TinLizzie WysiWiki and WikiPhone: Alternative approaches to asynchronous and synchronous collaboration on the Web

Yoshiki Ohshima[†]
yoshiki@squeakland.org

Takashi Yamamiya[†]
tak@metatoys.org

Scott Wallace[†]
scott.wallace@squeakland.org

Andreas Raab[‡]
andreas.raab@qwaq.com

[†]*Viewpoints Research Institute*
1209 Grand Central Ave.
Glendale, CA 91202

[‡]*Qwaq, Inc.*
460 S California Ave #304
Palo Alto, CA 94306

Abstract

This paper presents TinLizzie WysiWiki and WikiPhone, two systems which explore new approaches to media-rich end-user collaboration on the World Wide Web.

TinLizzie WysiWiki enables authoring of interactive, media-rich documents, containing graphical objects bearing user-defined scripts, on the Web. In TinLizzie WysiWiki, a user manipulates text and active objects in a WYSIWYG graphical editor in a manner similar to Squeak eToys.

A notable aspect of TinLizzie WysiWiki is that it allows both synchronous and asynchronous collaboration among multiple users. In asynchronous collaboration, the user content is saved in a common format and posted on the Web. Later, another user can visit and update the document on the server. In synchronous collaboration, more than one user can share a document and edit it simultaneously and collaboratively in real-time.

The second system presented is called WikiPhone. WikiPhone is a minimalist voice over IP (VoIP) system which uses only HTTP. WikiPhone allows multiple users to talk to each other using a web browser. WikiPhone's strength is its simplicity in terms both of the user experience and of its implementation. The user simply points a web-browser to a URL, and then directly participates in a conversation, or listens to ongoing conversations among others. In the implementation, it seeks the simplest possible approach, yet provides a usable VoIP system.

Both systems still require a small, portable web browser plugin, but otherwise they stay within the artificial limitations of today's World Wide Web. The authors think that they exhibit possible future directions for collaboration on the Web.

1 Introduction

The World Wide Web, or Web, has been very successful. It seems nowadays to dictate not only many end-users' behavior, but also the mindsets of researchers and software vendors; the perception is that a new system should run in web browsers to be successful. As a consequence, improvement in the overall user experience in applications has been held back by limitations of the Web and of commonly used Web browsers. For example, an application written in JavaScript runs much more slowly than it should, and has limited graphics capability. Considering the fact that the computer's capability and performance have improved dramatically year after year, it is noteworthy that the responsiveness of an application like an email client on the web is slower today than a typical email client was years ago – as if a perverse kind of inverse Moore's law were at play.

Two particular areas in which the authors think typical web applications fall short are *end-user multimedia scripting* and *real-time collaboration*. There is no end-user scripting system in which a non-technical user can create graphical objects, specify their behavior in an end-user accessible programming system, and share the result on the Web. Kay stated his criticism on the Web in the discussion page about the Logo programming language on Wikipedia [1]:

It really bothers me that the Logo examples in this article can't be tried out in 2006 (given that this was easy on an Apple][in 1979)!

Also, there is no system written in JavaScript running in a browser that will allow users at remote locations to talk to or see each other. Given the fact that the NLS system [2] in '68 already proved the feasibility of real-time collaboration on the computer, computer researchers and industry should have made it possible by now after four decades.

In this paper, we present two systems that explore the possibilities of end-user multimedia scripting and voice chat in Web browsers.

One of the systems, called TinLizzie WysiWiki, enables the authoring of media-rich content within Web browsers, and sharing among users. The content editing experience resembles Squeak eToys' [3], but the content is shared by multiple users. The user can put a hyperlink to other documents, so that it is possible to create a network of linked projects in a spirit similar to that of WikiWikiWeb. More notably, TinLizzie WysiWiki provides real-time collaboration; multiple remotely located users can edit the same document at the same time collaboratively.

The second system presented, WikiPhone, is a minimalist implementation of a voice over IP system. It exhibits how simple a VoIP system can be. Given this minimal effort, the user experience is surprisingly satisfactory.

While both systems require a small browser plugin to implement otherwise impossible features, they interoperate nicely with existing systems and standards. This approach gave us interesting trade-offs and constraints. In the remainder of this paper, more background is given in section 2, the design and trade-offs of TinLizzie WysiWiki and WikiPhone are given in section 3 and section 4, respectively, and a discussion of the results is given in section 5.

