise ontology and service profiles for Web services description so the right services could be matched and filtered according to the requirements and user's preferences. We do not expect that clustering of Web services will be a remedy for all the problems connected with the efficiency of semantic matching algorithms. However, the limitation of the set of compared services can save a lot of time, as reasoning on ontologies is undoubtedly time-consuming process. Our next step is to show the results proving the usability of the clustering in the filtering process.

Additionally, one of our goals is to improve the semantic matching effectiveness by better description of preconditions and effects. Well-prepared financial ontology would be a great tool to achieve this goal. We also plan to extend the list of non-functional features that are taken into account during the constraint-based filtering stage. Works driving at creation of upper and lower ontology of non-functional properties are in progress.

Another question is how the proposed approach deals with composite services, that is, services that are composed by other services that can be discovered and replaced dynamically during the runtime. Orchestration problems can arise during the execution of such composite services when new potential services arise during a discovery process. This is however, the aim of the further research.

7. References

[1] Bachlechner, D., Siorpaes, K., Fensel, D. and Toma, I. Web service Discovery – A Reality Check, DERI Technical Report, January 2006

[2] Berners-Lee, T., Handler, J., Lassila, O., The Semantic Web, Scientific American, May 2001

[3] OWL-S, http://www.daml.org/services/owl-s/

[4] Abramowicz W., Godlewska A., Gwizdała J., Kaczmarek M., and Zyskowski D. Application-oriented Web services Filtering, in Proceedings of the International Conference on Next Generation Web services Practices (NWeSP 2005), pages 63-68, IEEE August 2005

[5] WSMO, http://www.wsmo.org/

[6] WSDL-S, http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf

[7] Paolucci, M., Kawamura, T, Payne, T., Sycara, K., Semantic Matching of Web services Capabilites, In Proceedings of the 1st ISWC, 2002

[8] Gonzalez-Castillo, J., Trastour, D., Bartolini, C., Description logics for matchmaking of services, In KI Workshop on Applications of Description Logics, 2001;

[9] Li, L., Horrocks, I., A software framework for matchmaking based on semantic web technology, In Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, May 2003.

[10] Bussler, Ch., Maedche, A., Fensel, D., A Conceptual Architecture for Semantic Web Enabled Web services, ACM Special Interest Group on Management of Data: Volume 31, Number 4, Dec 2002

[11] Deng, S., Wu, Z., Li, Y., ASCEND: a framework for automatic service composition and execution in dynamic environment, in Proceedings of International Conference Systems, Man and Cybernetics, 2004, pages 3457-3461

[12] ASG http://www.asg-platform.org

[13] METEOR-S: Semantic Web services and Processes, lsdis.cs.uga.edu/proj/meteor/SWP.htm

[14] Lara, R., Laursen, H., Arroyo, S., de Bruijn, J., Fensel, D., Semantic Web services: Description Requirements and Current Technologies. In International Workshop on Electronic Commerce, Agents, and Semantic Web services, September 2003

[15] Abramowicz W., Godlewska, A., Gwizdała, J., Jakubowski, T., Kaczmarek, M., Kowalkiewicz, M., Zyskowski, D., A survey of QoS computation for Web Services Profiling, In the Proceedings of ISCA 18th International Conference on Computer Applications in Industry and Engineering 2005, Honolulu, USA

[16] Verma, K., Sivashanmugam, K., Sheth, A.. Patil, A., Oundhakar, S. and Miller, J., METEOR-S WSDI: A scalable P2P infrastructure of registries for semantic publication and discovery of web services. Inf. Tech. and Management, 6(1):17–39, 2005.

[17] Lynch, D., Keeney, J., Lewis, D., O'Sullivan, K., "A Proactive Approach to Semantically-Driven Service Discovery", in the Proceedings of 2nd Workshop on Innovations in Web Infrastructure, May 2006, Edinburgh

[18] Srinivasan, N., Paolucci, M., Sycara, K..., Semantic Web Service Discovery in the OWL-S IDE, HICSS, p. 109b, Proceedings of the 39th HICSS'06, Track 6, 2006

[19] Jaeger, M., Tang, S., Ranked Matching for Service Descriptions using DAML-S. 2004

[20] Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J., Similarity search for web services. In Proc. of VLDB, 2004

[21] Belkin, N. J., Croft, W.B., Information filtering and information retrieval: two sides of the same coin?, Communications of the ACM, 35(12):29-37,1992

[22] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q., Quality driven Web Services Composition, in Proceedings of the 12th international WWW conference, Budapest, Hungary, May 2003,

[23] Liu, Y., Ngu, A.H.H., Zeng, L., QoS computation and Policing in Dynamic Web Service Selection, in Proceedings of the 13th international WWW conference, New York, USA, ACM Press, May 2004

[24] Myeon Kim, S., Catalin Rosu, M., A Survey of Public Web Services, WWW 2004, May 17–22, 2004, New York

[25] SUPER: Semantics Used for Process management within and between EnteRprises, http://www.ip-super.org

[26] Aslam J.A., Pelekhov E., Rus D., The Star Clustering Algorithm for Information Organization Grouping Multidimensional Data, Recent Advances in Clustering, Springer-Verlag, Berlin, 2006

[27] Abramowicz W., Haniewicz K., Kaczmarek M., Zyskowski D., Filtering of Semantic Web Servics with F-WebS System, The Semantic Web: ASWC 2006 Workshops Proceedings, p. 317-324

[28] Semantic annotations for WSDL http://www.w3.org/TR/sawsdl/#Intro

[29] Taush, B., d'Amato, C., Staab, S., Fanizzi, N., Efficient Service Matchmaking using Tree-Structured Clustering, 5th ISWC 2006
Athens, GA, USA, November 5-9, 2006

[30] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y., Pellet: A practical OWL-DL reasoner, Journal of Web Semantics, 2006.

[31] Jaeger, M., Tang, S., Ranked Matching for Service Descriptions using DAML-S. 2004

[32] Sebastiani F.: Machine Learning in Automated Text Categorization, ACM Computing Surveys, 2002

[33] Noy N. F., Klein M.: Ontology evolution: Not the same as schema evolution, Technical Report SMI-2002-0926, Stanford Medical Informatics, 2002

[34] Srivastava B., Koehler J.: Web Service Composition Current Solutions and Open Problems, ICAPS 2003

[35] Kaufman L., Rousseeuw P.: Finding groups in data: an introduction to cluster analysis, New York: John Wiley and Sons, 1990.

[36] Al-Masri, E., Mahmoud, Q.H.,Investigating Web Services on the World Wide Web, WWW 2008.

[37] Abramowicz W., Haniewicz K., Kaczmarek M. and Zyskowski D. E-Marketplace for Semantic Web Services, Service-Oriented Computing – ICSOC 2008: 6th International Conference, Sydney, Australia, December 1-5, 2008.

