Freenet-like GUIDs for Implementing Xanalogical Hypertext

Tuomas J. Lukka
Hyperstructure Group
Dept. of Mathematical Information Technology
University of Jyväskylä, PO. Box 35
FIN-40351 Jyväskylä
Finland
lukka@iki.fi

Benja Fallenstein
Oberstufen-Kolleg
University of Bielefeld, PO. Box 100131
D-33501 Bielefeld
Germany
b.fallenstein@gmx.de

ABSTRACT

We discuss the use of Freenet-like content hash GUIDs as a primitive for implementing the Xanadu model in a peer-to-peer framework. Our current prototype is able to display the implicit connection (transclusion) between two different references to the same permanent ID. We discuss the next layers required in the implementation of the Xanadu model on a world-wide peer-to-peer network.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*architectures*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems, information networks*; C.4 [**Performance of Systems**]: [fault tolerance, reliability, availability, and servicability]

General Terms

Design, Reliability, Security, Performance

Keywords

P2P, Xanadu, Permanence, Transclusion

1. INTRODUCTION

(Non-)Permanence of content is a growing problem on the Internet, and various attempts have been made to alleviate it[4, 2, 12].

The Xanadu hypermedia model[7, 8, 9] handles the problem by assigning a permanent ID to content when it first enters the system. The immutable, permanent ID relates to the *physical act* of entering the smallest units of data, as in "the character 'D' typed by Janne Kujala on 10/8/97 8:37:18". Mutable documents are represented as *virtual files* containing lists of permanent media IDs. Copying content from one document to another is done by referring to the same permanent IDs.

In the Xanadu model, all virtual files containing a given piece of content (*transclusion*) can be found and displayed to the user. For example, an email quoting another email would be automatically and implicitly connected to the original via the transclusion. Bidirectional, non-breaking external links (*content linking*) can be resolved through the same mechanism. Nelson[9] argues that conventional software, unable to reflect such interconnectivity of documents, is unsuited to most human thinking and creative work.

In order to implement the Xanadu model, it must be possible to efficiently search for references to permanent IDs on a large scale. The original Xanadu design organized content IDs in a DNS-like hierarchical structure (tumblers), making content references arbitrary intervals (spans) in the hierarchy. Advanced tree-like data structures[6] were used to retrieve the content efficiently. Unfortunately, Project Xanadu's implementation was never finished and it is unclear whether the tumbler model can be implemented securely to avoid e.g. spoofing attacks.

In this article, we discuss a less ambitious structure based on Freenet-like GUIDs. This allows distributed hashing[11, 14] to be used for looking up references to permanent IDs.

2. FREENET: GLOBALLY UNIQUE IDS

Freenet[1, 2] is a decentralized P2P (peer-to-peer) architecture for anonymous uncensorable publishing. Data is stored in immutable sequences of bytes which are identified by SHA-1 cryptographic hashes[10] of their contents. Since in practice SHA-1 hashes are unique, they can be assigned as Globally Unique IDs (GUIDs) without needing a central naming authority.

GUIDs are like Uniform Resource Names[13]: they do not specify a physical storage location but an *identity*. In such a system, the primitive operation is *not* "get me file X from location Y" but "get me the data X, whereever it may be".

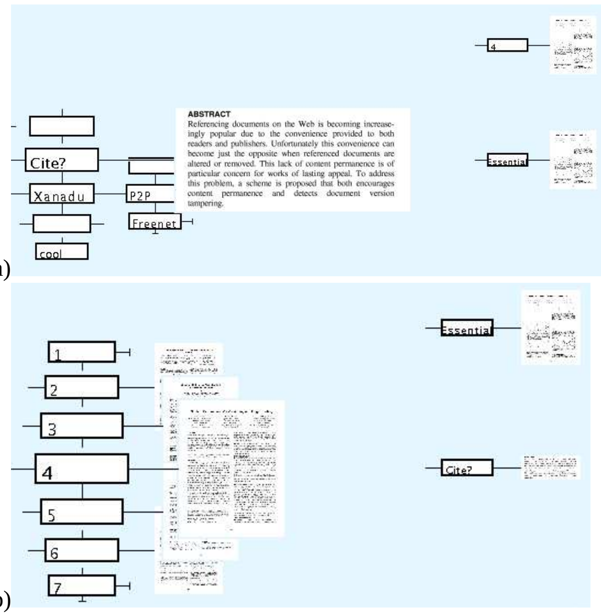
3. XANADU-MODEL HYPERTEXT ON TOP OF FREENET-LIKE GUIDS

While Freenet focuses on anonymity and uncensorability, the Xanadu model focuses on external linking, implicit connections via transclusions, and permanence. Despite the differences, Freenet-like GUIDs provide a useful lowest layer of abstraction for implementing the Xanadu model.

