Assessing the Design of Web Interoperability Protocols

Jorgina Paihama
Department of Computer Science
University of Cape Town
Private Bag X3, Rondebosh, 7701
Cape Town, South Africa
jpaihama@cs.uct.ac.za

Kyle Williams
Department of Computer Science
University of Cape Town
Private Bag X3, Rondebosh, 7701
Cape Town, South Africa
kwilliams@cs.uct.ac.za

Hussein Suleman
Department of Computer Science
University of Cape Town
Private Bag X3, Rondebosh, 7701
Cape Town, South Africa
hussein@cs.uct.ac.za

ABSTRACT

Existing Web interoperability protocols are, arguably, overly complex as a result of each protocol being designed by a different group, providing a single service, and having its own syntax and vocabulary. Some standards, such as RSS, are popular and are designed with simplicity in mind and include easy to understand documentation, which is a key reason for its high adoption levels. However, the majority of protocols are complex, making them relatively difficult for programmers to understand and implement and thus hampering communication between heterogeneous information systems. This paper proposes a possible new direction for high-level interoperability protocols by focusing on simplicity. The High-level Interoperability Protocol - Common Framework (HIP-CF) was designed and evaluated as a proof of concept that, if interoperability is simplified and it is made easier for programmers to understand and implement protocols, it could lead to having more interoperable systems as well as increased protocol adoption levels. Evaluation showed that this is a reasonable view and that there is a lot of room for improvement when it comes to interoperability protocols.

Categories and Subject Descriptors

H.3.0 [**Information Storage and Retrieval**]: General; H.3.5 [**Online Information Services**]: Data Sharing; H.3.7 [**Digital Libraries**]: Standards

General Terms

Standardisation, Design, Documentation

Keywords

Interoperability, Internet protocols, Web Services

1. INTRODUCTION

A large amount of research has been carried out on the creation and preservation of digital information systems and it has been shown how these systems can be beneficial in a number of environments. However, the benefits of these systems could be extended even further if the systems were able to communicate better. Improved interoperability between heterogeneous systems would allow for increased information accessibility, promote open access, allow for the easier creation of federated metadata archives, improve efficiency and reduce costs by allowing systems to share and retrieve data as necessary [27].

This kind of interoperability is beneficial in many contexts. For instance, interoperability among medical record systems is beneficial as it could allow a doctor, who is treating a patient for the first time, to have access to that patient's full medical history, instead of having to rely on the patient's memory.

Currently there are many protocols that facilitate interoperability between systems at various levels of communication. The current set of interoperability protocols are created by different groups, usually provide a single main service and make use of different syntax and semantics. Many of these protocols are efficient and provide advanced features. However, this often comes at the cost of them being overly complex and leads to implementation difficulties for programmers who, in most cases, have to read long and complex documentation.

In addition to the complexity, there are also some interoperability protocols and standards that forgo advanced features for the sake of simplicity. For instance, the Really Simple Syndication (RSS) standard is extremely popular and its popularity seems to be linked to the simplicity of the protocol and the ease with which RSS feeds can be implemented [10]. It could be argued that, interoperability protocol adoption rates can be improved if the protocols are simplified and one could argue for the simplification of all high-level interoperability protocols. Furthermore, it could be useful to create a suite of protocols that are consistent and that allow for the implementation of one particular protocol in the suite to make it easier implement others with minimal incremental work. The possibility of improvement comes from the premise that if protocols had more in common it would be easier to provide solutions to interoperability problems.

Thus, this paper proposes the design of an experimental suite that combines simplicity and efficiency to improve interoperability by combining various high-level interoperability services into a single suite of protocols. The goal of this suite is to support only the minimal required functionality, while not compromising efficiency. While the proposed suite combines multiple high-level interoperability protocols, each of the protocol is independent of the others, i.e. the implementation of one protocol should facilitate the implementation of another protocol given the acquired common knowledge, but it is not required that

SAICSIT '12, October 01 - 03, Pretoria, South Africa

all protocols be implemented. It is not the purpose of this paper to dissuade people from using existing and established protocols/standards and adopt the proposed suite. Instead, the focus of this paper is to show that there is, in fact, lots of room for improvement in the design of the current set of protocols and to suggest a possible alternative direction for high-level interoperability research and design.

The rest of this paper is structured as follows. Section 2 discusses the background and current state of different interoperability protocols. Section 3 presents the results of a user survey and the design of the experimental protocol suite. The evaluation of the suite is presented in Section 4 and, lastly, the conclusion is presented in Section 5.

2. BACKGROUND

Interoperability is the capability of different systems to communicate and exchange data with each other, using a set of predefined formats, standards and protocols that allow the systems to successfully use each other's services. In this section, some existing interoperability protocols and standards are discussed and analysed as well a brief discussion about some interoperability research projects.

2.1 Interoperability Protocols and Standards

A protocol is a set of formal rules that determine the way in which two systems communicate [31]. A standard is a documented agreement containing precise technical specifications to be used as rules/guidelines to ensure that the products/services are fit for their purposes [31].Protocols can be either low-level protocols or high-level protocols, where the former defines the physical and electrical characteristics of the communication while the latter defines the data formats for message encoding and information control, message syntax, syntax for communication between devices, flow control and error handling [31]. The experimental protocol framework proposed in this paper is a high-level interoperability protocol. There are a number of notable interoperability protocols and standards, some of which are briefly discussed below.

2.1.1 Really Simple Syndication (RSS) and Atom

RSS is an XML-based Web content syndication standard that periodically checks Web sites in search of updated content that is then delivered to subscribers [9][35]. It provides subscribers with an organised list of notifications about new and updated Web content [33], which is known as a RSS feed. A RSS feed consists of a number of entries, which can be news headlines, full-text articles, article excerpts, weather reports, podcasts, etc, where each entry contains a set of metadata elements e.g. title, link, description [32]. RSS feeds can be accessed by anyone who is interested in the content it provides. Maintaining an RSS feed involves creating an RSS document (an XML file) that is Web accessible to RSS aggregators. The RSS documentation is short (less than 10 pages), simple and most programmers can have an RSS feed up and running in a few hours.