[38] Abramowicz W., Haniewicz K., Kaczmarek M., Zyskowski D., Architecture for Web Services Filtering and Clustering, International Conference on Internet and Web Applications and Services (ICIW'07)

Towards Open Tracing of P2P File Sharing Systems

Danny Hughes
Computing, InfoLab21,
Lancaster University,
Lancaster, UK.
+44 (0)1524 510352
danny@comp.lancs.ac.uk

Kevin Lee,
School of Computer Science,
University of Manchester,
Manchester, UK.
+44 (0) 161 2756132
klee@cs.man.ac.uk

James Walkerdine
Computing, InfoLab21,
Lancaster University,
Lancaster, UK.
+44 (0)1524 510352
walkerdi@comp.lancs.ac.uk

Abstract

Since the release of Napster in 1999, peer-to-peer file-sharing has enjoyed a dramatic rise in popularity. A 2000 study by Plonka on the University of Wisconsin campus network found that file-sharing accounted for a comparable volume of traffic to web applications, while a 2002 study by Saroiu et al. on the University of Washington campus network found that file-sharing accounted for more than treble the volume of web traffic observed, thus affirming the significance of P2P in the context of Internet traffic. Empirical studies of peer-to-peer traffic are essential for supporting the design of next-generation peer-to-peer systems, informing the provisioning of network infrastructure and underpinning the policing of peer-to-peer systems. This paper surveys existing work in the field of peer-to-peer monitoring and based upon this assessment of the state-of-the-art describes the design and implementation of the Open P2P tracing system. This system aims to improve the research community's understanding of P2P file sharing systems by providing continuous and up-to-date traffic data which is anonymized and made freely accessible to all interested parties. Data from this system has been used in a variety of projects and papers, which are used to illustrate the broad range of research that can benefit from an open tracing system.

Keywords: Peer-to-Peer (P2P), File sharing systems, Monitoring approaches

1. Introduction

Since the release of Napster [1] in 1999, peer-to-peer (P2P) file-sharing has enjoyed a meteoric rise in popularity, to the point that P2P applications are now widely considered to be responsible for more traffic than any other Internet application [2]. Given the scale of P2P traffic, understanding traffic characteristics is of critical importance and has specific benefits in the context of: i) provisioning network infrastructure, ii) informing network policy, iii) informing the design of new P2P applications, iv) managing existing P2P communities and, v) policing P2P systems.

Many significant studies of P2P file sharing systems have been performed. These studies have illuminated a range of P2P characteristics; however, we believe that there remain significant shortcomings in the current body of research on P2P file sharing systems. These shortcomings include:

- ***The extensive use of closed data sets***, which prevents the findings of existing studies being revisited. Furthermore, as truly representative P2P traces may take months or even years to perform, the use of closed data-sets has led to significant and unnecessary duplication of effort.
- ***Trend analysis*** is poorly supported by existing studies, which, with a few exceptions [3] [4], are not of sufficient duration to reveal long-term trends in user behaviour.
- ***Cross discipline perspectives*** are often lacking in existing studies, which tend to concern themselves largely with technical factors and often fail to consider factors such as group psychology, the economics of file sharing and the ethics of monitoring real-world distributed systems.

We suggest that these shortcomings may be addressed through the development of an 'Open P2P Tracing System' [37] which aims to produce a significant, public and freely accessible data-set. Such a system would monitor P2P traffic on a long-term basis and make it available in near real-time, allowing the identification of trends and the revisiting of data points by researchers using different methodologies. Access to the data should also be simplified as far as possible to encourage the use of this data set by researchers from non-computing backgrounds and in particular sociology, psychology, economics and law.

The remainder of this paper is structured as follows: Section 2 provides a brief classification of P2P monitoring methodologies. Section 3 surveys the state-of-the-art in P2P monitoring technologies and studies. Section 4 discusses the limitations of current P2P approaches. Section 5 presents the design of an Open P2P Tracing System. Section 6 describes the initial evaluation of this system. Section 7 discusses how the open tracing system has been exploited to perform research. Finally section 8 discusses avenues for future work and concludes.

2. P2P Tracing Methodologies

Empirical studies of P2P systems may be classified as using one of three broad tracing methodologies: network-level tracing, passive application-level tracing or active application-level tracing [5].

Network-level traces are performed by deploying code on core or gateway network infrastructure and performing IP-level packet monitoring. Network-level tracing is transparent to the P2P network, however, this approach introduces local bias,

which results from deployment location and accurate identification of P2P traffic can be highly problematic.

Passive application-level traces are performed by monitoring the messages passed at the application level. In modern decentralized file-sharing systems all peers participate in message passing and therefore passive monitoring can be achieved simply by modifying a peer to log the messages that it is required to route. Passive application-level tracing is transparent and may be performed without access to core network infrastructure, though the rate at which data can be gathered using this methodology is significantly lower than that of network-level tracing.

Active application-level traces address the scalability shortcomings of passive application-level tracing by employing an aggressive querying and connection policy wherein the monitoring peer attempts to reconnect to and interrogate as much of the application-level network as possible; 'crawling' the P2P network in order to maximize the size and typicality of trace data. While this approach improves the quality of trace-data and the speed at which it is acquired, it does so at the expense of transparency due to the disruptive effect of repeated reconnections and high message generation on the P2P system being monitored.

Section 3 discusses significant empirical studies of P2P file sharing networks, organized according to the tracing methodology used. The findings of these studies are summarized along with their shortcomings.

3. Empirical Studies of P2P File Sharing systems

This section presents significant P2P traffic monitoring studies belonging to each of the tracing methodology classes introduced in section 2, spanning the period from 2000 to 2008. The specific methodology of each study is described alongside its significant findings. Based upon this survey, the benefits and limitations of each class of monitoring approach are discussed along with the general limitations of current P2P studies. While it is impossible to perform an exhaustive study in a single paper, this survey covers the most significant and oft cited studies of P2P networks.

3.1. Network-Level Monitoring

The first network-level study of P2P traffic was performed by Plonka et al. [6]. This study analyzed the bandwidth consumed by Napster [1] on the University of Wisconsin-Madison network during March 2000. A seven hour trace was gathered using a specially developed tool called FlowScan to monitor Napster traffic. FlowScan first identified nodes communicating with the napster.com servers as potential P2P participants and then applied simple heuristics to the node's incoming and outgoing traffic in order to identify Napster-related traffic. The Plonka study found that as early as 2000, P2P applications generated a comparable volume of traffic to the web at 23.1% of total bandwidth, compared to 20.9% for web traffic. Unfortunately, it is difficult to assess the accuracy of this study due to the lack of published details regarding the FlowScan traffic-catagorisation system. However, the short duration of the trace is likely to have resulted in inaccuracy, particularly as other studies have found significant time-of-day variations [10]. Nevertheless, the Plonka study was useful in highlighting the increasing bandwidth consumption observed during the early days of P2P applications.

Plonka's observations on the growing volume of traffic being generated by P2P applications were corroborated by in June 2002 by a University of Washington study conducted by Saroiu et al [2] Their nine day trace found that P2P traffic consumed 43% of campus bandwidth, compared to just 14% for web traffic - a significant increase since the Plonka study. The Saroiu study identified traffic generated by the two dominant P2P systems of the day; Gnutella 0.4 [22] and Kazaa [13] based upon common port usage. In addition to raw traffic data, the Saroiu study reported more fine-grained information about the P2P work-load. This included the finding that, on average, objects retrieved from P2P networks were three orders of magnitude larger than objects retrieved from the web along with the finding that a small subset of peers are responsible for the majority of P2P traffic - a finding that corroborates the results obtained by Adar et al [7] in their passive application-level study (see section 3.2).

Gummadi et al. continued P2P monitoring work at the University of Washington with a 200-day trace of Kazaa traffic in 2003 [3]. This was recorded using a similar methodology to the 2002 trace, except that traffic was identified based upon Kazaa-specific HTTP headers rather than by port use. Uniquely Gummadi's 2003 trace was long enough to observe seasonal variations in P2P traffic and the effect of changing network policies on P2P workloads. Using this trace, Gummadi developed a detailed parameterized model of P2P workloads, which can be used by developers to generate realistic evaluation data.