In this model, a block of media content (e.g. keystrokes) obtains a GUID when first saved to permanent storage. The byte sequence also contains metadata such as the author and the creation time[1]. All documents transcluding the saved keystrokes use the GUID and an offset inside the block.

[1]Metadata that is subject to change should naturally be stored outside the immutable block; in our model, it would be stored in other immutable blocks with a revocation or expiration mechanism.

HT'02, June 11-15, 2002, College Park, Maryland, USA.



a) b)

Figure 1: Two screenshots of Gzz: a) Three transclusions from an article and cells connected to each transclusion. b) After moving the focus to another one of the transclusions in a) and zooming out. (The images have been edited to fit the page.)

With GUIDs, searching for the references can be implemented through distributed hashtables[11, 14], mapping a block's GUID to a list of references to that block. (Tumblers, being intervals, could not be used as hashtable keys.)

One of the advantages of using permanent blocks with GUIDs as a primitive is robustness. If a conventional application crashes while saving a file, the whole file can get mangled. In our design, blocks with GUIDs are used to emulate append-only storage such as CD-R. Only the difference between the previous and the current version is stored in a new permanent block. Even if the software crashes, the older blocks will not be harmed.

The block-level granularity can cause complications when publishing a block: if it contains something that the user does not want to distribute, they must either create a new, Bowdlerized block and refer to that, or make a *surrogate* block containing the publishable parts and their permanent IDs. Trust that a surrogate block contains the correct contents for an ID depends on digital signatures and requires a public key infrastructure. Checking the cryptographic hash is not possible, as only part of the original block is available. Also, to counter a brute-force attack for finding out the non-distributable parts of the block, 20 cryptographically secure random bytes should be included in the beginning of every block and never published in surrogate blocks.

Being able to assign secure, permanent identities to the individual letters would solve this problem, but there is no known way to do this efficiently.

4. THE GZZ IMPLEMENTATION

The Gzz prototype[5] uses globally identified, immutable byte sequences for all permanent data. While the prototype is not yet able to operate on a world-wide P2P network, low-level synchronization among users can be performed without a central server by simply copying the blocks.

Figure 1 shows two screenshots of an annotated bibliography of hypertext publications. The focus+context[3] view shows two things as the context of a transclusion from a PDF file: users' annotations, which have explicitly been connected to it; and different transclusions of the same content, only implicitly connected to the focused transclusion through the Xanadu model.

This functionality is currently in a pre-alpha stage, but we expect to release a first version of it as a part of Gzz 0.8.0 (*"chartreuse"*) shortly (available at [5]).

5. CONCLUSIONS

We have shown how a subset of the Xanadu media model can be implemented using Freenet-like GUIDs. Our current prototype is still limited: to approach the full functionality of the Xanadu model, content links have to be implemented, and the system needs to be extended to 1) fetch data interactively through a P2P network, and 2) use distributed indices of references to permanent IDs.

The permanence of the Xanadu model affects performance in several ways: while checking hashes and assembling the content from different blocks may be slow, caching is on the other hand easier. More research on performance issues both on a single computer and on a network is necessary.

In addition to the implementation of the Xanadu model, research is also needed on visualizing and navigating the structures arising from it.

6. REFERENCES

[1] I. Clarke et al. The Freenet project. http://www.freenetproject.org/.

[2] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Protecting free expression online with Freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[3] G. W. Furnas. Generalized fisheye views. In *Proceedings of CHI '86*, pages 16–23. ACM Press, 1986.

[4] The Internet Archive: The Wayback Machine. http://web.archive.org.

[5] T. J. Lukka et al. Gzz. http://gzz.info/, http://savannah.gnu.org/projects/gzz.

[6] T. Nelson. Xanadu® technologies– an introduction. http://xanadu.com/tech/, 1999.

[7] T. Nelson. The Xanadu® vision. http://xanadu.net/xuvision.html, 1999.

[8] T. Nelson. Deep hypertext: the Xanadu® model. http://xanadu.net/xuTheModel/, 2001.

[9] T. H. Nelson. Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31(4es), 1999.

[10] NIST. *FIPS PUB 180-1: Secure Hash Standard*, 1995.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[12] J. Simonson, D. Berleant, and A. Bayyari. Content permanence via versioning and fingerprinting. In *Proceedings of the eleventh ACM conference on Hypertext and hypermedia*, pages 226–227, 2000.

[13] K. Sollins and L. Masinter. Functional requirements for Uniform Resource Names. RFC 1737, IETF Network Working Group, 1994.

[14] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications, 2001.