The simplicity of RSS is a controversial issue with some arguing that it is excessively simple and therefore lacks structure/semantics that are crucial for improved levels of functionality and security [23]. However, the vast majority see the simplicity of RSS as its main success factor [34], a view that is strongly supported by the large number of Web sites that have implemented RSS feeds to share content, including sites for major news broadcasters, such as CNN, BBC and the New York Times [33]. RSS is seen by some as being synonymous to syndication [34]. Based on the success of the RSS simplicity model, one can't help but ask "Can a similar simplicity model be applied to other standards and protocols"?

Atom is a syndication standard this is similar to RSS but aims to improve the way in which RSS provides syndication of Web resources (i.e. weblogs and news headlines) to Web sites and directly to users [30]. A feed in Atom is equivalent to an RSS feed with some minor differences, to support the shortcomings its creators identified in RSS.

2.1.2 Z39.50

The Z39.50 protocol is an application layer protocol that supports distributed search and retrieval between structured network services [6]. This protocol stipulates data structures and interchange rules that allow a client machine to search and retrieve records from databases on a server machine, across different platforms [29]. It is widely used by librarians and very often integrated into library systems and personal bibliographic reference programs (e.g. interlibrary catalogue search with Z39.50 queries).

Z39.50 supports lower level OSI services [28], but despite that many implementers chose to layer it on top of TCP/IP, as opposed to implementing it in an OSI environment to benefit from the full OSI services. At its peak time some of the reasons given for this were: **a.** The size and complexity of OSI implementation was daunting; **b.** There was no mature OSI software available for the full range of computing environments in use at the implementing institutions; **c.** The architectural structures within the OSI application layer were seen as unstable; **d.** The complexity of the OSI upper layers would outweigh whatever benefits an OSI implementation had to offer; **e.** The protocol required a particular type of software to be installed, configured and maintained in order to work, making it expensive; **f.** No bulk data transfer. In more recent years researchers hoped that the use of newer technologies (e.g.XML, RDF, SRU etc) [1] would help revive the now almost obsolete protocol.

2.1.3 OAI-PMH

The Open Archives Initiative - Protocol for Metadata Harvesting (OAI-PMH) [2] is a protocol that provides an application-independent framework for metadata transfer. It was designed to be easy to implement (based on widely accepted standards such as HTTP, XML and Dublin Core) and highly efficient. There are two actors in the OAI–PMH framework: a data provider and a service provider. A data provider uses OAI–PMH to expose metadata about repository content to service providers [25] and maintains one or more repositories. A service provider uses OAI–PMH to harvest metadata from data providers. In the context of OAI–PMH, the term harvesting refers to collecting metadata from different repositories and the possible storage of all metadata in a central database.

2.1.4 OAI-ORE

The Open Archives Initiative - Object Reuse and Exchange (OAI-ORE) is a protocol that facilitates the description and exchange of Aggregations of Web Resources [26] and is used in many applications [19]. For instance, JSTOR, an online academic journals archive, uses OAI-ORE in results visualization and topology navigation tools; and the DANS (Data Archiving and Networked Services) uses OAI-ORE to improve its "Durable Enhanced Publica-

tions" by providing durable access to value-added services and datasets [22].

2.1.5 Simple Web-service Offering Repository Deposit (SWORD)

SWORD is a lightweight protocol for depositing content from one location to another [4]. SWORD's main focus is on depositing content into repositories, but this can potentially be used to deposit content into any system that is willing to receive it [4]. SWORD can be used to facilitate e-Learning applications as demonstrated by examples, such as its use: in a drag-and-drop desktop tool; for bulk deposit for sharing metadata; to deposit from a content creation tool; or to drag-and-drop news feed resources into a repository [14].

2.1.6 Search/Retrieval via URL or WEB (SRU/W)

Search/Retrieval is a service for search and retrieval of Web resources across the Internet where a client makes a search request for the retrieval of matching records from the server [7]. This standard is based on the Z39.50 protocol. The protocol can be used in two different ways [5]: as parameters in a URL, called Search/Retrieval URL Services or SRU; or as SRU via HTTP SOAP, formerly know as Search/Retrieval Web Services or SRW [8].

2.2 Interoperability Research

In addition to the wide variety of interoperability protocols and standards, there are some notable interoperability research projects that have set out to provide solutions to a range of interoperability issues, as well as experiment with and extend existing protocols. Some of these are discussed below.

2.2.1 The Kahn-Wilensky Architecture

Kahn-Wilensky is an open architecture that supports a large and extensible class of distributed digital information services, such as digital library services [24]. This architecture provides a naming principle and a service that uses those names for the identification and location of digital objects, as well as for providing a protocol access to objects. Kahn-Wilensky's underlying architecture forms a base for extensions that can be customized for information of various formats [12]. Implementations based on the Kahn-Wilensky architecture include the Interoperable Secure Object Stores (ISOS) and the Dienst protocol [17].

2.2.2 The Dienst Protocol

Dienst is an architecture and protocol for digital libraries across multiple servers [16]. Initially called the Computer Science Technical Report Project, this ARPA funded project originated from the need to create a digital library of Computer Science technical reports [15]. Over one hundred institutions used Dienst to be part of the Networked Computer Science Technical Reference Library (NCSTRL) but, with the creation of OAI-PMH, many of them have transitioned to the latter. The design of OAI-PMH was largely based on improvements and in some cases simplifications made by considering the lessons learned from the design of Dienst; for example, Dienst supports over 30 verbs and OAI-PMH only supports 6 verbs [21]. Furthermore, a harvesting approach is preferred to a cross archive searching approach because it avoids the problem of not getting up to date results (possible in federated search if one or more repositories are down) by collecting and storing all data in a central location. In 2003, a Dienst OAI-PMH Gateway (DOG) was created to allow OAI-PMH harvesting from existing and at risk Dienst repositories [21].