Accurate identification of P2P traffic is a vital component of network-level P2P monitoring. In the case of the Plonka trace [6], identification was simplified by Napster's semi-centralized architecture [1], while the Saroiu [10] and Gummadi [3] trace identified traffic by port number and header data respectively. However, recent research [20] has demonstrated that users are increasingly moving to P2P systems that aim to avoid monitoring through the use of non-standard ports and encrypted header data. To address this issue, Subhabrata et al. [23] have developed a system for real-time network-level identification of P2P traffic. This system was implemented as an extension to the AT&T's Gigascope [24] high speed traffic monitor. Subhabrata et al. evaluated their traffic identification approach using a 24 hour week-day trace and an 18 hour weekend trace gathered in November 2003 on a major internet backbone. This was augmented with a 6 day trace of traffic on a VPN where network administrators attempt to block P2P traffic, also conducted in November 2003. Subhabrata's approach proved capable of identifying traffic from today's popular P2P systems in real-time for traffic flows of up to 1Gbps while maintaining misidentification rates of less than 5%. While the trace data gathered for this study was used to evaluate their traffic monitoring approach, the authors did not attempt to further characterize or examine the P2P traffic that they observed. The extended version of Gigascope used in this

study is capable of identifying traffic from Gnutella [12], Fasttrack [4], eDonkey [13], Direct Connect [14] and Bittorrent [16]. Subhabrata's identification approach is based upon the flexible concept of *application signatures*, which can be used to categorize traffic using a wide range of metrics.

The growing use of public and application independent anonymization services such as Tor [25] provides an interesting new target for network-level monitoring. Tor is itself a P2P system, which uses an overlay network of routers to enable anonymous outgoing and incoming connections. Any traffic sent via Tor is forwarded from peer to peer on the Tor overlay, ultimately reaching an exit peer, at which point the packet is forwarded on to its original destination. Viewed from the destination, the traffic appears to originate at the Tor exit node, thus protecting client user's identity. Tor also allows nodes participating in the overlay to provide services which may be accessed as though they are hosted at the exit node, allowing for the anonymous hosting of internet services. As Tor is effectively application independent, it is possible to implement network-level monitoring by hosting an exit peer and monitoring the outbound and inbound traffic. Standard network-level monitoring tools may be used for this purpose just as they would on a network gateway; however, monitoring Tor traffic has two significant advantages over monitoring at gateway locations. Firstly, as Tor is accessible world-wide, monitoring Tor exit peers are unlikely to exhibit geographic bias. Secondly, no special access to network infrastructure is required.

The first study of Tor was performed by Bauer et al. at the university of Colorado, Boulder in 2007 [26]. Bauer analyzed Tor's vulnerability to attack and discovered an inverse relationship between the degree of optimization in the Tor overlay and its resilience against attack. In 2008, McCoy et al. extended Boulder's work by performing a detailed characterization of Tor traffic [27] using the methodology outlined above. The study first provided a breakdown of Tor traffic by type, finding that web and Bittorrent data makes up the majority of traffic. The study also analyzed the location of Tor users, finding a significant geographic bias. McCoy also found that a significant level of Tor misuse occurs, for example snooping on plain-text data such as POP email traffic. Loesing et al. performed a detailed performance measurement of Tor traffic in 2008, analyzing the QoS properties and specifically the latency of Tor. Based upon this detailed analysis, performance optimisations to the Tor protocol were suggested [35].

In each of the studies discussed above, network-level tracing was used to record the low-level characteristics of P2P traffic flows on private networks. Network-level tracing is potentially transparent, scalable and allows comparison of traffic from multiple domains side-by-side. However, with the exception of Tor monitoring, this approach is dependent upon access to core network infrastructure, which is not always feasible. While researchers may have access to gateway infrastructure on large private networks, such as academic networks, data obtained from such sources should be viewed as potentially biased due to differences between the characteristics of the private network's users and general Internet users.

3.2. Passive Application-Level Monitoring

The first passive application-level trace of a P2P system was performed by Adar and Huberman in 2000 on the Gnutella 0.4 network [7]. This 24-hour trace logged resource-discovery traffic which was then used to assess the prevalence and characteristics of a problem known as 'free riding', wherein users download resources from, but do not upload resources to a P2P file-sharing system. The Adar trace was performed by modifying the open-source 'Furi' Gnutella client (no longer available) to monitor search, response and peer discovery messages. Adar and Huberman discovered that participation in Gnutella was highly asymmetrical with 66% of peers sharing no files at all and almost 50% of all files being served by the top 1% of hosts. This finding was significant as it contradicted the (then) conventional wisdom that user participation in P2P file sharing systems is symmetrical. Adar's result was later corroborated by Saroiu's 2002 network-level study [10].

Hughes et al [4] revisited the results of the Adar trace in 2004 on the Gnutella 0.6 network [12] based upon a one week trace. The trace was performed using a specially developed monitoring tool based on the Jtella base classes [17]. The monitoring peer connected to the Gnutella network as an Ultrapeer [12] and periodically reconnected in order to maximize the size and typicality of its sample-base. Hughes discovered that in the four years since the Adar study, the proportion of free-riders had increased from 66% to 85%, while corroborating Adar's finding that the top 1% of hosts serve almost 50% of all files. Hughes speculated that the increase in free riding may be the result of an increase in prosecution of copyright infringement. Hughes et al. revisited this data point using their 2005 trace and found that the level of free riding on Gnutella had continued to increase - to over 95% [28]. This trace was later used to assess the level of illegal pornographic material being distributed on the Gnutella network [8]. The study found that an average of 1.6% of searches and 2.4% of responses contained references to illegal pornography, though this material is distributed by a tiny subset of peers that typically share nothing else. This result was subsequently refined in 2008, looking only at the level of traffic relating to child abuse media. This study found that more than 1% of search traffic and 1.6% of search-response traffic relates to child abuse related media.

In each of the cases discussed above, passive application-level monitoring is used to study application level properties in an Internet-wide context. Like network-level monitoring, passive application-level monitoring is transparent, however, it does not require access to low-level network infrastructure. Unfortunately, in cases where a very large sample of network traffic is required quickly, passive monitoring would be unsuitable due to the small-world properties of modern P2P networks.

3.3. Active Application-Level Monitoring

Ripneau and Foster [9] performed the first active application-level trace of the Gnutella network from November 2000 to May 2001. This study attempted to map the Gnutella network in terms of the average number of links between hosts and the number of hops that these links represent on the underlying IP network. To achieve this, a specialized Gnutella peer known as

a '*crawler*' was developed. The crawler connects via the normal Gnutella boot-strapping system and uses Gnutella's peer-discovery mechanism [12] to find new peers. The IP address of these peers is added to the list of those observed and the crawler attempts reconnection in a new location, repeating the process and gradually building a 'map' of the network. The resulting map includes the total number of nodes, the total number of links and average traffic data. Based upon the findings of this study, Ripneau concluded that the emergent structure of the Gnutella network was such that the network's bandwidth consumption would limit its scalability, as predicted by Ritter [29]. Unfortunately, Ripneau's crawling approach is invasive, as repeated reconnection affects the P2P network. It is also un-scalable due to the computational and network expense incurred when crawling the application-level network.

Saroui et al. [10] extended Washington University's work on monitoring P2P systems to the application level with a one month crawl of the Gnutella network in May 2001. The crawler used a similar methodology to Ripneau and observed between 8,000 and 10,000 unique peers, which at that time would have accounted for between 25% and 50% of the Gnutella network. The 2001 Saroiu trace recorded low-level data, including each peer's IP address, latency and bottleneck bandwidth between peers; along with higher level data including each peers advertised bandwidth and the number and size of files being shared. These high-level properties were measured by logging Gnutella's resource discovery and network maintenance messages, while bottleneck bandwidth was measured using SProbe [30], a network tool that uses a TCP exploit to accurately measure bottleneck bandwidth without the need for remote cooperation.

