Openly Accessing Linkbases

Erik Wilde and Christian Stillhard
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology, Zürich

TIK Report 134
January 2002

Abstract

In this paper, we investigate the requirements for linkbase access on the Web. The recent advancements of Web technologies (XML, XLink, and XPointer) have brought us one step closer to the vision of using the Web as an *Open Hypermedia System (OHS)*. However, some of the pieces to make this work are still missing, and this paper discusses which they are and the status of the current work in these areas. Concentrating on one of these pieces, the access to linkbases, the paper than continues by describing the prerequisites and requirements of such an access mechanism, and closes with a list of requirements and design principles that we will use in a next step to specify and implement a linkbase access protocol.

Keywords: XLink, Linkbase

1 Introduction

Open Hypermedia Systems (OHS) [25, 31] have been the focus of many interesting developments, one of them being the ability to separate links from content, creating *third-party links* (other terms such as *out-of-line links* and *external links* have also been used in various publications). The most important feature of third-party links is the ability to create links associating any content, in particular content to which a link's creator does not have write access. One of the prerequisites of third-party links is the existence of a common content format and the existence of a common link format, which make it possible to separate the process of content and link creation.

From a computer communications view, however, this is not enough, because for a truly *open* system communication protocols must exist which make it possible for clients to access content and links on any server. Seen from this perspective, we also need protocols for communications in order to make an OHS really work. In this paper, we investigate how much of this vision of OHS is already available with today's Web technologies, and how much still needs to be done in order to make the Web work as an OHS. Following this investigation, we concentrate on one of these issues and describe possible ways to solve the problem, as well as consequences for the overall vision of the Web as an OHS.

1.1 Current Technology

As stated above, the Web still has some shortcomings when looking at it from an OHS point of view, and the current list of key requirements for an OHS system shows what is there already and where missing pieces need to be filled in.

- *Common content format*

  The success of the *Extensible Markup Language (XML)* [8] as the new standard for content on the Web is so convincing that all major browsers (Microsoft Internet Explorer, Netscape Navigator, Mozilla, Opera, Konqueror) support it. Even though XML currently is mostly used for generating HTML for delivery to browsers, XML will increasingly be delivered to the client, either as XHTML [32], or for new content formats such as *Scalable Vector Graphics (SVG)* [19]. In addition, styling (ie, presentation) of XML content is supported by style sheet languages such as *Cascading Style Sheets (CSS)* [6] and the *Extensible Style Language (XSL)* [2].

- *Common access protocol*

  The *Hypertext Transfer Protocol (HTTP)* [20] is the protocol which is used not only for communications between clients and servers, but also increasingly for Web services in fully automated scenarios. In addition to HTTP, established mechanisms can be used for ensuring privacy (HTTP-S [35]) and for authentication (HTTP authentication [21]).

- *Common link format*

  Embedding links in XML documents so far is still in its infancy, even though the *Extensible Linking Language (XLink)* [17] has been finalized in June 2001, and has been the subject of standardization for more than two years. Client support for XLink still is marginal, but the multitude of possible applications and the opportunity to create much richer hypermedia documents than with HTML's restricted linking model [42] will hopefully help XLink to reach critical mass within the next year.

- *Common link presentation model*

  Even though the identification of links in content is possible using XLink, the actual presentation of links, and the behavior of a client traversing links, are not yet determined. First steps towards a model of how to present XLink have been taken [39], but it will take some time until a stable standard described how clients should handle XLinks.

- *Common fragment identification method*

  The embedding of links into XML documents using XLink is only one side of the coin. The other side is the issue of what these links can link to. This is the question of fragment identification. While resources in the Web are always identified by *Universal Resource Identifiers (URIs)* [4], the format of fragment identifiers is another issue. The *XML Pointer Language (XPointer)* [16] is currently being standardized, but it is not clear whether it will be universally accepted and implemented.

- *Common access to linkbases*

  While XLink includes a mechanism for pointing to linkbases (described in detail in Section 2), many of the questions arising when thinking of XLink being used as the

format for OHS information on the Web are still unsolved. Tempting perspectives have been presented by Weinreich et al. [40], but some of the underpinnings still need to be created in order to make the vision become reality.