2.2.3 Simple Digital Library Interoperability Protocol (SDLIP)

SDLIP is a protocol that defines simple interfaces for interoperability between data providers [20]. SDLIP's main goals are: the simplification of both client and server side implementations; server support for stateful and stateless operations; dynamic load balancing for servers; support for thin clients; and implementations via distributed object technology (CORBA and HTTP/CGI) [3].

This section has provided a brief introduction to interoperability by discussing and analysing some of the most popular and well-known protocols and standards as well presenting a discussion of various research interoperability projects. The next section describes a study aimed at gaining insight into what users wanted most from an interoperability protocol. The data collected from the survey was used to make the design decisions of the suit of protocols described in this paper.

3. USER STUDY AND PROTOCOL DESIGN

An online questionnaire was conducted in order to gain information from protocol implementers and users about the state of currently used protocols and how, if at all, they could be improved. Invitations to participate were sent to mailing lists from different communities[1] and participants were asked to answer 6 questions about various protocols/standards, namely RSS, Atom, APP, Z39.50, OAI-PMH, OAI-ORE, SRU/W and SWORD. [2]. The participants were asked to answer questions related to:

1. Their level of confidence ranking from expert implementer to having never heard of the protocol.
2. To list other protocols they were familiar with and their expertise.
3. Most useful feature(s) of each of the protocols.
4. Least useful feature(s) of each of the protocols.
5. To list possible improvements to existing protocols.
6. Any general comments they might have.

30 participants took part in the survey (7 null responses). The responses from the remaining 23 participants showed that OAI-PMH is the most popular, known by 61% of participants, followed by RSS (39%) and Atom (22%). APP was the least known, unknown by 74% of participants, followed by SRU/W and SWORD, which were both unknown by 44% of participants. The features that were most frequently mentioned as most useful were: simplicity, aggregation, platform independence, the use of popular standards and flexibility. The features most frequently mentioned as least useful were: lack of features, semantics issues, complexity and quality of documentation/specifications. Some of the improvements suggested were: better documentation, standardisation, simplicity and efficiency.

The results of this survey led to the conclusion that a good interoperability protocol should be: simple enough

[1] i.e. srw, eprints, dspace, atom, oai, rss and UCT grads

[2] These were chosen for appearing to be some of the more popular protocols/standards on the discussions of various interoperability related mailing lists

to allow programmers to implement, explore and experiment while requiring only operations that are crucial to the performance of the protocol, but also robust. These findings were important in drawing up a plan to achieve the simplification goal. Thus, in order to try to find potential solution(s), a set of experimental protocols were designed to be evaluated against the existing ones.

Having simplicity and efficiency as the key design goals, a "ground-up minimalistic design approach" was chosen to design the set of experimental protocols. In this approach the design starts with the basic and absolutely necessary support (i.e. modes for the transmission and encoding of data) and then only adds the features that are needed for an efficient protocol, as opposed to adding every possible combination of features.

3.1 High-level Interoperability Protocol - Common Framework (HIP-CF)

HIP-CF is a common framework that facilitates high-level interoperability between heterogeneous systems. It has different interoperability protocol services built as a layer on top of a common framework, thus creating a suite of protocols. The protocols use HTTP for data transfer and therefore follows the HTTP specifications as stated in RFC 2616 [18]. The protocols supported by the HIP-CF suite are: Xbrowse for browsing data; Xharvester for harvesting data; and Xsearch for identifying and retrieving specific data. In this section, HIP-CF is described. This description begins with a listing of the terminology used, followed by the principles and guidelines for implementing HIP-CF protocols. Thereafter, the HIP-CF general model is described, including the protocol model, the application layer protocol, the parameter model and error messages. Thereafter, an example of one protocol based on HIP-CF will be described.

3.1.1 Terminology

The terms presented in this section are used to define the roles and/or actions involved in the functionality of the protocol[3]. The terminology is presented in alphabetical order and not in order of occurrence or use.

Client - A computer connected to a network that makes a request for resources or services from a server located elsewhere on the network [31]. A client can also be a server and vice-versa.

Data Source/Repository - A system that exposes its contents and allows access to it by external sources.

Digital Library - A managed collection of digital information, with associated services, where the information is accessible over a network [11].

Metadata - Data that describes data, for example metadata about a journal article can include the author's name, the title and the publication date.

Query - A request made to a computer system (e.g. database, digital repository) to retrieve a particular set of data records [31]. A query can have one or more parameters.

Record - A data structure consisting of a collection of fields, each possibly containing a different data type [31]. A record here refers to a metadata record presented in XML.

Request - A message that requires some form of reply from its recipient [31].

Response - A document returned by the server as a result of a client request.

Server - A computer or computer program that is designed to provide shared services to other computer systems on a network.

URL - The Uniform Resource Locator is a string of characters used to identify a resource on the Internet (e.g. a Web page, a server or a a file) [11].

XML - eXtensible Markup Language (XML) is a set of rules for encoding structured documents in a machine-readable form[4].

3.1.2 HIP-CF Principles and Guidelines

Implementations of HIP-CF services MUST comply with the principles and guidelines below:

Simplicity.

HIP-CF implementations **SHALL** be as simple as possible. The protocol is designed to be efficient and reliable but, most importantly, to be simple, as opposed to being complex and computationally expensive. The motto is "do what needs to be done in the simplest possible way, and add complexity only when or where strictly necessary". A bottom-up approach is **RECOMMENDED** for the implementation of all protocol. In other words: start from nothing, and build up the implementation until the desired functionality and efficiency is achieved. Use the simplest available solutions and avoid any non-crucial elements. Simplicity is important because it is a key factor to reduce costs, save time and possibly increase compliance with the protocol specification.

Robustness.

HIP-CF implementations **MUST** be robust, i.e. have the ability to recover from invalid input data and other error conditions, as well as operate in adverse conditions.

Data Storage.

There are no requirements for where or how data should be stored. It is left to the organisation or individual implementing the protocol to choose a solution that best suits their needs. Data sources **MUST** ensure that the repository/system is configured to allow clients to find and access the resources available in the repository.