Chu et al [11] performed the first study that attempted to quantify the availability of peers and files on the Gnutella network using a forty day trace performed in early 2002. This trace was gathered by a tool based upon the Jtella API [17] that followed a similar methodology to the Ripneau crawler [9]. Search-response messages were intercepted by the crawler and unique peers were identified based upon their advertised IP and port pairs. The crawler was used to gather a list of 20,000 unique peers using the 'BearShare' and 'SwapNut' clients, at which point a second program, known as the 'tracking manager' attempted to download each peer's file-list using proprietary BearShare and SwapNut extensions. Using this methodology, the availability of peers and files was monitored for a period of 40 days beginning on March 28th. Chu reported a strong correlation between time-of-day and node availability and proposed a model to describe peer availability. Additionally, Chu provided a breakdown of relative file-type popularity and corroborated the finding of Saroiu [10], that file popularity is highly skewed with the top 10% of files accounting for more than 50% of shared data. A clear limitation of Chu's study lies in the use of proprietary extensions to obtain file lists, which limits the size of the trace and introduces possible bias due to the limited user-group studied.

Bittorrent is peculiar amongst P2P file sharing systems in that it does not implement a resource discovery mechanism. Thus, passive application-level monitoring is rendered ineffective due to the lack of resource discovery traffic. For this reason, Bittorrent studies generally actively query Bittorrent 'trackers'; specialized peers which mediate connection to the file distribution overlay, joining this overlay to participate in file distribution if further data is required. The first Bittorrent study was performed by Izal et al. in 2004 [31] shortly after Bittorrent's release and analyzed the life-span of a files being distributed using Bittorrent over a five month period. Izal showed that Bittorrent was a highly effective distribution mechanism for popular media effectively addressing the problem of 'flash crowds'. In 2005 Thommes et al. [32] further examined the 'fairness' of the Bittorrent protocol. The study found that Bittorrent performs near-optimally in terms of uplink bandwidth utilization, but that low bandwidth peers frequently downloaded significantly more than they uploaded. While the differential experience of Bittorrent peers may be considered unfairness, as it was by the authors of this paper, it may also be responsible for Bittorrent's high level of popularity; as the protocol is capable of catering to the needs of users on both high bandwidth and slow connections.

In each of the cases discussed above, active application-level monitoring has been used to study P2P traffic properties in an Internet-wide context, where a very large and typical body of trace data was required (e.g. mapping the Gnutella network). Active application-level monitoring is easy to deploy and should not contain local bias; however, the aggressive reconnection and interrogation approach employed makes this approach invasive and limits its scalability. Due to the invasiveness of this approach, active application-level monitoring may be easily detected and due in part to extensive copyright enforcement activities, file sharing user communities actively search for and attempt to circumvent active monitoring approaches.

3.4. Summary of Monitoring Approaches

This paper introduced a classification scheme for empirical studies of P2P file sharing systems based upon the tracing methodology that they employ: network-level monitoring, passive application-level monitoring or active application-level monitoring. In the context of this classification, significant empirical studies were reviewed along with the benefits and drawbacks of each approach. These are summarized below:

Network-level monitoring is transparent to the network and highly scalable. It is capable of comparing traffic flows from multiple P2P systems side-by-side and is well suited to characterizing P2P traffic on large private networks; however, it is poorly suited for performing global monitoring of P2P systems due to the possibility of local bias. Moreover, network-level monitoring requires low-level access to core network infrastructure, which is often unfeasible. Examples of network-level monitoring studies include [2] and [6].

Passive application-level monitoring is also scalable and transparent to the network. It can be performed without access to core network infrastructure, though it does not provide as large a volume of trace data as network-level monitoring or crawler-based application-level monitoring. Furthermore, it is inherently protocol specific. Passive application-level monitoring is thus best suited to instances where network-level monitoring is impossible or where a non-invasive approach is

desirable. Examples of passive application-level monitoring studies include [7] and [8].

Active application-level monitoring is less transparent and scalable than either network-level or passive application-level monitoring; however, it allows large volumes of trace data to be gathered without low-level access to the network infrastructure. It is thus the best approach where global network information is required and access to the underlying network infrastructure is not possible. Examples of active application-level monitoring studies include [10] and [11].

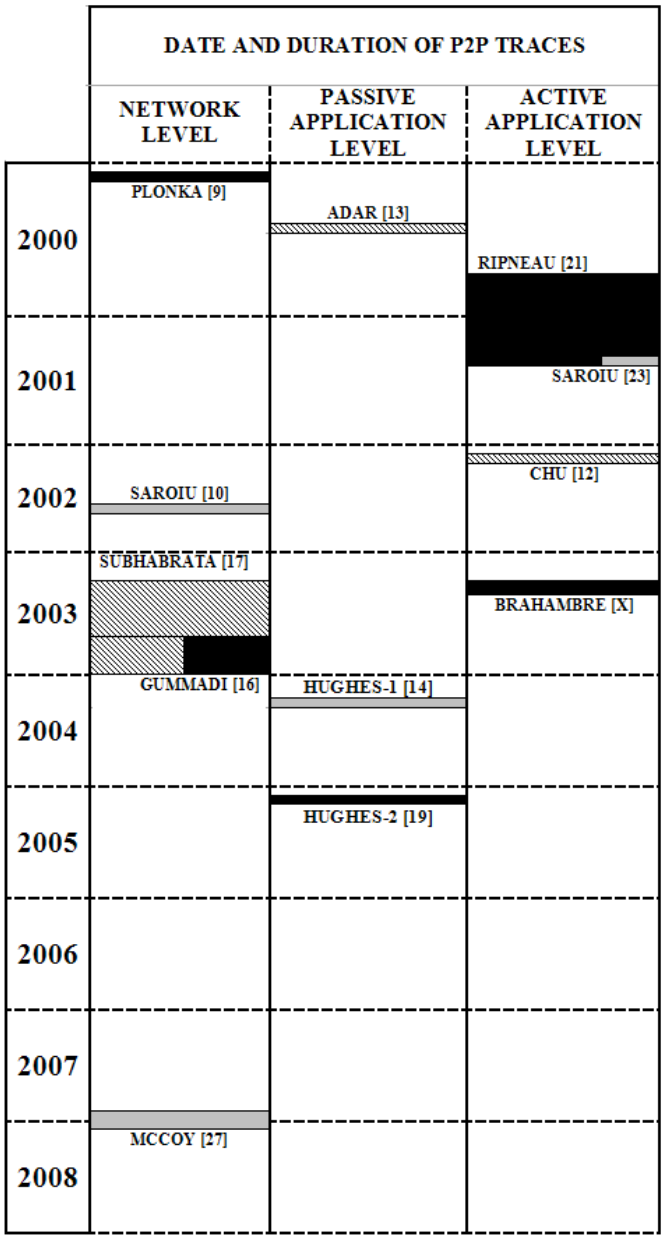

Figure 1. Time Distribution of P2P Traces

P2P traces such as those presented in this paper have proven invaluable in informing research in the field of P2P systems, however, each of these studies provides only a piece of the puzzle; describing a subset of P2P traffic characteristics for a subset of protocols over the duration of the trace. Often, papers which cite these studies fail to adequately consider such limitations. For example, the data-point provided by Adar's 2000 study of free riding [7] has been used in a significant body of research until the present day, however, when this study was revisited by Hughes et al. [4] in 2005, it was discovered that free riding had increased, revealing a significant, and (until that point), unidentified trend.