It can be seen that even though the recent developments in Web technology have made available some of the required components for making the OHS vision come true, there still are some pieces missing. And without the link presentation, the fragment identification, and the linkbase access question being solved, it is simply impossible to implement interoperable clients and servers for an OHS-enabled Web. While link presentation and fragment identification are currently worked on, the question of a common method for accessing linkbases still needs to be tackled, and this paper is a contribution to this process.

1.2 Where we're heading

More generally however, this paper aims at paving the path for the adoption of XLink and its full support by browsers. We believe that in order to be successful for the Web, open hypermedia must be really open. Existing systems supporting third-party links such as the *Distributed Link Service (DLS)* [9], or *Webvise* [23,24], implement OHS, but make it hard for independent software developers to create components which could be used in conjunction with the prototypes, because they are based on their own proprietary data and link models. While the applications scenarios of these systems were very inspiring, they also to differing degrees rely on a closed-world assumption. Based on the recent developments in Web linking technologies, Weinreich et al. [40], Wilde and Lowe [41], and El-Beltagy et al. [18] discuss interesting applications of third-party links, but even though XLink is stable, there still is no access protocol for linkbases.

In the context of the *Open Hypertext Protocol (OHP)* [14] activities, it was planned to specify a protocol for an OHS which also would include linkbase access as one of its tasks. However, OHP is still under development, and major shifts in the orientation of OHP have been proposed [28]. Furthermore, we strongly believe that OHP mainly (and very well) serves research purposes, while our goal is to work towards a linkbase access protocol that seamlessly fits into the existing and evolving components of the Web. Recently, De Roure et al. [15] have published a survey of possible link service infrastructures (as part of the ongoing DLS development efforts), and have come to the conclusion that the *Lightweight Directory Access Protocol (LDAP)* is well-suited to match the requirements of access to large, distributed linkbases (other protocols considered were HTTP and Whois++).

In Section 3 we present a linkbase access model that is targeted for the Web, and therefore based on the XLink linking model. One of the key requirements is that the access protocol should be as ubiquitous as possible, for example to aid linkbase access through firewalls. We are trying to walk the path of many Internet standards which tend to be on the simple side in favor of being easy to understand, use, and implement. Of course, the danger here is to make it too simple to be useful, but the success of the Web is a great example of the fact that this threshold apparently seems to be very low, as long as the possible applications provide clear benefits to users. We do not attempt to provide a protocol for all possible linkbase access scenarios, our intention is to create a linkbase access method which fits into the frame given by existing or emerging Web technologies.

2 XLink

The *Extensible Linking Language (XLink)* [17] defines how links can be embedded in XML documents. Basically, XLink defines a mechanism for assigning link semantics to XML elements. The linking model of XLink has been the topic of a long discussion, and finally W3C settled for a compromise between HTML's extreme simplicity and very complex linking models (for a detailed analysis of the expressiveness of links refer to Moreau and Hall [29]). One of XLink's most important steps beyond HTML's linking model is its support of third-party links. The advantages of external links for a multitude of hypermedia applications have been described by many authors, for example by Vitali [38] and Verbyla [37].

While XLink's linking model is rather simple and by some people may be considered too simple, it should always be kept in mind that XLink should be regarded as the Web's link *interchange* format[1], this perspective of looking at XLink for example has been explored by Halsey and Anderson [26]. It is perfectly possible for information providers to use a much richer linking model internally, as long as they provide a mechanism to map this internal linking model to XLink as soon as linking information has to be delivered over the Web.

2.1 XLink Linkbases

XLink explicitly mentions linkbases and reserves a special role for links to point to linkbases. In XLink, a linkbase simply is an XML document containing XLinks. XLink remains silent how this could be effectively implemented when linkbases become very large, but this is acceptable because XLink is merely dealing with the exchange of links. However, in order to effectively implement large linkbases, additional mechanisms are required for querying a linkbase and retrieve only a subset of the links contained in this linkbase. The challenges and problems of dealing with large linkbases have been described by Thistlewaite [36], Ashman et al. [3] and Davis [12,13], and for this paper we consider the question of how to create and maintain such a linkbase out of scope.