2 Background and Related Work

In this section, the motivation of the work, and of related work in similar domains, is discussed.

As described in the previous section, the Web has been very popular. However, because it wasn't designed to allow bi-directional, symmetric communication, nor with content authoring in mind, most Web-based applications operate under various artificial limitations.

On the other hand, being on the Web certainly has practical benefits. A web browser is indeed a primary application platform for virtually every Internet user. And modern web browsers cope well with some kinds of artificial restrictions such as Internet firewalls. Also, schools and corporations tend to have policies which make it difficult, if not impossible, to install new software. Since the JavaScript engines are already ubiquitous, an application written in JavaScript can run with zero-installation effort. It also means that software update can be done on the server side.

There have been various attempts to enable some form of authoring of Web content within Web browsers. The WikiWikiWeb, or Wiki, system [4] takes advantage of the "Form" feature of the browsers to let end-users edit textual content on the Web.

From the user's point of view, the strength of a Wiki is that hyperlinked text, or hypertext, seems to be a well-accepted description of a set of knowledge; a set of knowledge is a network of inter-dependent concepts, so hyperlinked text seems to match with it well. The success of Wiki motivated a lot of "clones" which provide different features and different interfaces to try to improve the user experience. However, it is clear that a Form-based Wiki system will not reach a threshold that provides an appropriate comfort level for many users. The problem is that the editing is not "direct"; instead, the user has to type in text that follows somewhat cumbersome markup rules, and then check the rendered result afterwards. For this problem, some recent Wiki systems implement a kind of WYSIWYG editor in JavaScript [5]. These systems are often dubbed "WysiWiki" or similar phrases.

While these attempts have been helpful for the near term, we should not be held hostage to the artificial limitation that only text (and perhaps some static pictures) can be used. The kinds of content our group is interested in are the ones with multi-media and, more importantly, with graphical objects with user-defined behavior.

Let us imagine an article in an Internet-based encyclopedia that discusses, for example, a topic in physics of some phenomenon. The writer of the article might wish to have a simulation of the phenomenon. To write such a simulation, the author creates some graphical objects and write rules to specify the behavior of them. When a reader visits the page, the graphical objects follow the rules, or *scripts*. and will move around, change size and color, interact with one another, appear and disappear. Note that this should not be limited to canned animations; the scripts should be accessible to the readers as well. With such interactive scripts, a reader can try "what-if simulations", and refine the behavior so that he can gain a deeper understanding of the phenomenon. Along this line, our group experimented with a Logo execution engine which, because it was written in JavaScript, is accessible from any web browser [6]. This system is effective at what it sets out to do, but it is limited to a turtle and lines.

Another drawback of a typical Wiki system (or, we might say, another design trade-off to accommodate Web limitations) is that its mode of collaboration is limited to "asynchronous". Even if there are two or more users looking at the same article, they cannot interact with each other. They might want to paint a picture together or construct different parts of a simulation together. There are a few systems written in JavaScript for real-time collaborative editing on the Web which begin to address such desires, but these are usually limited to simple text or line drawing [7].

A natural expectation for real-time collaboration is that users should be able to have voice conversations with each other. The phone system has provided this ability all of our lives, but it is costly and cumbersome. On the Internet, there are now a number of Voice over IP (VoIP) telephony systems such as Skype and Gizmo Project. Good VoIP systems are already mature and have been used by millions of users. However, one might wonder, out of curiosity, how simple a VoIP system can be.

WikiPhone tries to provide the simplest imaginable implementation of a VoIP system, with a minimum of features. Feature-laden commercial VoIP systems do not necessarily suit the needs of all users. A much lighter-weight VoIP such as WikiPhone may suit users wanting direct web-based voice conversations without authentication ("log in"), or simply to listen. It provides an experience similar to CB Radio.