Reuse.

HIP-CF implementations **MUST** take advantage of existing technologies and standards such as HTTP, XML and REST.

3.1.3 HIP-CF General Model

The framework provides a consistent and extensible standardised structure (vocabulary, principles and guidelines) (see Figure 1) that is based on the Representational State Transfer (REST) architecture, and allows the implementation of different high-level interoperability protocols.

Protocol Model.

The protocol has two main components: a client and a server. The client and server communicate through request-response pairs. For simplicity, it is **RECOMMENDED** that, when possible, HIP-CF implementations use 1 request-response pair for record retrieval instead of sending a request to find out if there are matching

[3]Unless otherwise indicated by a reference, the terms used in this document are defined according to their specific use in or for HIP-CF.

[4]http://en.wikipedia.org/wiki/XML

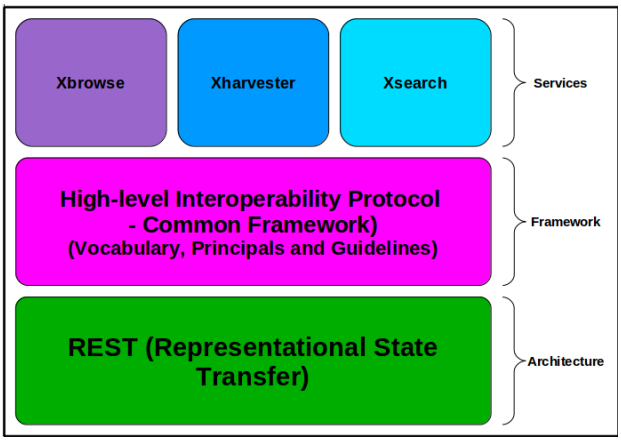

Figure 1: High-level Interoperability Protocol - Common Framework.

records and then sending a second request to retrieve the records.

The Application Layer Protocol.

The **RECOMMENDED** protocol to transfer the request and response pairs between the client and the server is the HyperText Transfer Protocol (HTTP) (see Figure 2). HTTP is an application layer protocol for distributed, collaborative and hypermedia information systems [18]. Requests can be sent using either one or both HTTP GET and HTTP POST. Special characters in the URL (e.g. space, $, <, :) should be encoded according to URL encoding standards[5].

Using HTTP GET the parameters are sent via the URL using the standard method of parameter passing where parameters are separated from the URL by a question mark and from each other by an ampersand, as is shown below:

```
http://www.serverexample.com?parameter1=one&
parameter2=2.
```

Using HTTP POST the arguments are carried in the message body, which has the advantage of not limiting the length of the arguments. For example:

```
Post  http://www.serverexample.com
Content-Length: 128
Content-Type: application/x-www-form-urlencoded
Request=parameter1=one&parameter2=two
```

It is **RECOMMENDED** that, whenever possible, customized messages and solutions for HTTP errors are used in order to help the client/user better understand errors and find solutions. HTTP error messages may also be used to control data traffic, for example error code 307 can be used to temporarily redirect requests to a different location.

Parameter Model.

Each protocol defines a set of parameters. There are the mandatory parameters (parameters that MUST be implemented) and optional parameters (parameters that improve the quality of the results but are only used by choice).

[5]http://www.w3schools.com/tags/ref_urlencode.asp

Error Messages.

When a request generates an error the server responds by sending the client an error message. While an error message **MAY** contain both machine and human readable formats, at a minimum it is **RECOMMENDED** that it contains an HTTP error code.

3.2 Xsearch

The previous section described the requirements and recommendations for a protocol based on the HIP-CF framework. In this section, Xsearch, which is an example of a protocol based on the HIP-CF framework will be described. Xsearch is only one of the three protocols based on HIP-CF; however, it demonstrates how simple a protocol based on HIP-CF can be and how easily it can be implemented. The Xsearch protocol is a high-level interoperability protocol that provides a simplistic service for clients to search for resources from digital libraries, databases and other sources of digital data. Search is the process by which a client can retrieve an available resource from a server. To search for a resource, a client sends a request or query, which is made up of a search term and zero or more optional parameters to the server. The server responds by sending back a list of resources that match the client's request or a message to inform the client that none of the resources available on the server match the client's request.

3.2.1 Parameter Model

Xsearch has four defined parameters (see Table 1). A request has to contain the mandatory parameter *queryword*, and any combination of the optional parameters.

The *queryword* is the mandatory request parameter and it is made up of one or more words sent by the client in a request and is used by the server to retrieve resources whose content matches the parameter. The server has control over which parts of the resources/records are searched to find a match. The *rpp* parameter allows the client to decide on the number of records the server should display in a single response page. This is only applicable when there is more than one record that matches the query. On the response page, the server tells the client how many records in total match the query. If the number of records that match the query is higher than the number of records displayed on the results page, then the client can use the *rpp* parameter and *start* parameter, which is the starting point in the result set from which the user wants to retrieve results, to retrieve the remaining matching records. When the *rpp* parameter is not used, the server returns the default number of matching records per page starting from the first matching record. The default number is determined by the server settings.

There are a number of metadata formats that can be used to present a query's response, for example Dublin Core, MARC 21 and MODS. The choice of metadata format is related to the needs of the community implementing the protocol and the data source. As **RECOMMENDED** by the W3C[6], if necessary, an XML namespace[7] can be created to clear any ambiguities that may exist for elements that happen to have the same name and to group common elements together.