Figure 1 illustrates the date and duration of each of the P2P traces discussed in this paper. As figure 1 illustrates, few of the P2P studies presented in this paper are of sufficient duration to identify trends in P2P traffic, rather they simply provide a data-point for the monitored characteristics. The notable exceptions to this are Gummadi's 2003 Kazaa trace [3] which was long enough to observe seasonal variations and Hughes' 2005 study of free-riding [5] which, by revisiting Adar's 2000 experiment [7] was able to show an intervening trend in user behavior.

4. Limitations of Existing Work

There are a number of significant shortcomings in the current body of research on P2P traffic monitoring. The first and perhaps most significant of which is the wide-spread use of closed data sets. As can be seen from Figure 1, P2P studies may require weeks or even months of P2P traffic data. While it is understandable that after investing significant time and effort in gathering a data set, researchers may be reluctant to make this data public, this prevents the findings of studies being verified using different methodologies and prevents trace data being revisited in new contexts.

Another significant gap exists in the body of work on P2P monitoring regarding the identification of underlying trends. For example, the data-point provided by Adar's 2000 study of free riding [7] was revisited by Hughes in 2004 [5] and 2005 [28], each time revealing a significantly different data point. It may be is possible that other equally significant trends might be discovered by revisiting past studies. For example, would the growing popularity of digital video be reflected by an increase in the availability of such files since Chu's [11] 2002 study of file availability? Despite the possibility of exposing significant trends in user behaviour, few studies choose to revisit earlier data-points.

Most empirical studies of P2P file sharing systems are concerned only with the technical characteristics of P2P traffic (files shared, bandwidth usage etc.). While this information is critical for simulation of P2P traffic and for the development of approaches to encouraging positive user behaviour, the next step, reasoning about the social and psychological factors which produce this behaviour, is rarely taken. Furthermore, most studies do not take into account the real-world factors which may affect P2P traffic. Notable exceptions to this are the studies by Adar [7] and Hughes [5] [8] [28] [33], that explicitly consider the social factors which are responsible for observed behaviour.

5. Design of an Open P2P Tracing System

This paper has made the case that an open, easy to access and long-term P2P trace is required to improve our understanding of P2P file sharing systems. This section now discusses the design and implementation of such a system: The Open P2P Tracing System. As previously described, the system will use a passive application-level tracing methodology [5] to gather data. The implementation of this functionality will now be described.

5.1. Tracing Functionality

Implementation of tracing functionality is dependent upon the P2P system being monitored. As the Open Tracing System aims to provide a widely reusable data set, we intend to monitor several of today's most popular P2P systems, including Gnutella [12], Fasttrack [13], eDonkey [14], DirectConnect [15] and Bittorrent [16]. In order to minimize the time required to port monitoring code to additional P2P networks we implement logging functionality by modifying existing open source clients available for each P2P network. Analysis of such clients, which include Jtella [17], Open DirectConnect [18] and Azureus [19] revealed that each shared elements of common structure. Of particular significance in terms of implementing tracing support was that each client implements a single routing component which is used to process incoming and outgoing messages. It is into this routing component that we insert monitoring code. This is shown in Figure 2.

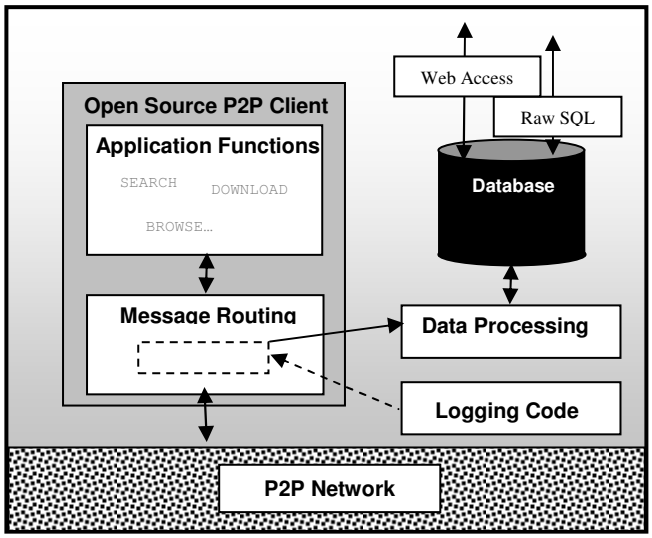

Figure 2. System Architecture

In order to ensure that sufficient data is gathered, the system is capable of maintaining a large number of network connections, for example by connecting as an Ultrapeer when monitoring Gnutella. Furthermore, in order to ensure data is representative, the system periodically re-connects to different areas of the P2P network.

5.2. Maintaining User Anonymity

Open publication of IP addresses and other identifying data is ethically dubious and would likely have a number of undesirable effects. Furthermore, studies have suggested that P2P users are migrating to those file sharing systems which are more difficult to monitor [20]. It is therefore likely that publication of user data from one P2P system would drive users to other, unmonitored systems or perhaps even result in the P2P community attempting to exclude the tracing client. Recent research [20] has also suggested that the level of perceived anonymity offered by P2P networks has a significant effect on user behaviour. This implies that the publication of IP addresses might cause a significant 'observer effect'.

While maintaining anonymity is desirable, a globally unique user identifier (GUID) is often required to accurately track the behaviour of users over time. For this reason, as data is gathered, all IP addresses and user-names are switched for a randomly assigned GUID. Any additional information encapsulated in the original identifier, such as country and service provider, is resolved and stored separately in the database.

Replacing real world identifiers with a randomly assigned but consistent GUID prevents third parties from associating trace data with individuals. However, in the long term this method would lead to the accumulation of data on millions of P2P users, which gives rise to significant security implications. We have therefore arrived at a compromise solution, wherein we only attempt to ensure that GUIDs remain unique during a typical period of connection (session), after which time the IP/GUID mapping is discarded and, if that peer is observed again, it will be assigned a new GUID.

This compromise between maintaining anonymity and user tracking is evaluated in section 6.

5.3. Data Collection and Storage

Due to the scalability problems associated with resource discovery on decentralized P2P networks, P2P systems have increasingly moved towards Super-node architectures such as the architecture used in Kazaa [13] or the Gnutella 0.6 ultra-peer scheme [12]. Concurrently, the scalability problems which arise from the use of a single indexing server have prompted centralised systems to move towards more decentralized architectures that utilize user-hosted indexing servers as demonstrated by DirectConnect and eDonkey. In both cases, the presence of peers on the application-level network which are responsible for routing a greater proportion of messages facilitates application-level monitoring. By connecting to the network as a Gnutella 'ultra-peer', a Direct Connect 'hub' or eDonkey 'server', a greater proportion of traffic can be captured using passive application-level monitoring.

The Bittorrent network is a special case. As Bittorrent does not support resource discovery, torrents are indexed on publicly available trackers, which are accessible on the web. Thus tracing of Bittorrent may be achieved by querying the trackers for details of users currently participating in each per-file

overlay. An overview of this tracing methodology is described in more detail in [36].