3 TinLizzie WysiWiki

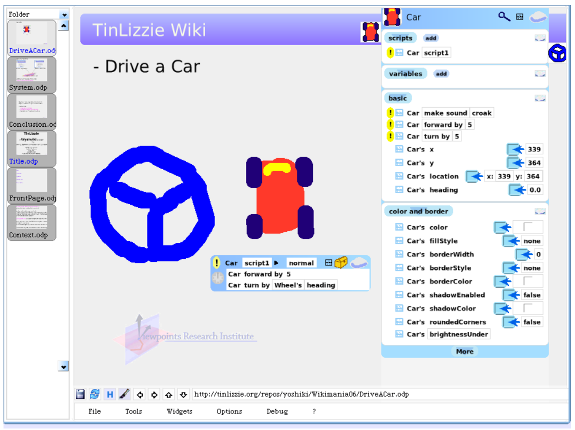
The characteristics of TinLizzie WysiWiki can be summarized as follows:

Media-rich authoring with scripting Content is authored in an eToys-like environment [3]. All objects can be edited in a WYSIWYG fashion, by direct manipulation with a pointing device, and can be scripted via tile-based scripts. The result is saved in a document file and shared on a server so that other users can open and see it. The document appears the same to the user as it did to its author as he edited it.

Interoperability User content is saved in an Open Document Format (ODF) [8] so that OpenOffice.org applications can open it. Some actual editing can be undertaken using OpenOffice as well. The content transmission between the server and the client is done in WebDAV (for reading/writing) or HTTP (for reading only). The system requires some small support from a browser plugin, and is built on top of the highly portable Squeak virtual machine [9] so that it runs in the majority of web browsers, on all major platforms, and also as a standalone application.

Asynchronous and Synchronous Collaboration In addition to traditional Wiki style (asynchronous) collaboration, real-time, or synchronous, collaboration is also supported. Multiple users can edit a document simultaneously. (The network transport for synchronous communication doesn't use HTTP in the current implementation.)

In the following, these items are explained.



Car script1 normal
Car forward by 5
Car turn by Wheel's heading

Car script1
Car make sound croak
Car forward by 5
Car turn by 5
Car's x 339
Car's y 364
Car's location x: 339 y: 364
Car's heading 0.0
Car's color
Car's fillStyle none
Car's borderWidth 0
Car's borderStyle none
Car's borderColor
Car's shadowEnabled false
Car's shadowColor
Car's roundedCorners false
Car's brightnessUnder

http://tinlizzie.org/repos/yoshiki/Wikimania06/DriveACar.odp

Figure 1. A screenshot of the TinLizzie WysiWiki user interface.

3.1 Media-rich authoring with scripting

The TinLizzie WysiWiki system is built on top of an object system called Tweak [10]. Tweak draws upon Squeak [9], the Morphic graphical user interface framework [11] [12] and eToys [3]. Tweak not only tries to overcome various limitations of Morphic and eToys, but also, more notably, it provides a foundation for concurrent programming and distributed computing.

On top of the core architecture of Tweak which provides the object model, the authors and colleagues have implemented an eToys-like end-user scripting system. In that system, the user can create graphical objects, and modify the behavior and the properties of the objects they create. It is important to support this "exploration style of programming", as the user often doesn't have a clear view of the final results in advance. As is the case with eToys, the scripting system of TinLizzie WysiWiki allows the user to modify the behavior of existing objects freely, and the user gets feedback immediately because all objects in the system are constantly active.

A library of pre-defined prototypes of graphical, multimedia objects that the user can instantiate is provided. Such objects includes a movie player, a sound recorder and player, a painting tool, a text field, etc. Also, a set of widgets that support a programming system for massively parallel particles similar to the Kedama system [13] is implemented. With this rich set of objects, the user can create a wide variety of multimedia content.

3.2 Interoperability

The authors wanted TinLizzie WysiWiki to run on all commonly-used platforms, so we tried to make the system be compatible with commonly used systems and standards, as much as possible. As for portability, TinLizzie WysiWiki is built on top of the Squeak virtual machine (Squeak VM). Squeak VM has been ported to more than two dozen platforms, ranging from the commonly used platforms such as Windows, Linux, and Macintosh to some exotic platforms such as various PDAs and the prototypes of the OLPC laptop [14]. One of the notable features of Squeak is that a program written for the Squeak VM runs "identically" everywhere; the pixels on the screen are guaranteed to be the same across all platforms. Also, the browser plugin version of the Squeak VM allows TinLizzie WysiWiki to run in most commonly-used web browsers. The user simply points a web browser to a URL to access content in TinLizzie WysiWiki. When the user accesses a page on the server, a document file is downloaded to the client side and an editing session takes place on the user's machine. This requires the program or plugin to be installed on the computer, but enables us to write a system with multimedia capability that runs efficiently.