3.2.2 Processing Model

Processing occurs on the server side, where the server:

[6]World Wide Web Consortium. http://www.w3.org/
[7]http://www.w3.org/TR/xml-names/

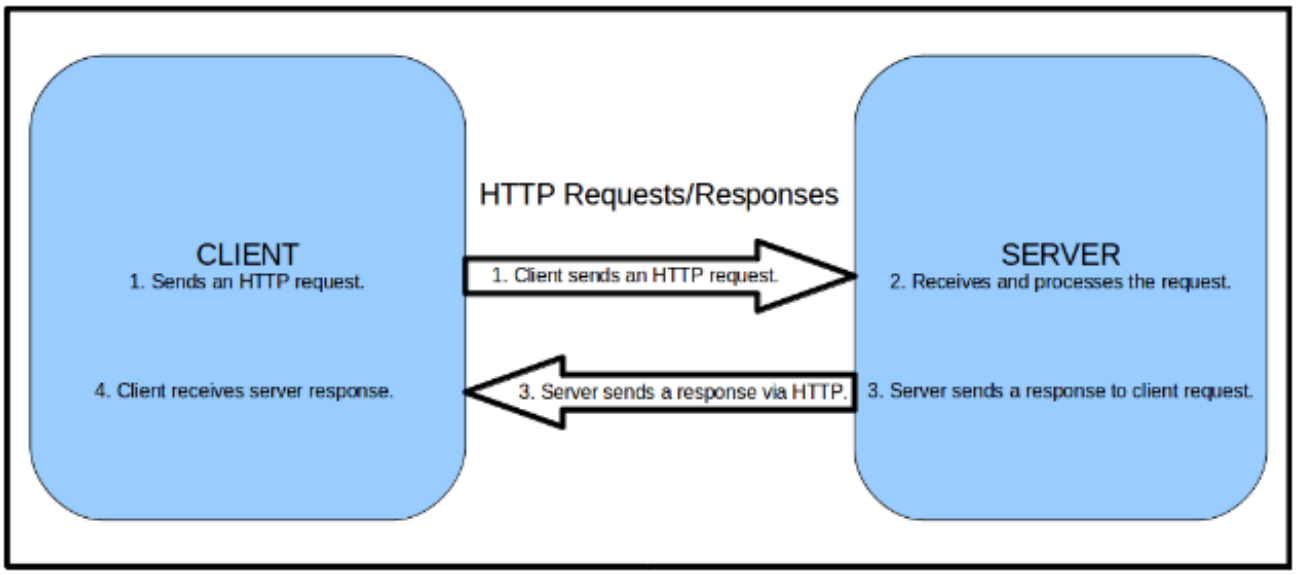

Figure 2: HTTP Request and Response Model.

Table 1: Xsearch parameters and their occurrence and descriptions

Parameters	Occurrence	Description
queryword	mandatory	The word/phrase submitted by the client which the server uses to check its resources in order to find matching records.
rpp	optional	Records Per Page (rpp) indicates the number of records to display per results page.
start	optional	The number of the first record on the results page. For example if the records counter is initialized at 0, start=1 will return a page of results starting from the second matching record.
metadataFormat	optional	Tells the server to only return results that are in the metadata format specified by the client.

- Receives a request (in a language/format that it understands).
- Decodes the request to get the parameter values.
- Uses the parameter values to perform a search query on its own data source.
- Gets a response from the data source and encodes it in a format that the client understands.
- Sends back the response.

The client also processes the results page(s) returned by the server and uses the data/matching record(s) as it sees fit. However, the use of the retrieved data by the client is outside the scope of Xsearch.

3.2.3 *Query Model*

Any query language may be used in the implementation of Xsearch. The choice of query language will be influenced by, amongst other things, the data source.

3.2.4 *Result Set Model*

The result set is a list of records or an error response returned by the server as a response to a request. The result set is usually an ordered list whose format and contents are defined according to request parameter values or server settings. Result sets usually only contain metadata.

3.2.5 *Protocol Example*

An example of the use of the Xsearch protocol is given below, where parameters help fine-tune the result set. In the example below, the record returned will be the first record that matches the query, which searches for the *queryword* "John."

```
<results>
<total_matches>3</total_matches>
<record>
<title>
Survey of the Effectiveness of Current
Interoperability Protocols
</title>
<creator>John Doe</creator>
<description>
Interoperability is the capability of different
systems to communicate and exchange data with
one another...
</description>
<keyword>
interoperability, interoperability survey,
interoperability protocols
</keyword>
</record>
</results>
```

Figure 3: An example of the data returned by an Xsearch query.

```
http://127.0.0.1/cgi-bin/Search.pl?queryword=john
&start=0&rpp=1
```

An example of the result to the above query is given in Figure 3

This section has described HIP-CF and provided an example of a protocol based on HIP-CF. In the next section, protocols based on HIP-CF are evaluated in terms of their perceived complexity and descriptive capabilities.

4. EVALUATION

Three methods of evaluation were used: a case study in which the three HIP-CF protocols were implemented; a user study investigating the perceived complexity of HIP-CF compared to existing protocols; and an evaluation of the entropy capabilities of HIP-CF.

4.1 Eprints Case Study

The first step of the evaluation was a case study of the practicality of implementing the suggested experimental protocol. As a proof of concept, the three protocols (Xbrowse, Xsearch and Xharvester) were implemented using the EPrints Digital Library platform as the storage unit. The client applications were all developed in Perl and the query language used was MySQL version 5.1.54. Although an online interface was available, the client application sent HTTP requests via the URL and not by using the HTML forms available on the user interface. All response files were XML files. It was found that all services could be successfully implemented and each protocol was able to provide the services as expected.

4.2 User Understandability Evaluation

The understandability/complexity experiment is a measure of how complex or simple it is to understand a protocol. Users were asked to read the documentation of a HIP-CF protocol and an existing protocol that provides equivalent functionality. Based on their understanding of the documentation, they completed a questionnaire to indicate which they thought was the simplest protocol. It should be noted that this was not an evaluation of the individual features supported by the protocols but rather an evaluation of how easily the main function of a protocol could be implemented/achieved. However in future, for a deeper analyses, it would be useful to evaluate programmers actual implementations and use of the different protocols.

The evaluation involved a total of 27 people all of whom were computer science students chosen evenly across the different academic levels.