As we intend that tracing data should be made accessible to a broad audience, we will use a standard MySQL database for data storage. As SQL is currently the most popular database technology for online applications we hope this will maximize the accessibility of the system. A separate SQL database is maintained for each P2P system being monitored and each of these databases contains per-message tables. Each message that is stored in the database is time-stamped, facilitating the retrieval of data for a specific instant or time-period. In order to maintain flexibility, the system also logs all message types as it is difficult to predict in advance what data may be of interest to other researchers

5.4. Data Access and Presentation

Alongside raw SQL access, we also provide a web-based method of data access for interested parties. We hope this will allow the system to support a range of users with diverse requirements. We envision that three classes of user will make use of the system: i) corporate users, ii) computing researchers and iii) non-computing researchers.

Corporate users of the system might include P2P developers, who could use the system to assess the market penetration of their P2P products, and the music and film industry that might use the system to assess the extent to which their products were being distributed on P2P systems. To facilitate access for corporate users in particular, the system supports on-the-fly generation of common graphs illustrating both current and historic data based on a number of criteria including: P2P client popularity, file popularity and availability, level of user participation and free-riding. The system is also capable of exporting this same data in common formats such as comma separated value (CSV) files and Excel (XLS) spreadsheet documents. To further facilitate the association of P2P traffic with real-world factors, graphical data is annotated with news articles containing references to P2P, which are culled from RSS feeds. This functionality may be used to answer questions such as whether high-profile copyright prosecutions increase levels of free-riding, or whether news about a specific P2P client affected its level of use.

Computing researchers are most likely to be interested in accessing raw traffic data provided by the system. This is possible through direct access to the SQL database which allows more versatility in interrogation than hard-coded trend data that the system provides.

Non-computing researchers are supported by the systems ability to export traffic data in CSV and XLS formats, which can both be accessed using common office software. It is also possible that 'casual' Internet users may find this data of interest, though the requirements of these users have not been explicitly considered in the design of the system. The web interface is shown in Figure 3.

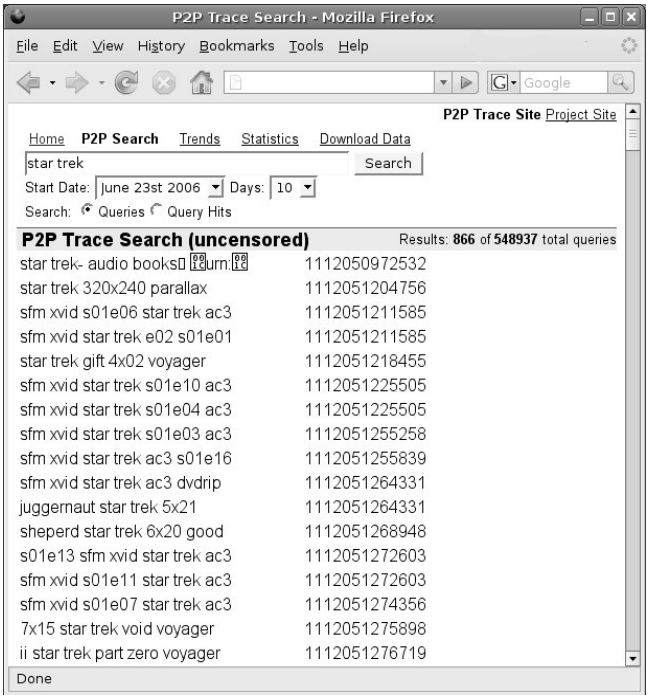

Figure 3. Web Interface of the Open Tracing System

5.5. Implementation Status and Access

The current implementation of the Open Tracing System focuses on the tracing of the Gnutella network, the results of which are being used as a basis to evaluate system functionality (as will be discussed in section 6). Adding support for tracing additional networks is being implemented in parallel to this.

The system is currently at a pre-alpha stage and therefore access to it must currently be arranged through the authors of this paper. However, we are actively looking for case studies, such as those described in section 7, which we hope will guide system development. We anticipate that, in due course, the open P2P tracing system will be made freely accessible online.

6. Initial evaluation results

We have begun analyzing the performance of the Open P2P Tracing System in terms of its network, computational and storage requirements. The system is hosted and evaluated on a 2.8GHz Intel P4 with 512MB RAM and a 100GB hard drive connected to the Internet via a high-speed academic network.

In order to minimize invasiveness during evaluation, the modified tracing peer maintains a single ultra-peer connection and allows unlimited incoming leaf-node connections. As previously described, in order to ensure the typicality of our trace, the system periodically reconnects to the network at an interval of six hours. This figure was derived empirically – we found that a shorter time resulted in connections to less stable peers and less volumes of data, whilst a longer time introduced local effects (e.g. the sharing preferences of core peers) that could impact on the data.

6.1. Networking Requirements

The local network requirements of tracing Gnutella have been assessed through experimentation, while gathering trace data. This reveals that the system consumes an average bandwidth of 98kbps as a result of routing resource discovery messages and an additional 9kbps due to routing control messages, which is commensurate with results obtained elsewhere [7]. The networking requirements of passive application level tracing can easily be met by our available networking infrastructure.

6.2. Storage Requirements

The storage requirements of our tracing methodology were assessed during the gathering of a single-connection Ultrapeer trace of the Gnutella network, conducted over a period of one month. Experimental results are shown in Figure 4.

The storage requirements of tracing the Gnutella network using MySQL's standard data compression range from a minimum of 40MB per day to a maximum of 95MB per day. While this makes long-term tracing feasible using standard desktop storage hardware, available storage capacity still forms the bottleneck in our tracing capability and for this reason, only one tracing connection per monitored network will be maintained by the Open Tracing System for the immediate future.

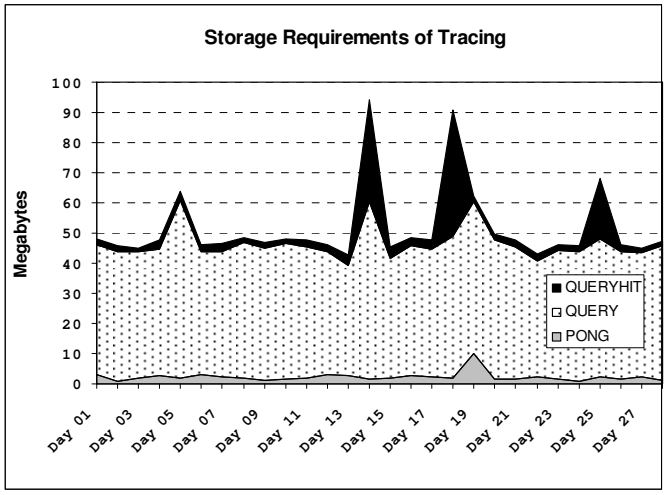

Figure 4. Storage Requirements of Tracing

6.3. Anonymization

As previously discussed, the anonymization approach used is a compromise between storing large volumes of user records and providing a consistent GUID to support session tracking. During our month long trace of the Gnutella network, we performed a number of experiments to determine an optimal IP discard time.

We first monitored session lengths across our trace and found that more than half lasted less than one hour and that more than 80% less than two hours, this is commensurate with results obtained elsewhere [10]. Figure 5 shows the relationship between IP discard time and the percentage of sessions where any data would have been lost. The 'long tail' of the graph shown in Figure 5 is due to the presence of a small number of highly available peers with server-like characteristics and implies that total session coverage would require an unfeasibly long ID-discard period, in turn leading to the maintenance of very large numbers of IP addresses.

Figure 6 explores the relationship between discard time and the number of IP addresses stored by the system. The graph shows that the number of stored IP's varies significantly over the period of our trace and based upon the discard time used. Based upon these results, we have selected a discard time of 6 hours. This period successfully captures 93% of sessions as shown in Figure 6 and results in the open tracing system storing an average of fewer than 800 IP addresses at any one time as shown in Figure 6.