However, the question remains how clients can access and query linkbases. As a first approach, the *XML Query Language (XQuery)* [5] or a subset thereof can be considered, but a closer view reveals some problems: While XLink defines its own linking data model, XQuery only operates on the XML structure, not taking into account the semantics of XLink constructs (such as default arcs for links in case of no arcs being explicitly specified). Furthermore, XQuery (still under development) is going to be rather complex, and it is questionable whether it would be a good idea to require all clients and servers which want to support linkbase access have to support XQuery.

Our approach therefore is to first define an abstract data model of XLink information, and then to define the linkbase access on top of this model. Even though this approach introduces another abstraction layer, it avoids all problems that come from a query method operating directly on a syntax, rather than its underlying semantics.

2.2 XLink Information Set

The first thing of XML to be standardized was the syntax. Later, when it became apparent that several other standards should be based on XML's data model rather than its syntax,

[1] In the same way as HTML currently is the content interchange format, which very often is generated from more complex data formats such as databases or content management systems.

the *XML Information Set (XML Infoset)* [11] was standardized, describing the abstract data model of XML documents. XLink's development is pretty similar. So far, only the XLink syntax has been formally standardized (the underlying data model is described by the specification's prose, but there is no formal model of it). However, it has already become apparent that a formal XLink Infoset, for example in the form of XLink Infoset contributions, would be useful, and Walsh [39] has published a first approach towards an XLink Information Set.

At the time of writing, it is not clear if and when W3C will work on the XLink Infoset, but we believe that in the same way as the XML Infoset has become essential for a number of XML-related standards (eg, XQuery), the XLink Infoset will be very useful for different link-related areas, such as the presentation of links [39], or the linkbase access discussed in this paper. We therefore assume that an XLink Information Set will become available in the near future, and as a first approach we base our considerations on the model presented by Walsh [39].

3 Linkbase Access

Based on an abstract model of XLinks linking model, linkbase access can be regarded as a way for clients to request links from a server providing this service. This approach to linkbase access is a bit different from the query-based approach and deserves some closer inspection.

3.1 Queries vs. Services

Seeing that XLink views linkbases simply as documents containing XLinks, it is tempting to see the task of extracting links from a linkbase simply as a query into this XML document. W3C currently is developing the *XML Query Language (XQuery)* [5] for making queries into XML documents. However, the query view of linkbase access has some serious disadvantages:

- *XQuery operates on the XML Information Set*

  While access to a linkbase should operate on XLink's data model (the XLink Infoset we introduced in Section 2.2), XQuery operates on the XML Infoset. Consequently, certain characteristics of XLink (such as default arcs) are not reflected in the data model XQuery operates on.

- *XQuery is a complex standard*

  XQuery is still under development, but the currently available draft documents clearly show that it will be a complex standard. XQuery is targeted at a much more complex goal than the requirements we have for linkbase access. An alternative would be to define a profile of XQuery (once it is stable), so that linkbase access would not require full XQuery.

- *Exposing the linkbase to the outside*

  Querying the linkbase with XQuery would result in exposing the linkbase to the client, who could download the complete linkbase systematically by using appropriate XQueries. This problem again could be addressed by using XQuery profiles, but generally a query language exposes a lot of the underlying data structures.

- *Requires intelligence on the client side*

  Depending on the goal of the client, it may be necessary to create very different queries in order to retrieve the desired links. The client would have to be intelligent enough to compose these queries, which — depending on the linkbase's design — can be hard.

Based on these observations it is our conclusion that a query-based access to linkbases is not the best approach. It is more advisable to model linkbase access as a service, where the server provides the service to deliver links based on the parameters specified by the client. This by no means restricts the way how servers actually implement the service, and this is one of the beauties of the service-based approach. In the simplest case, a linkbase server may directly use the parameters as input to XML technologies such as XQuery or *XSL Transformations (XSLT)* [10] in order to produce the links to be returned. On the other hand, servers may use very complex mechanisms to generate the links. For the client, this question of how the service is implemented is irrelevant, as long as the server returns the XLink-encoded links as requested.

In accordance with Internet design principles, the service interface should be as simple as possible. We believe that basically two kinds of requests can be differentiated, that must be supported if large linkbases should be usable via the linkbase access service:

- *Browsing the linkbase schema*

  For large linkbases, it is crucial that the links are structured according to a schema[2]. For example, a user might want to have all links that associate occurrences of a certain name where this name is participating as a "book author" in a link. However, in order to make such a request, it is necessary for the client to have access to the linkbase schema. The linkbase access service therefore must make it possible for a client to access and/or browse the linkbase's schema.