4.2.1 Individual Protocol Service Evaluation

The first part of the evaluation involved 23 participants, all computer science students at undergraduate or honours level. The students were asked to read the documentations of a HIP-CF protocol and an equivalent protocol and then complete a questionnaire. The participants were each given a copy of official documentation for existing protocols and HIP-CF documentation. It was asked that they read the documentations, see how well their understood them, evaluate how difficult they expect an implementation to be and based on that do a comparison and complete the questionnaires. The protocols that were compared were: OAI-PMH vs. Xharvest and SRU vs. Xsearch.

The questionnaire required the participants to:

- Name the two protocols they read.
- Rate their own level of programming skills.
- State if they were familiar with high-level interoperability protocols before the evaluation session and, if they were, to name the protocols and their level of expertise.
- State the type of service each of the protocols provided.
- Rate their level of understanding for each of the two protocols on a scale of 1-10.
- Rate the writing and presentation style of the protocols on a scale of 1-10.
- Rate the perceived degree of implementation difficulty on a scale of 1-10.
- Provide general comments about the protocols.

Of the 23 participants, 10 provided answers for the SRU vs. Xsearch comparison, 7 provided answers for the OAI-PMH vs. Xharvest comparison and 6 were null answers.

Xsearch vs. SRU.

In the search comparison, 4 of the 10 participants rated themselves as being average programmers and the other 6 rated themselves as being above average. 7 participants chose "no" to having previous knowledge of high-level interoperability protocols, 2 participants chose "yes" and 1 participant had a null answer.

After reading the documentation, the arithmetic mean level of understanding for Xsearch was 7.7 (closer to 'I now have a good understanding of the protocol') and 3.6 for SRU (closer to 'I still don't understand the protocol'). Comparing Xsearch to SRU in terms of levels of understanding, 80% of users said their understanding of Xsearch was above average as opposed to 70% of SRU users saying their understanding was *below* average. Based on this result, Xsearch is potentially simpler to understand than SRU.

The arithmetic mean for the writing and presentation style of Xsearch was 7.9 (closer to very good) and 3.7 for SRU (closer to poor). Comparing Xsearch to SRU in terms of writing and presentation style, the results showed that the majority of users (90%) were above average, which indicates that they are satisfied with Xsearch's writing and presentation style and also the majority of users (60%) were below the average for SRU, which indicates that they were not satisfied with SRU's writing and presentation style. Based on this result, Xsearch is potentially simpler than SRU.

The arithmetic mean for the perceived degree of difficulty associated with Xsearch's implementation was 5 (average) and 7.7 for SRU. Comparing Xsearch to SRU in terms of perceived degree of difficulty associated with the implementations, the results indicate that, although the number is higher for SRU, for both protocols more than 50% of participants did not perceive implementation to be a trivial task. However, based on the mean values, Xsearch's implementation is expected to be easier than that of SRU.

The overall conclusion from the comparison of Xsearch and SRU is that Xsearch is not only simpler to understand but also has a more usable writing and presentation style than SRU. Although the ratings of the perceived degree of implementation difficulty between the two protocols are very close to each other, from this result it is possible to assume that, because of the advantage in the other categories, Xsearch is better overall, which supports the claims made by this research and implies that the SRU protocol could benefit from greater simplicity.

Furthermore, it was found that previous knowledge of protocols had a much stronger impact on participants' responses than programming skills did. In a lot of cases, self-rated expert programmers rated the levels of understanding lower and the degree of perceived implementa-

tion difficulty higher than average programmers with previous knowledge of interoperability protocols. This can be an indication that knowledge acquired from implementing one protocol from a suite of protocols may be more advantageous in implementing another protocol in the same suite than a higher level of programming expertise could be.

Xharvester vs. OAI-PMH.

In the harvesting comparison, 3 of the 7 participants rated themselves as being average programmers and the other 3 rated themselves as being good programmers. All 7 participants chose "no" to having previous knowledge of high-level interoperability protocols.

The arithmetic mean level of understanding for Xharvester was 7.9 and 4.6 for OAI-PMH. Comparing Xharvester to OAI-PMH in terms of levels of understanding, the numbers clearly show that Xharvester, with 90% above average, can be simpler to understand than OAI-PMH with 60% below average. Based on the arithmetic mean values, Xharvester is potentially simpler to understand than OAI-PMH.

The arithmetic mean for writing and presentation style for Xharvester was 8.3 (closer to very good) and 4.4 for OAI-PMH (closer to poor). Comparing Xharvester to OAI-PMH in terms of writing and and presentation style, the results show that Xharvester is potentially simpler.

The arithmetic mean for the perceived degree of difficulty associated with Xharvester's implementation was 5.7 (half way between easy and difficult) and 7.9 for OAI-PMH (closer to difficult). Comparing Xharvester to OAI-PMH in terms of perceived degree of difficulty associated with the implementations, the results show that, while the numbers differ for each protocol, the majority of participants see the implementation of either of the protocols to be a relatively difficult task.

The overall conclusion from the comparison of Xharvester vs. OAI-PMH is similar to that of Xsearch vs. SRU in the fact that the HIP-CF protocols are deemed simpler. Xharvester also proves to be simpler to understand as well as more usable in terms of writing and presentation style than OAI-PMH. The ratings of the perceived degree of implementation difficulty between the two protocols indicate that, according to the participants, neither one of the implementations would be an easy task. As in the case of SRU, OAI-PMH can also benefit from some simplification on how the data/information is presented to possible users.

4.2.2 Protocols Suite vs. Individual Protocol Service Evaluation

The second part of the user evaluation involved 4 expert masters students. It was carried out in a manner similar to the individual protocol evaluation but, in this case, each participant had to read the complete HIP-CF documentation as well as both OAI-PMH and SRU documentation. This part of the evaluation had a considerably smaller number of participants when compared to the individual protocols evaluation, that is also reflective of the number of Masters students versus the number of undergraduate and honours students in tertiary institutions. The participants were given the same set of questions as before; however, in this case they had to evaluate HIP-CF, OAI-PMH and SRU. In addition to this, they were also asked to answer the following question according to the scale below:

Having a protocol that supports multiple high-level interoperability services is? Rate according to the scale below.