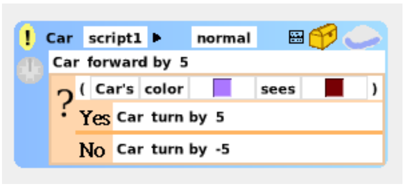


Figure 2. Graphical view of a script in TinLizzie WysiWiki.

TinLizzie WysiWiki documents are stored externally in a standard ODF format. ODF has a few variants for presentation, spreadsheet, and text processing, and we have chosen to use the "presentation" variant as the primary format. This decision was made because a project in Tweak eToys tends to be graphics-oriented, with all the content typically residing on a single screen. Our ODF converter in Tweak converts the Tweak eToys objects into the XML-based ODF. This can be a lossy conversion in the current implementation; not all available Tweak eToys objects, and not all properties of some, are perfectly preserved at present, but future work should repair this defect.

When saving a user project, one of the biggest problems was how to externalize user-defined scripts. ODF defines a way to store program fragments (typically in Basic or JavaScript), but the primary representation of a script in TinLizzie WysiWiki is a graph of live objects. This is because what the user manipulates is the graphical objects that represent the syntax elements, so that the natural representation of them is to keep them in the dynamic form. Figure 2 shows an example script. The elements in the script can be moved around by the user to edit it. In the current implementation of TinLizzie WysiWiki, a script is first rendered into equivalent Smalltalk textual code, and then stored within the XML as an attribute of a node. Upon loading, the textual code is converted back to the Tweak's object structure. Along with the textual representation, a static picture of each scriptor is stored into ODF as an element of the document. Thus, when the ODF file is opened with the Impress OpenOffice.org application, the visual look of the project is reproduced, including the graphical appearance of the scripts.

The ODF file created is then sent to the server. Since the transport of documents is done via WebDAV, the user doesn't have to change any settings such as proxies. The server could be an ordinary Web server, but a feature we wanted was version control of documents. We decided to use the SVN version control system with Apache (via `mod_dav_svn`).

The user can put hyperlinks in the document. The link is a local name of document or a fully qualified URL. When a reader of the document clicks on the hyperlink, the specified document is fetched from the server or the local disk and opened.

So far, the typical use of TinLizzie WysiWiki has appeared to be similar to that of Wiki. A user starts editing, and submits the change. Then another user takes the revised version from the server, and continues. This can be called "asynchronous" collaboration. The benefit of asynchronous collaboration is that the users don't have to be working at the same time, but an obvious downside is that they cannot communicate with fast turn-around.

3.3 Asynchronous and Synchronous Collaboration

In addition to asynchronous collaboration, TinLizzie WysiWiki supports synchronous, or real-time, collaboration. When two or more users find that they are editing the same page at the same time, one can "invite" others to his page. Then, his page is serialized and sent to the invited users. In other words, they get "identical" working snapshots of the page. From there, all users' interactions are sent to all other participants' computers; thus the computations on all participants' computers can be carried out identically.

This real-time collaboration model is called the Simplified TeaTime (STT) mechanism of Croquet [15] [16]. The Croquet project is known to have a 3D graphical interface, but the underlying object model doesn't have any dependency on an object's graphical appearance. Croquet's object model is unified with Tweak's in the TinLizzie WysiWiki system, so that any Tweak eToys object can behave as a Croquet object. This mechanism of distributed computation relies on the fact that if the computation is started from an identical state on different computers and if there are no

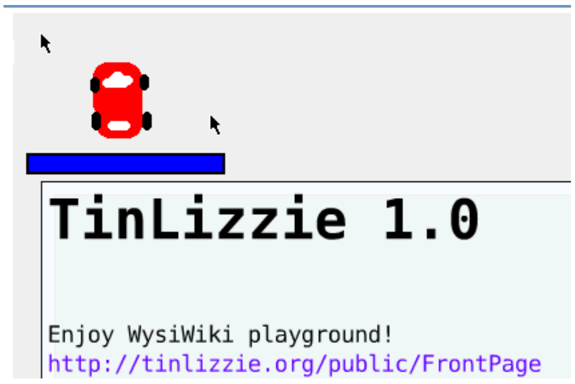

TinLizzie 1.0
Enjoy WysiWiki playground!
http://tinlizzie.org/public/FrontPage

Figure 3. Two users are interactively editing a WysiWiki page.

non-deterministic events, the computation will be identical on different computers as is advances. And if a user event is time-stamped properly and delivered to the other computers at the "same (pseudo) time", the invariant of identical computation will be kept.