  If, for security reasons, the linkbase provider does not want to expose the linkbase's schema, it is of course possible to disable these services. However, for large and diverse linkbases, schema access will be crucial to enable users to explore and fully exploit the information stored in the linkbase.

- *Retrieving XLinks*

  Even though linkbase schema browsing may be essential to define reasonable filters for requesting links, the ultimate goal of linkbase access is the retrieval of links. Retrieving XLinks simply works by submitting a request, and as a result getting back an XML document (an XLink linkbase) containing all the links that satisfy the request's criteria.

While the data format for the second service (retrieving XLinks) is obviously XLink, the question becomes imminent which data model and format to chose for the access to linkbase schema information. XLink's support for link semantics is pretty simple by providing three attributes to carry application-specific semantic data (in the form of text and URI references). Applications may choose to use that data for semantic information that is associated (such as that the arc roles "daughter" and "son" are both sub-roles of the arc role "child"), but this is beyond XLink's scope.

[2]Here we do not speak of the schema of XLink itself, but a higher-level semantic schema that in many cases will be used to semantically describe the links. This schema can be thought of as *metadata* describing the links.

In theory, navigating through a linkbase's schema data could be very interesting and also would enable users to at least partly "understand" the linkbase's content. Linkbase schema data could be delivered in a number of way, such as XLink itself, *Resource Description Framework (RDF)* [27], or *Topic Maps* [33]. However, since XLink does not prescribe any particular schema format, we believe that linkbase schema access should be as simple as XLink's model of semantic information, which is simply sets of values. Any interpretation of these values is application-specific and should be processed by higher layers.

3.2 Underlying Protocol

While the service-based perspective is much better suited for linkbase access than the query-based (mainly it focuses more on the "what" of linkbase access than on the "how"), there still remains the question how to access the service over a network. This question involves questions which are important for all protocols which should be used on a large scale. Some of the issues include:

- *Support*

  It would not make sense to develop a completely new protocol, so it is very likely that existing reliable end-to-end protocols will be used as transport mechanism. Any transport protocol that should be considered should be supported by as many existing applications as possible, so that adoption is more likely.

- *Efficiency*

  A client may access several linkbases for every resource it is processing, so linkbase access should be as efficient as possible. In particular, it should be possible to use persistent connections.

- *Privacy*

  In some scenarios it may be desirable to protect the exchange of information, so an encryption mechanism should be supported.

- *Authentication*

  Linkbases may very well be offered commercially, in which case users will need to pay for linkbase access. In this case, authentication procedures are required, which also should be supported.

- *Caching*

  Caching should be supported, even though it is unclear how well caching will work in a scenario where most clients will compose unique queries.

- *Firewalls*

  Firewalls a very common today, and many of them are configured in a way which only permits a limited number of protocols to pass through. It should be taken care that the linkbase access protocol is not blocked by firewalls.

- *Redirection*

  While linkbase services are expected to have a long lifetime, it is very well possible that the service (if only for technical reasons) has to be moved. In this case, it should be

easy to redirect users of the service to the new location, and maybe also inform them whether this move is temporary or permanent.

Looking at this list of requirements, it becomes clear that HTTP is a very good candidate for the underlying protocol. Not only does it satisfy the requirements, it also has become the de facto standard for Web services, and in many cases the approach now is to choose HTTP unless there are really convincing reasons against it. We believe that HTTP is appropriate in our case, but this does not really solve the problem.

Layering services on top of HTTP can be done in different ways. Queries can be appended to the URI in the request, in which case the query needs to be encoded in URI syntax (HTML forms work this way). It is also possible to make services accessible by inventing new methods and/or header fields, which in fact extends the protocol (one example for this approach is WebDAV [22]). As another alternative, service access data can be encoded in the messages' entity field, in which case data often is encoded in XML (there are many examples for this which are collected by W3C's *XML Protocol* activity). All these alternatives have advantages and drawbacks, and while be are strongly in favor of using HTTP as the underlying protocol for linkbase access, we haven't yet decided in which way we will do it.