- A great idea, since developers who implement one of the services (e.g. search) are likely to also implement the other services (e.g. browsing and/or harvesting) and knowledge of the common framework may facilitate the overall process.
- A good idea, one option for multiple requirements.
- Unnecessary. The current situation works just fine.
- Bad idea, why mix the different services. It is simpler if each service is covered by an individual protocol.
- Will not work, in trying to cover too many areas the suite would end up not covering any of them properly.
- None of the above. (Please elaborate)

The results are summarised in Table 2 and they suggest that OAI-PMH is the simplest of the three protocols, followed by HIP-CF and then SRU. However, all protocols are perceived as being difficult to implement.

It is also noted that previous high-level interoperability protocol knowledge is an important factor for participants/possible users to understand any protocol. This is evident from the two participants who were familiar with OAI-PMH, which helped them in understanding the other protocols as was noticeable by their scores for all protocols which were, in most cases, higher/better than those of the other two participants.

It's significant that all participants agreed that a protocol suite approach is good. So it can be argued that, ideally, high-level interoperability protocol should be part of a suite of protocols and at least as simple as OAI-PMH.

4.3 Entropy

The last way in which HIP-CF was evaluated was in terms of its entropy, which is a measure of the amount of order or predictability in a message [13] and of its information content and quantity. It is directly related to data compression, in the sense that, ideally, the length of a message after it is encoded should be equal to its entropy. The value of entropy is equal to the minimum number of bits necessary to encode a message without losing any valuable information. This number helps eliminate non-crucial information from the message, such as pieces of information that have a probability of 1 because they do not change and are always present, such as the `<html>` and `<body>` tags in an .html file. Elements with a probability of 1 have an entropy of 0. With the entropy value, a message can be compressed to obtain a representation of the data file that occupies less space but preserves all of the information [13] and the overall entropy is the average of the entropy of the individual probabilities occurring. It is calculated by the following formula [13]:

$$E = -\sum_{i}^{n} P_i \ \ logP_i \ \ bits \tag{1}$$

Using Equation 1, the entropy was calculated for a single file using HIP-CF protocols and their equivalents. The entropy values are shown below:

Xsearch = 1324 bits

Table 2: Protocols Suite vs. Individual Protocol Service Evaluation Results

Evaluation Factor	Results
Understandability Results: HIP-CF	All participants rated their levels of understanding of HIP-CF above the average level. With 100% above average, the results indicate that the users' understanding HIP-CF ranges from relatively good to really good.
Understandability Results: OAI-PMH	OAI-PMH was rated at average or above average by all participants. There were two participants who rated OAI-PMH at level 10. These two participants had previous knowledge of interoperability protocols and specifically with OAI-PMH. The two participants who rated their understanding at average and just above average were not familiar with OAI-PMH prior to the evaluation. With 75% above average, OAI-PMH's understanding also ranges from relatively easy to really easy.
Understandability Results: SRU	SRU was the only protocol to get a below average rating. The ratings for this protocol were divided by about 50% between really easy to understand and average or below, suggesting that it is easier for people with knowledge of high-level interoperability protocols to understand it, than it is for those with no prior knowledge.
Writing and Presentation Style Results: HIP-CF	1 participant rated this category below average and the other 3 rated it above average. With 75% above average the results indicate that HIP-CF's writing and presentation style is well accepted by the participants, suggesting that it is simple and easy to follow and comprehend.
Writing and Presentation Style Results: OAI-PMH	1 participant rated this category average and the other 3 all rated it above average.None of the participants rated this category below average. This result indicates that OAI-PMH's writing and presentation style is also well-accepted, making it simple and easy to follow and comprehend.
Writing and Presentation Style Results: SRU	1 participant rated this category below average, 1 participant rated it average and the other 2 participants rated it above average. SRU ratings for writing and presentation style were balanced between average and above average, with 50% of participants rating it as average and below and the other 50% rating it above average. While its lowest rating is the same as the HIP-CF lowest rating, its highest rating is not as high as the other protocols highest ratings.
Perceived Implementation Difficulty: HIP-CF	1 participant rated the perceived degree of difficulty associated with implementing HIP-CF as an average task and the other 3 participants rated it above average. With 75% of ratings above average, 2/3 of it at level 9, HIP-CF's implementation is perceived as difficult.
Perceived Implementation Difficulty: OAI-PMH	1 participant rated the perceived degree of difficulty associated with implementing OAI-PMH below average and the other 3 rated it above average. The results indicate that as with HIP-CF, OAI-PMH implementation was considered above average difficult by most participants, in this case also by 75% of participants, although at different levels.
Perceived Implementation Difficulty: SRU	1 participant rated the perceived degree of difficulty associated with implementing SRU below average and the other 3 participants rated it above average difficult. Similar to the other two protocols, SRU results also indicate that 75% of participants perceive implementation to be an above-average difficult task.
A suite of protocols vs. individual protocols	The participants were asked to give their opinion on whether it is better to have individual protocols (which is the current situation) or have a suite of protocols, i.e. one suite that supports multiple high-level interoperability protocols (proposed in this research). Three participants chose option A and 1 participant chose option B. The results indicate that all participants agree that a suite of protocols is better than having individual protocols.

SRU = 3737 bits

Xharvester = 9625 bits

OAI-PMH = 14407 bits

For both search and harvesting, entropy calculations show that the HIP-CF protocols require considerably fewer bits in order to encode the data messages compared to both SRU and OAI-PMH, thereby indicating that HIP-CF can provide a better data compression ratio.

5. CONCLUSION

In this paper, it was hypothesised that simpler interoperability protocols and standards will lead to an increase in adoption levels, thereby making it is easier for programmers to understand and implement them and therefore leading to more interoperable systems. To that end, an experimental suite of protocols was designed, implemented and evaluated. The usability evaluation tested to see how programmers would react to an alternative suite of protocols. The evidence showed that the programmers would rather implement the suite of experimental protocols than the existing set because the former appeared to be simpler and easier to implement, while still providing equivalent functionality. Furthermore, entropy calculations showed how the simpler protocols were more efficient at encoding messages than their existing equivalents. While this paper does not suggest that anybody should ever implement the HIP-CF protocols, it has shown that there is enough evidence from this experimental study to suggest that it is possible to do better than we are currently doing. It also calls attention to the possibility of a new route for the design of high-level interoperability protocols in order to make them easier to understand and implement and, in doing so enhance interoperability and the ways in which heterogeneous systems share information.