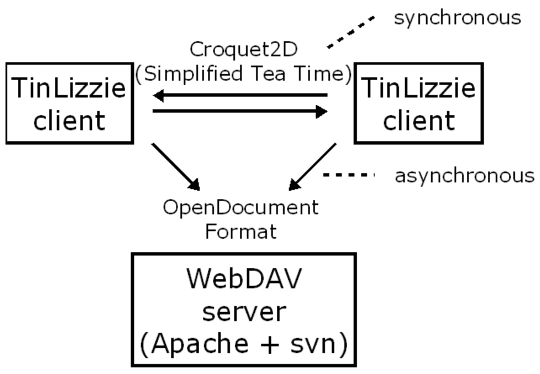


Figure 4. A schematic view of network topology and data format used in TinLizzie WysiWiki.

In a system with STT, the network traffic is very low; only the user events and periodic "heartbeat" events need to be sent. Currently, a typical user event is encoded as a packet whose size is 170-200 bytes. When the user drags an object for example, 40-50 packets are sent for every second. This results in 7k bytes/sec to 9k bytes/sec network traffic. Obviously, a mouse event wouldn't need that much data and we could optimize it; on the other hand, as long as a packet fits in an Ethernet packet, such optimization wouldn't make as much improvement. The latency is dictated simply by the underlying network. The latency is comparable with the results from "ping" program. A user in the group sees the other participants' cursors on his screen and sees the cursors making changes simultaneously. Once the editing is done, a participant can save the document in ODF. Figure 3 shows that there are two cursors on a computer screen.

Figure 4 depicts the communication protocols and data format used in TinLizzie WysiWiki.

4 WikiPhone

The characteristics of the WikiPhone system can be summarized as follows:

Interoperability The transmission of voice data is done solely in HTTP, which minimizes firewall problems. The system requires a browser plugin, but it is the same Squeak virtual machine used by eToys and TinLizzie WysiWiki.

Simplicity The user interface is extremely minimal. The user only points his web browser to a specific URL, and voice conversations become instantly possible with any other users who are on the same URL.

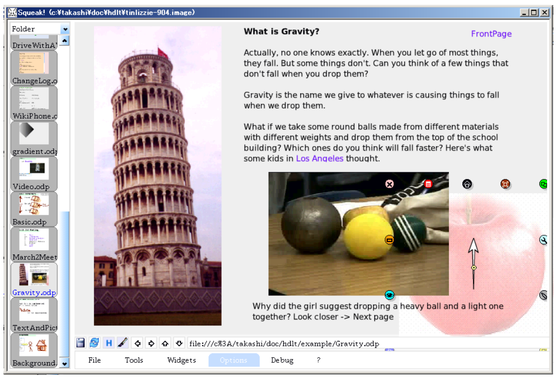
In the following, these items are explained.

4.1 Interoperability

The implementation can be considered an experiment in how simple a VoIP system can be. A typical VoIP system has the concept of sessions, and uses protocols such as SIP [17]. These systems are modeled after the telephone system. However, if the purpose is just to transmit voice data two ways and have conversations, the layers upon layers of protocols become unnecessary.

The approach that WikiPhone took was to use two HTTP/1.1 connections between a client and a server because HTTP protocol can only handle one way connection. To make the latency shorter, HTTP chunked data encoding is used. The small plugin program running on the user's browser compresses the audio from the microphone input and sends the data to a server. The server simply distributes incoming data packets to different clients listening to the URL, and it doesn't modify the packets itself. This approach could be easily extended to other streaming media such as video. The client program continuously fetches data packets from the URL and plays them back on the client computer. The URL can be thought of as a radio channel that a user can not only tune in to, but also talk back on.