4 Existing Implementations

The service-based approach to linkbase access opens a very wide field for linkbase service implementations. In particular, any of the existing linkbase projects or products, such as DLS and Webvise mentioned above, could be easily adapted to the new access paradigm by adding a component that maps their internal data model to XLink and vice versa. Users could then transparently choose from different linkbases without the need to know how they are built internally.

Very simple implementations of linkbases could use XSLT to extract links from their internal linkbase (even though XSLT tends to be slow, so performance problems would be likely to occur), while more advanced linkbase services could use any information they want to, for example the client's capabilities and preferences as communicated through the *Composite Capabilities/Preference Profiles (CC/PP)* [30], in order to service the requests.

In fact, when thinking about accessing linkbases, it should always be kept in mind that XLink can be regarded as an exchange format for links on the Web, so nothing forces linkbase providers to use XLink internally. Linkbase providers can use whatever link model they see fit, the only thing they must implement is a mapping of their internal link model to the XLink Infoset and vice versa, so that they are able to process requests.

5 Requirements and Design Principles

Based on the observations made so far, we have compiled a list of requirements and design principles for the linkbase access protocol. In order to make the list more easily understandable, we have chosen to adopt the terminology from RFC 2119 [7], which is commonly used to indicate requirement levels.

- *Supported link model*

  The protocol MUST provide full support to the linking model of XLink. Servers MAY

choose to use other linking models internally, but if they do so MUST provide a mapping to the linking model of XLink.

- *Access modes*

  The protocol MUST NOT allow write access to linkbases. Even though write access is desirable for many interesting application scenarios (such as annotation systems), the requirements of managing write accesses to link bases are very complex and out of scope of our work. The protocol therefore MUST only allow read access to linkbases.

- *Adaptable to the existing Web infrastructure*

  The protocol SHOULD NOT require any changes in related standards. Existing technologies SHOULD be used as far as possible. However, because some of the required foundations are not yet in place (in particular, the XML Infoset), the protocol MAY be based on available draft documents.

- *Simplicity*

  To ensure quick adoption and implementation, easy understanding and maintainability of the protocol, the protocol SHOULD be as simple as possible, and the specification SHOULD be concise and easy to understand.

- *Extensibility*

  The protocol SHOULD be designed to be open for evolutionary steps in order to allow evolution of the implementing parties. In particular, even though the link model supported by the protocol itself is the XLink link model, it MUST be possible for implementations to exchange information about link model extensions, and to exchange information about the supported extensions. However, in order to avoid non-interoperable implementations, all implementations MUST support the XLink link model.

- *Openness*

  The protocol MUST be able to cover future extensions to XLink, without invalidating all existing implementations.

- *Related standards*

  Related standards SHOULD be considered and MAY be used as parts of the protocol if they satisfy all requirements. In particular, their stability, simplicity, and functionality need to be appropriately developed. The protocol SHOULD NOT define yet another query mechanism if the existing ones cover the requirements.

- *Transition to the new protocol*

  Both parties involved in a linkbase access, client and server, SHOULD NOT make assumptions whether the other party supports the linkbase access protocol. A request to a server that does not support the protocol SHOULD result in the delivery of the complete linkbase. On the other hand, a client that does not support the protocol SHOULD be able to request a linkbase from a server that implements the protocol. However, servers MAY refuse to respond with a complete linkbase for security or efficiency reasons.

- *Configuration language*

  User configurations for accessing linkbases include predefined queries or query parameters that are maintained by the client. These configurations can become complex, especially if they cover many different linkbases. There MUST be a language to submit configurations from a server to a client (eg, to communicate a configuration which has been determined via a Web-based user dialog), and from a client to a client (eg, if a user switches clients and wants to use the same configuration for linkbase access with both clients).

- *Generic links*

  XLink's data model does not explicitly include generic links. However, the protocol SHOULD include mechanisms so that generic linking applications are supported, for example by clients submitting content to the server, which then uses this content to compute the required links.

- *Access to every attribute in the XLink namespace*

  XLink's data model heavily depends on the XML attributes defined in the specification. The protocol MUST provide access to all these attributes and enable clients to request links based on filtering these values. However, the server MAY choose how to provide access to these attributes, or refuse to serve certain requests (eg, requests which would result in the whole linkbase being returned as a result).