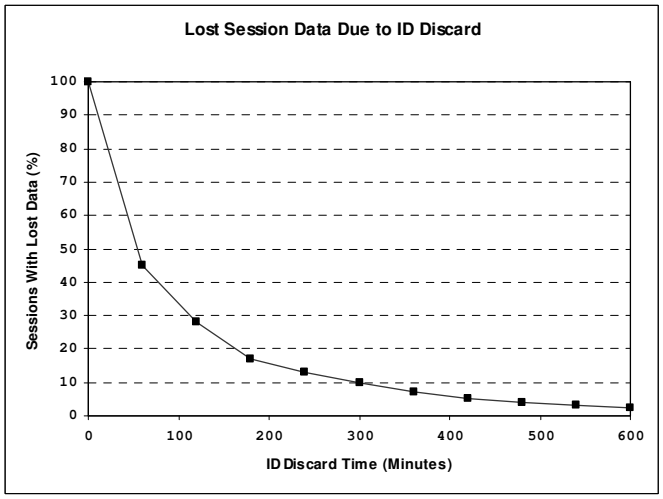

Figure 5. Effect of ID Discard Period on Lost Session Data

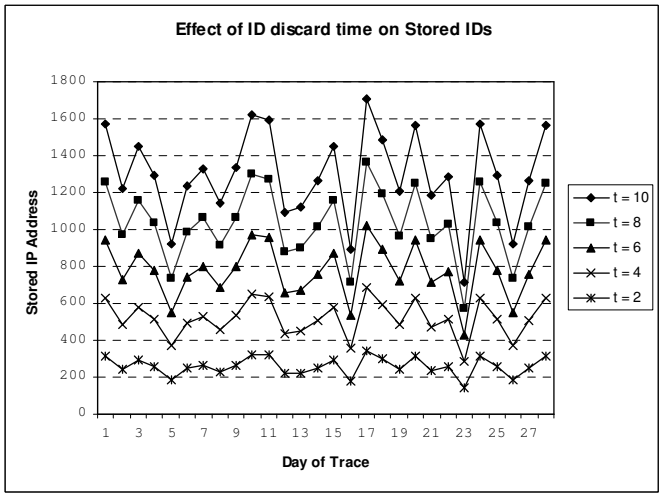

Figure 6. Effect of Discard-Period on Number of Stored IPs

7. Exploiting Open Tracing

In order to better illustrate how an open P2P tracing system may be used to expand our understanding of how P2P file sharing systems are used, this section provides a detailed summary of studies that have been performed by Lancaster University using the same passive application-level tracing methodology as used in the open tracing system.

"Free Riding on Gnutella Revisited: the Bell Tolls?" revisited Adar's 2001 study of Gnutella traffic. Adar found that (i) over two thirds of Gnutella users download files while not uploading – effectively free riding and (ii) user contribution was highly asymmetric with the top contributors providing a disproportionate share of files. Our 2005 study found that in the intervening four years, this situation had worsened. While we found similarly asymmetric contribution levels, the number of contributing peers had fallen by more than half. The stark difference between these data points shows the benefit of revisiting established data points, a key goal of the open tracing system.

"Is Deviant Behaviour the Norm on P2P File Sharing Networks?" was performed in 2005 in collaboration with psychologists, to answer the question of whether the perception of anonymity and lack of censorship on Gnutella encourages the distribution of illegal pornographic material. Our study found that while the level of resource discovery traffic relating to illegal pornographic material was high (1.6% of searches and 2.4% of responses), that this traffic was generated by a small, yet active subset of the network that shared nothing else. This finding has useful implications for the policing of P2P file sharing networks, but more importantly shows how collaboration between academic disciplines, another key goal of the open tracing system, can lead to better understanding of P2P user behaviour.

"Supporting Law Enforcement in Digital Communities through Natural Language Analysis" revisited our 2005 trace data, looking specifically at the level of traffic related to child abuse material. We found that 1% of search traffic related to such material and 1.6% of response traffic. Building upon this analysis in collaboration with linguistics researchers and child protection researchers and professionals, this paper suggested an approach to automating the policing of P2P file sharing systems for child abuse material. As with the previous study this illustrates the significant benefits of making P2P file sharing trace data to other disciplines.

8. Future Work and Conlusions

This paper has highlighted significant shortcomings in the existing body of work on P2P monitoring, and described the implementation of a large-scale, open and ongoing trace that can be freely accessed by researchers from diverse backgrounds. Based upon an extensive review of existing P2P studies, we have selected a non-invasive tracing methodology that we will incrementally apply to five of today's most popular P2P file sharing networks. At the current time, tracing functionality has been implemented for the Gnutella network and evaluation of the system shows that our methodology is capable of gathering, anonymizing and logging Gnutella traffic in real-time using standard desktop hardware. The system facilitates access for users from diverse backgrounds- a direct interface to the SQL database allows versatile access for computing researchers, while a simplified web interface and on-the-fly computation of common P2P characteristics such as the level of 'free riding' and relative file-type popularity facilitate access for those from non-computing fields.

In the short term, future work will focus on the implementation of tracing functionality for additional P2P systems. In the longer term we intend to investigate incorporating Natural Language Processing mechanisms into the system to allow the user to perform more sophisticated analyses. In addition to this we will also examine the feasibility of using technologies such as Aspect Oriented Programming to assist in the non-invasive monitoring of P2P systems, and also to investigate alternative, more scalable data storage solutions.

In parallel to extending tracing support, we intend to evaluate the usefulness of the system as a tool, using a number of case studies. Part of this will include working with psychology researchers to investigate the process of group formation in P2P communities. This will build upon our previous work [5] [8] [28] [33] and allow us to explore the extent to which the system can support inter-disciplinary research.

9. References

[1] Merriden T., Irresistible Forces: the Business Legacy of Napster and the Growth of the Underground Internet, Capstone Publishing, 2001, pages 1-11.

[2] Saroiu S., Gummadi K., Dunn R. J., Gribble S. D., Levy H. M., An Analysis of Internet Content Delivery Systems, in the proceedings of the 5th International Symposium on Operating Systems Design and Implementation (OSDI), San Francisco, CA, 2004, pages 315-327.

[3] Gummai K., Dunn R. J., Saroiu S., Gribble S. D., Levy H. M., Zahorjan J., Measurement, Modeling and Analysis of a P2P File-Sharing Workload, in the proceedings of the 19th Symposium on Operating Systems Principles (SOSP'03), Bolton Landing, NY, 2003, pages 314-329.

[4] Hughes D., Coulson G., Walkerdine J., Free Riding on Gnutella Revisited: the Bell Tolls?, in IEEE Distributed Systems Online, volume 6, number 6, 2005. sdl2.computer.org/comp/mags/ds/2005/06/o6001.pdf

[5] Hughes D, Walkerdine J., Lee K., Monitoring Challenges and Approaches for P2P File Sharing Systems, in the proceedings of the 1st International Conference on Internet Surveillance and Protection (ICISP'06), Cap Esterel, France, 2006, page 18-18.

[6] Plonka D., Napster Traffic Measurement, University of Wisconsin-Madison, 2000, available online at: http://net.doit.wisc.edu/data/Napster

[7] Adar E., Huberman B., Free Riding on Gnutella, First Monday, volume 5, number 10, October 2000, available online at: www.firstmonday.org/issues/issue5_10/adar/

[8] Hughes D., Gibson S., Walkerdine J., Coulson G., Is Deviant Behaviour the Norm on P2P File Sharing Networks?, in IEEE Distributed Systems Online, volume

7, number 2, 2006. csdl.computer.org/comp/mags/ds/2006/02/o2001.pdf

[9] Ripeanu M., Iamnitchi A., Foster I., Mapping the Gnutella network, published in IEEE Internet Computing., volume 6, number 1, 2002, pages 50-57.