The server program is written in Squeak, which provides a variety of sound codecs. Currently the audio data is generally compressed with the GSM codec that gives about 10:1 compression rate for 22 kHz 16-bit mono audio. The server gets about 4k bytes data from per second from each client. Thus, the network traffic is a bare minimum.



What is Gravity?

Actually, no one knows exactly. When you let go of most things, they fall. But some things don't. Can you think of a few things that don't fall when you drop them?

Gravity is the name we give to whatever is causing things to fall when we drop them.

What if we take some round balls made from different materials with different weights and drop them from the top of the school building? Which ones do you think will fall faster? Here's what some kids in Los Angeles thought.

Why did the girl suggest dropping a heavy ball and a light one together? Look closer -> Next page

Figure 5. A page that explains gravity with a scripted simulation.

The number of users who can be on a server is limited by the server-side network capacity. Because any heavy computation like compression and sound mixing is done on the client side, the server's computation cost is not a primary performance issue. Furthermore, no centralized database is needed.

Latency is the largest issue with this protocol. Our experiments have shown that it is longer than 300ms for shorter-distance conversations, and longer than 500ms for inter-continental conversations. Conventional VoIP systems use UDP as their transportation layer to reduce latency. But WikiPhone uses TCP to take advantage of extensibility and interoperability.

WikiPhone still requires a browser plugin when it is used in web browser directly, because neither JavaScript nor even a common plugin such as Flash supports sound synthesis. While having to require a browser plugin is not ideal, the plugin is again a Squeak VM and the code of WikiPhone is downloaded from the server on demand. Many features that makes a VoIP system practical are missing, but given its simplicity, it works strikingly well.

4.2 Simplicity

In a typical use of WikiPhone, the user simply points his Web browser to a URL, then starts the conversation right away. No login procedure is required. If you coordinate with your friends, you can have effortless conversations, with an interface affording ambient presence of others.

The protocol used for WikiPhone is also simple. It is designed as a module just for transport streaming media on top of HTTP. There is no authentication, and no security management. This is reasonable because HTTP already has various kinds of such methods, such as the TLS layer. The WikiPhone protocol can be combined with them if it is needed.

5 Discussion and Future Work

In this section, the current status of the projects and possible future work are discussed.

5.1 TinLizzie WysiWiki

5.1.1 WYSIWYG

TinLizzie WysiWiki aims to be a WYSIWYG editor. The promise of a WYSIWYG editor, as the acronym suggests, is that the author can closely control the final appearance of the document by editing content in a form closely resembling the form in which it is to be viewed. However, perfect WYSIWYG is a partial fiction, because, for example, the pixel resolution and color characteristics of various readers' screens will vary. (In the community of Wiki, the concept is sometimes called WYSIWYM: "what you see is what you mean").

A trade-off related to screen resolution is the choice of what a page in TinLizzie WysiWiki should be modeled after. A typical article on Wiki is primarily in text, and is typically longer than can be viewed on one screen. The browser that is showing the article usually has scroll bars. If there are non-text contents (such as pictures) in the article, they are anchored to positions in the flow of text and move on the screen when the user scrolls the text. On the other hand, a page in TinLizzie WysiWiki is modeled as a presentation slide with no scroll bars around it. There is a text widget with scroll bars, but such a text widget is embedded as a sub-part of a page. In other words, the current abstraction of a document is a multimedia content with some embedded text objects, but not the other way around. (See figure 5.)

From our experience, even in an article with simulations, the primary information will still be in text. For future WysiWiki systems, we will explore different abstractions for documents.

Another aspect of screen resolution is that the line end wrapping and justification can be different on different computers; also the relative size of text and embedded contents may be different. Our solution for this problem is to replace the rendering engine of Tweak and Squeak with Cairo [18], which supports scalable graphics. We have an experimental version of such graphics system called Rome, and will continue further refinement.

5.1.2 Version Control

We currently use the SVN repositories for versioning documents. A problem with this approach is that ODF files are in binary so that we don't get the full benefit of using a version control system. In a future system, the internals of

ODF files (XML files) should be taken into account.

5.1.3 Synchronous Collaboration

The synchronous collaboration in TinLizzie WysiWiki uses the framework provided by Croquet. It works both in the LAN and WAN environment with low latency. At the same time, the framework is under development and still missing some pieces like peer discovery in wide-area networks, and failure recovery. The authors believe that the fundamental concepts of TeaTime, such as keeping identical computation on distributed computers, strong notion of time, and peer to peer network topology, have desirable characteristics. However, these still need refinement.