- *Access to attributes of linking elements that are from a non-XLink namespace*

  XLink is open for extension. For example, XLink elements may have attributes with linking semantics that are not contained in the XLink namespace. Additionally, in future XLink versions, the set of global attributes in the XLink namespace might be extended. Therefore, it MUST be possible to specify conditions for non XLink-namespace attributes as well. However, the server MAY choose how to provide access to these attributes, or refuse to serve certain requests (eg, requests which would result in the whole linkbase being returned as a result).

- *Resources that are referenced in semantic attributes*

  XLink supports application linking semantics by defining attributes which must contain URI references to resources which then define the semantics. The format and interpretation of the resources is not part of the XLink specification. The linkbase access protocol MUST provide access to these linking semantics by supporting some kind of container format for conditions for the content of XLink's semantic attributes. Implementations are free to support any number of these additional semantics (including none). The protocol SHOULD NOT make any assumptions about the format of these semantics resources.

- *Restrict the size of the returned linkbase*

  Linkbases can be very big. Thus, accessing a linkbase may result in very many links being returned, and in order to avoid unnecessary exchange of data, clients MUST be able to specify the maximum size of the returned linkbase in terms of number of links and/or number of arcs and/or number of locators. Additionally, servers MAY use internal limits and refuse to serve requests which result in more links/arcs/locators than permitted.

- *Delivery of local resources*

  XLink supports local resources, which are resources contained in the link itself[3]. These resources may be of any size, so clients may want to exclude them. Consequently, clients MUST be able to suppress the delivery of local resources, in which case the server MUST strip the resources from the links before delivering them. The server MUST make sure that after stripping the local resources the links still remain valid (ie, there are no arcs pointing to stripped local resources).

- *Chained linkbases*

  XLink allows chained linkbases by using links within linkbases that point to linkbases. Servers MAY follow links to linkbases themselves (thus acting as clients in another linkbase access, creating a chain of linkbase accesses), or they MAY choose to not make chained requests and return the linkbase link to the client. Consequently, clients MUST be able to handle links to linkbases being returned from servers, and they MAY handle them as appropriate for the application (ie, either ignore them or use them for further linkbase accesses).

- *Processing steps to integrate query results must be provided*

  Linkbase accesses can be initiated because of different reasons, such as processing a resource referencing a linkbase, accessing a linkbase because of client configuration, or accessing a linkbase on user request. The protocol MUST define processing steps how to integrate the results of multiple queries.

- *Processing model*

  In general, the protocol MUST specify a processing model which describes how to process the data being exchanged. In particular, the connection between the linkbase schema (as represented by the attribute values of semantic attributes) and linkbase content MUST be clearly defined.

- *Usage scenarios*

  In order to make adoption and implementation of the protocol more easy, usage scenarios (including sample messages being exchanged) MUST be provided. This part probably is going to be informative only, but can nevertheless be crucial for the success of the protocol.

This list of requirements and design principles is rather long, and there are many possible design to satisfy them. We are currently working on specifying a protocol by defining services for accessing linkbases. When these services satisfy all our requirements and design principles, we will work on layering them on top of an existing transport mechanism. It is very likely that this will be HTTP, but the exact way in which this embedding will be done is an open issue.

6 Future Work

As mentioned already, the linkbase access protocol only is one of the missing pieces in the open linkbase access jigsaw. What also need to be done is work on the XLink Infoset, which

[3]Because XLink uses XML syntax, local resources are always XML resources.

we hope will be done by W3C in the near future. The issue of link presentation also need to be solved, and even though the W3C note [39] presents interesting concepts, more work needs to be done in this area. XPointer also has not yet reached recommendation status, but the specification seems to be stable, even though there are problems with its adoption by software suppliers.

On a different layer, the whole issue of linkbase schemas still needs to be explored, and this is a very wide and open field. From the linkbase access point of view, this can be regarded as a higher layer problem, which could be solved by a separate method for accessing linkbase schemas. Other words for this concept are linkbase metadata or linkbase ontology, and the number of activities currently investigating this kind of information is enormous. While there certainly already is a lot of knowledge about metadata/ontologies, this still needs to be brought to the level where it could be used by average users using their standard browser, and we believe that this would be a huge step towards the vision of the *Semantic Web*.