6. REFERENCES

[1] LC Z39.50/SRW/SRU Server Configuration Guidelines. Library of Congress. http://www.loc.gov/z3950/lcserver.html.

[2] *The Open Archives Forum Online Tutorial*. http://www.oaforum.org/tutorial/.

[3] *The Simple Digital Library Interoperability Protocol (SDLIP-Core). Stanford Digital Libraries Technologies*. http://diglib.stanford.edu:8091/ testbed/doc2/ SDLIP/.

[4] Simple Web-service Offering Repository Deposit (SWORD). http://swordapp.org/.

[5] SRW: Search/Retrieve Webservice. SRW-EditorialBoard, May 2004. http://srw.cheshire3.org/SRW-1.1.pdf.

[6] JISC Information Environment Architecture Glossary, 2005.

[7] CQL: Contextual Query Language (SRU version 1.2 specifications). Library of Congress, 2008. http://www.loc.gov/standards/sru/specs/cql.html.

[8] *SRU Protocol Transport (SRU Version 1.2 Specifications). Library of Congress*, February 2008. http://www.loc.gov/standards/sru/specs/transport. html.

[9] Really Simple Syndication Specifications, Tutorials and Discussion. RSS 2.0 Specifications, Version 2.0.11. RSS-Advisory-Board, March 2009. http://www.rssboard.org/rss-specification.

[10] Atom Publishing Protocol - Popularity, 2010. http://atompub.org/popularity.html.

[11] W. Arms. Manuscript of Digital Libraries. M.I.T. Press, 2000. http://www.cs.cornell.edu/wya/DigLib/MS1999/ index.html.

[12] W. Y. Arms. Key Concepts in the Architecture of the Digital Library. In *D-Lib Magazine*. July 1995. http://www.dlib.org/dlib/July95/07arms.html.

[13] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall Inc, 1990.

[14] S. Currier. SWORD: Cutting Through the Red Tape to Populate Learning Materials Repositories, February 2009. http://www.sarahcurrier.com/publications.html.

[15] J. R. Davis and C. Lagoze. Dienst, A Protocol for a Distributed Digital Document Library. *Communications of the ACM*, 38(4), April 1995. DOI = http://doi.acm.org/10.1145/205323.205331.

[16] J. R. Davis and C. Lagoze. NCSTRL: Design and Deployment of a Globally Distributed Digital Library. *JASIS*, 51(3):273–280, February 2000.

[17] D. Feng, W. C. Siu, and H. J. Zhang. *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications*. Signals and Communication Technology. Springer, 2003.

[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *RFC2616 - HypertextTransfer Protocol: HTTP/1.1*, June 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[19] E. Giglia. Report on OAI 6: CERN Workshop on Innovations in Scholarly Communication. In *D-Lib Magazine*, volume 15, (9/10). September/October 2009.

[20] N. Green, P. G. Ipeirotis, and L. Gravano. SDLIP + STARTS = SDARTS - A Protocol and Toolkit for Metasearching. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (New York, NY, USA)*, pages 207 – 214. ACM Press, June 2001. DOI = http://doi.acm.org/10.1145/379437.379496.

[21] T. L. Harrison, M. L. Nelson, and M. Zubair. The Dienst-OAI Gateway. In *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries (Washington, DC, USA)*, pages 309–311. ACM Press, June 2003.

[22] M. Hoogerwerf. Durable Enhanced Publications. In *Proceedings of the African Digital Scholarship Curation Conference (Pretoria, South Africa)*, May 2009. http://www.ais.up.ac.za/digi/docs/hoogerwerf _paper.pdf.

[23] S. Housley. RSS Security. http://www.feedforall.com/rss-security.htm.

[24] R. Kahn and R. Wilensky. *A Framework for Distributed Digital Object Services*, May 1995. http://www.cnri.reston.va.us/k-w.html.

[25] C. Lagoze and H. Van de Sompel. The Open Archives Initiative: Building a Low-barrier Interoperability Framework. In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (New York, NY, USA)*, pages 54–62. ACM Press, June 2001. DOI = http://doi.acm.org/10.1145/379437.379449.

[26] C. Lagoze, H. Van de Sompel, P. Johnston, M. Nelson, R. Sanderson, and S. Warner. *Open Archives Initiative Object Reuse and Exchange. ORE Specification - Abstract Data Model*. Open Archives Initiative, October 2008. http://www.openarchives.org/ore/1.0/datamodel.

[27] C. Lagoze, H. Van de Sompel, M. Nelson, and S. Warner. *Open Archives Initiative Frequently Asked Questions (FAQ)*. http://www.openarchives.org/documents/FAQ.html.

[28] C. Lynch. *RFC1729 - Using the Z39.50 Information Retrieval Protocol*. Network Working Group, December 1994.

[29] C. Lynch. The Z39.50 Information Retrieval Standard. Part 1: A Strategic View of its Past, Present and Future. *D-Lib Magazine*, April 1997.

[30] M. Nottingham and P. Sayre. *RFC4287 - The Atom Syndication Format*, December 2005.

[31] D. Pountain. *Concise Dictionary of Computing*. Penguin Group, 2003.

[32] RSS Advisory Board. RSS Specification, March 2009. http://www.rssboard.org/rss-specification.

[33] What is RSS? Software Garden, July 2004. http://rss.softwaregarden.com/aboutrss.html.

[34] H. Wittenbrink. *RSS And Atom: Understanding And Implementing Content Feeds And Syndication*. Packt Pulishing, 2005.

[35] W. Wu and J. Li. Rss Made Easy: A Basic Guide for Librarians. *Medical Reference Quarterly*, 26(1), 2007.