[10] Saroiu S., Gummadi K., Gribble S. D., Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts, published in Springer-Verlag Multimedia Systems volume 9, number 2, 2003, pages 170-184.

[11] Chu J., Labonte K., Levine N., Availability and locality measurements of peer-to-peer file systems, in the proceedings of ITCom: Scalability and Traffic Control in IP Networks, Proceedings of SPIE, volume 4868, Boston, MA, 2002, pages 310-321.

[12] The Gnutella Protocol Specification v 0.6, available online at: rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[13] Kazaa homepage, available online at: www.kazaa.com

[14] eDonkey homepage, available online at: www.edonkey2000.com

[15] Direct Connect homepage, available online at dcplusplus.sourceforge.net

[16] Cohen B., Incentives Build Robustness in Bittorrent, available online at: http://www.bittorrent.com/bittorrentecon.pdf, May, 2003.

[17] Jtella homepage, available online at: jtella.sourceforge.net

[18] Open DirectConnect homepage, available online at: sourceforge.net/projects/odc/

[19] Azureus home page, available online at: azureus.sourceforge.net/

[20] Karagiannis, T., Broido, A., Brownlee, N., Faloutsos, M., Is P2P Dying or Just Hiding?, In the Proceedings of Globecom'04, Dallas, TX, 2004, pages 1532-1538.

[21] Qiao Y., Bustamante F.E., Structured and Unstructured Overlays Under the Microscope - A Measurement-based View of Two P2P Systems That People Use, published in the Proceeding of the USENIX Annual Technical Conference, Boston, MA, 2006, pages 10-10.

[22] The Gnutella Protocol Specification v0.4 (Document Revision 1.2), available online at: http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, 2001.

[23] Subhabrata S., Spatscheck O., Wang D., Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures, in the proceedings of the thirteenth international world wide web conference (WWW2004), New York, NY, 2004, pages 512-521.

[24] AT&T Gigascope homepage, available online at: http://public.research.att.com/viewProject.cfm?prjID=129

[25] The Onion Router (Tor) homepage, available online at: http://www.torproject.org/index.html.en

[26] Bauer K., McCoy D., Grunwald D., Kohno T., Sicker D., Low-resource Routing Attacks Against Tor, in the proceedings of the 2007 ACM workshop on Privacy in Electronic Society (WPES'07), Alexandria, VA, 2007, pages 11–20.

[27] McCoy D., Bauer K., Grunwald D., Kohno T., Sicker D., Shining Light in Dark Places: Understanding the Tor Network, in the proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS'08), Leuven, Belgium, 2008, pages 63-76.

[28] Walkerdine J., Hughes D., Lee K., The Effect of Viral Media on Business Usage of P2P, in the proceedings of the 7th international IEEE conference on Peer-to-Peer Systems (P2P'07), Galway, Ireland, 2007, pages 249-250.

[29] Ritter J., Why Gnutella Can't Scale, No Really, available online at: http://www.darkridge.com/~jpr5/doc/gnutella.html.

[30] SProbe homepage, available online at: http://sprobe.cs.washington.edu

[31] Izal M., Urvoy-keller G., Biersack E., Felber P., Al Hamra A., Dissecting BitTorrent: Five months in a torrent's lifetime in the proceedings of the Passive and Active Measurement workshop (PAM'04), Antibes Juan-les-Pins, France, 2004, pages 1-11.

[32] Thommes R., Coates M., BitTorrent Fairness: Analysis and Improvements, in the proceedings of the 4th Workshop on the Internet, Telecommunications and Signal Processing (WITSP'05), Noosa Heads, Australia, 2005, available online at: http://www.tsp.ece.mcgill.ca/Networks/projects/pdf/thommes_WITSP05.pdf

[33] Hughes D., Rayson P., Walkerdine J., Lee K., Greenwood P., Rashid A., May-Chahal C., Brennan M., Supporting Law Enforcement in Digital Communities through Natural Language Analysis, in the proceedings of the 2nd International Workshop on Computational Forensics (IWCF'08). Washington D.C., USA, 2008, pages 122-134.

[34] Piatek M., Kohno T., Krishnamurthy A., Challenges and Directions for Monitoring P2P File Sharing Networks – or– Why My Printer Received a DMCA Takedown Notice, available online at: http://dmca.cs.washington.edu/dmca_hotsec08.pdf

[35] Loesing K., Sandmann W., Wilms C., Wirtz G., Performance Measurements and Statistics of Tor Hidden Services, in the Proceedings of the International Symposium on Applications and the Internet (SAINT), Turku, Finland, July 2008, pages 1-7.

[36] Peer Guardian home page, available online at: http://phoenixlabs.org/pg2/

[37] Hughes D., Lee K., Walkerdine J., An Open Tracing System For P2P File Sharing Systems, published in the proceedings of the second International Workshop on P2P Systems and Applications (P2PSA '07), Morne, Mauritius, May 2007, pages 3-9.

Preliminary 2009 Conference Schedule

http://www.iaria.org/conferences.html

NetWare 2009: June 14-19, 2009 - Athens, Greece

- SENSORCOMM 2009, The Third International Conference on Sensor Technologies and Applications
- SECURWARE 2009, The Third International Conference on Emerging Security Information, Systems and Technologies
- MESH 2009, The Second International Conference on Advances in Mesh Networks
- AFIN 2009, The First International Conference on Advances in Future Internet
- DEPEND 2009, The Second International Conference on Dependability

NexComm 2009: July 19-24, 2009 - Colmar, France

- CTRQ 2009, The Second International Conference on Communication Theory, Reliability, and Quality of Service
- ICDT 2009, The Fourth International Conference on Digital Telecommunications
- SPACOMM 2009, The First International Conference on Advances in Satellite and Space Communications
- MMEDIA 2009, The First International Conferences on Advances in Multimedia

InfoWare 2009: August 25-31, 2009 – Cannes, French Riviera, France

- ICCGI 2009, The Fourth International Multi-Conference on Computing in the Global Information Technology
- ICWMC 2009, The Fifth International Conference on Wireless and Mobile Communications
- INTERNET 2009, The First International Conference on Evolving Internet

SoftNet 2009: September 20-25, 2009 - Porto, Portugal

- ICSEA 2009, The Fourth International Conference on Software Engineering Advances
  - SEDES 2009: Simpósio para Estudantes de Doutoramento em Engenharia de Software
- ICSNC 2009, The Fourth International Conference on Systems and Networks Communications
- CENTRIC 2009, The Second International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services
- VALID 2009, The First International Conference on Advances in System Testing and Validation Lifecycle
- SIMUL 2009, The First International Conference on Advances in System Simulation

NexTech 2009: October 11-16, 2009 - Sliema, Malta

- UBICOMM 2009, The Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies
- ADVCOMP 2009, The Third International Conference on Advanced Engineering Computing and Applications in Sciences
- CENICS 2009, The Second International Conference on Advances in Circuits, Electronics and Micro-electronics
- AP2PS 2009, The First International Conference on Advances in P2P Systems
- EMERGING 2009, The First International Conference on Emerging Network Intelligence
- SEMAPRO 2009, The Third International Conference on Advances in Semantic Processing