7 Conclusions

We believe that linkbases will become very popular with Web users, but this will only happen if the technology is there to make it happen. While the development of XLink is finished now, other parts of the jigsaw are still under construction or simply missing. This paper is our contribution to the issue of accessing linkbases through an open protocol, which is based on open data models. Our current work is focused on specifying and implementing a first prototype of our linkbase access protocol, and it is our goal to submit this protocol as a proposal to the Web community in the form of a W3C note.

The ideal way would then be to see the protocol being discussed by the Web community, re-design it, and then provide open implementations in the form of an Apache module and integration into Mozilla, so that real-world applications can be tested and evaluated. At this point in time, advanced linkbase interfaces like the one sketched by Weinreich et al. [40] could be implemented and tested on a large scale.

The lessons we learned when thinking about the requirements were the following: Even though "Open Hypermedia Systems" are "open" in the sense that they are open to create hypermedia structures separately from existing content, they are still "closed" in the communications systems sense of the word. The vision of "Open Communication Systems" includes open and standardized communication protocols and data formats, and it will take some more time until the Web is an "Open Hypermedia Systems" as well as an "Open Communications System".

References

[1] *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia*, San Antonio, Texas, May 2000. ACM Press.

[2] Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Paul Grosso, Eduardo Gutentag, R. Alexander Milowski, Scott Parnell, Jeremy Richman, and Stephen Zilles. Extensible Stylesheet Language (XSL) Version 1.0. World Wide Web Consortium, Recommendation REC-xsl-20011015, October 2001.

[3] Helen Ashman, Alejandra Garrido, and Harri Oinas-Kukkonen. Hand-Made and Computed Links, Precomputed and Dynamic Links. In Norbert Fuhr, Gisbert Dittrich,

and Klaus Tochtermann, editors, *Proceedings of Hypermedia — Information Retrieval — Multimedia 1997*, pages 191–208, Dortmund, Germany, September 1997.

[4] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identifiers (URI): Generic Syntax. Internet draft standard RFC 2396, August 1998.

[5] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu. XQuery 1.0: An XML Query Language. World Wide Web Consortium, Working Draft WD-xquery-20011220, December 2001.

[6] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. CSS2 Specification. World Wide Web Consortium, Recommendation REC-CSS2-19980512, May 1998.

[7] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels. Internet Best Current Practice RFC 2119, March 1997.

[8] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, Recommendation REC-xml-20001006, October 2000.

[9] Leslie A. Carr, David C. De Roure, Wendy Hall, and Gary J. Hill. The Distributed Link Service: A Tool for Publishers, Authors and Readers. In *Proceedings of the Fourth International World Wide Web Conference*, pages 647–656, Boston, Massachusetts, December 1995.

[10] James Clark. XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, Recommendation REC-xslt-19991116, November 1999.

[11] John Cowan and Richard Tobin. XML Information Set. World Wide Web Consortium, Recommendation REC-xml-infoset-20011024, October 2001.

[12] Hugh C. Davis. Referential Integrity of Links in Open Hypermedia Systems. In *Proceedings of the 9th ACM Conference on Hypertext*, pages 207–216, Pittsburgh, Pennsylvania, June 1998. ACM Press.

[13] Hugh C. Davis. Hypertext Link Integrity. *ACM Computing Surveys*, 31(4), December 1999.

[14] Hugh C. Davis, David E. Millard, Siegfried Reich, Niels Olof Bouvin, Kaj Grønbæk, Peter J. Nürnberg, Lennert Sloth, Uffe Kock Wiil, and Kenneth M. Anderson. Interoperability between Hypermedia Systems: The Standardisation Work of the OHSWG. In *Proceedings of the 10th ACM Conference on Hypertext*, pages 201–202, Darmstadt, Germany, February 1999. ACM Press.

[15] David C. De Roure, Nigel G. Walker, and Leslie A. Carr. Investigating Link Service Infrastructures. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia* [1], pages 151–160.

[16] Steven J. DeRose, Eve Maler, and Ron Daniel. XML Pointer Language (XPointer) Version 1.0. World Wide Web Consortium, Candidate Recommendation CR-xptr-20010911, September 2001.

[17] Steven J. DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[18] Samhaa El-Beltagy, Wendy Hall, David C. De Roure, and Leslie A. Carr. Linking in Context. In Proceedings of the 12th ACM Conference on Hypertext and Hypermedia [34], pages 151–160.

[19] Jon Ferraiolo. Scalable Vector Graphics (SVG) 1.0 Specification. World Wide Web Consortium, Recommendation REC-SVG-20010904, September 2001.

[20] Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. Internet proposed standard RFC 2616, June 1999.

[21] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart. HTTP Authentication: Basic and Digest Access Authentication. Internet proposed standard RFC 2617, June 1999.

[22] Y. Goland, E. James Whitehead, A. Faizi, S. Carter, and D. Jensen. HTTP Extensions for Distributed Authoring — WebDAV. Internet proposed standard RFC 2518, February 1999.

[23] Kaj Grønbæk, Lennert Sloth, and Niels Olof Bouvin. Open Hypermedia as User Controlled Meta Data for the Web. In *Proceedings of the Nineth International World Wide Web Conference*, pages 553–566, Amsterdam, Netherlands, May 2000. Elsevier.

[24] Kaj Grønbæk, Lennert Sloth, and Peter Ørbæk. Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the World Wide Web. In *Proceedings of the Eighth International World Wide Web Conference*, pages 253–267, Toronto, Canada, May 1999. Elsevier.

[25] Kaj Grønbæk and Uffe Kock Wiil. Towards a Common Reference Architecture for Open Hypermedia. *Journal of Digital Information*, 1(2), 1997.

[26] Brent Halsey and Kenneth M. Anderson. XLink and Open Hypermedia Systems: A Preliminary Investigation. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia* [1], pages 212–213.

[27] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium, Recommendation REC-rdf-syntax-19990222, February 1999.

[28] Dave Millard, Hugh C. Davis, and Luc Moreau. Standardizing Hypertext: Where Next for OHP? In Siegfried Reich and Kenneth M. Anderson, editors, *Proceedings of the 6th Workshop on Open Hypermedia Systems and Structural Computing*, volume 1903 of *Lecture Notes in Computer Science*, pages 3–12, San Antonio, Texas, June 2000. Springer-Verlag.

[29] Luc Moreau and Wendy Hall. On the Expressiveness of Links in Hypertext Systems. *The Computer Journal*, 41(7):459–473, 1998.

[30] Mikael Nilsson, Johan Hjelm, and Hidetaka Ohto. Composite Capability/Preference Profiles: Requirements and Architecture. World Wide Web Consortium, Working Draft WD-CCPP-ra-20000721, July 2000.

[31] Peter J. Nürnberg and John J. Leggett. A Vision for Open Hypermedia Systems. *Journal of Digital Information*, 1(2), 1997.

[32] Steven Pemberton. XHTML 1.0: The Extensible HyperText Markup Language. World Wide Web Consortium, Recommendation REC-xhtml1-20000126, January 2000.

[33] Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification xtm1-20010806, August 2001.

[34] *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*, Århus, Denmark, August 2001. ACM Press.

[35] Eric Rescorla. HTTP over TLS. Internet informational RFC 2818, May 2000.

[36] Paul Thistlewaite. Automatic Construction and Management of Large Open Webs. *Information Processing and Management*, 33(2):161–173, 1997.

[37] Janet Verbyla. Unlinking the Link. *ACM Computing Surveys*, 31(4), December 1999.

[38] Fabio Vitali. Versioning hypermedia. *ACM Computing Surveys*, 31(4), December 1999.

[39] Norman Walsh. XML Linking and Style. World Wide Web Consortium, Note NOTE-xml-link-style-20010605, June 2001.

[40] Harald Weinreich, Hartmut Obendorf, and Winfried Lamersdorf. The Look of the Link — Concepts for the User Interface of Extended Hyperlinks. In Proceedings of the 12th ACM Conference on Hypertext and Hypermedia [34], pages 19–28.

[41] Erik Wilde and David Lowe. From Content-Centered Publishing to a Link-Based View of Information Resources. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, January 2000. IEEE Computer Society Press.

[42] Erik Wilde and David Lowe. *XPath, XLink, XPointer, and XML: A Practical Guide to Web Hyperlinking and Transclusion*. Addison Wesley, Reading, Massachusetts, 2002.