5.2 WikiPhone

WikiPhone shows an extraordinarily simple implementation of VoIP. However, it has limitations to overcome to be practical.

5.2.1 Network Topology

One may think that the server-based network topology is the biggest problem keeping WikiPhone from being practical. The server-based approach surely adds latency and contributes to the scalability problem. On the other hand, the HTTP based approach, which is the central idea of WikiPhone, provides very simple foundation for implementation. The user's perception is closer to the Web accessing compared to typical VoIP systems; WikiPhone and Web shares single point of entry for different services.

5.2.2 Audio Quality

For a telephony system, minimizing the latency is very important. While our approach has inherent overhead, there are areas where overall latency is improved.

The Squeak VM (running on Linux in particular) itself adds 300ms or more of latency because of the buffering. The VM primitives should be visited and the buffer size should be adjusted. In the current implementation, the overall latency can be as low as a few hundred milliseconds but can also be up to a few seconds.

Also, the arrival rate of data packets is not uniform on the Internet. In the current implementation, there is no attempt to smooth them. An interesting approach is to combine the data transmission with Croquet's message mechanism so that the packets are time-stamped, and the playback timing is adjusted on the client computers.

The choice of codec is also important. The GSM codec, which is currently used, is fine for simple voice, but the quality is not very satisfying.

Some of the problems here will be solved by Moore's Law. The sound quality with codecs, and some part of the latency problems can and will be solved. It will be interesting to watch these developments.

5.2.3 Understandability and Educational Value

Also, the simplicity has some educational value in learning about VoIP, and, more generally, about digital audio. A user can open the URL for the data packets, and see the actual binary data right there in the browser. A learner thus has a way to see that audio is something that can be represented as digital data, and sent around.

5.3 Future of Web Scripting

Today, all commercially viable Web browsers have JavaScript engines integrated. JavaScript is a fine language, but the way it is integrated with the browsers and the typical implementations of the engine aren't designed for writing programs with multimedia content.

Our group is working on a malleable dynamic language execution engine. One of the applications of this engine is to improve the performance of JavaScript. In fact, our aim is to make JavaScript fast enough to write all the graphics system in it. We envision that this, or similar systems, will lift some of the artificial limitations of today's web programming, and bring about better platforms for collaboration.

6 Conclusions

We have presented two systems which try to provide better collaboration over the Internet, in particular on the Web. TinLizzie WysiWiki enables multi-user collaboration using user-scriptable, media-rich graphical objects. In addition to providing Wiki-like asynchronous collaboration, it also allows synchronous, real-time, collaboration similar to a remote desktop. WikiPhone is a minimalistic VoIP system that lets users talk to each other. Both systems still require minimum support of a locally installed browser plugin, but we hope to lift this limitation in future systems.

The prototypes of these systems are working, and they have given us some insights about collaborative systems. We discussed these lessons and future directions.

Acknowledgement

We would like to express thanks for the valuable insights that Alan Kay, Kim Rose, Ted Kaehler, Bert Freudenberg, Michael Rueger, Kazuhiro Abe, David Smith, and other colleagues have given us.

References

[1] A. Kay, `http://en.wikipedia.org/wiki/Talk:Logo_%28programming_language%29`.

[2] D. C. Engelbart and W. K. English, "A research center for augmenting human intellect," in *In Proceedings of the AFIPS Fall Joint Computer Conference*. The Thompson Book Company, 1968, pp. 395–410.

[3] A. Kay, K. Rose, D. Ingalls, T. Kaehler, J. Maloney, and S. Wallace, "Etoys & SimStories," February 1997, ImagiLearning Internal Document.

[4] W. Cunningham, "The WikiWikiWeb," `http://c2.com/cgi/wiki`.

[5] I. döt Net, C. West, C. Dent, M. Liggett, R. King, D. Rolsky, and K. Liu, "Wikiwyg," `http://www.wikiwyg.net/`.

[6] A. Bryant, C. Putney, L. Andrews, and A. Kay, 2006, `http://logowiki.net`.

[7] C. Allen, "SynchroEdit," `http://www.synchroedit.com/`.

[8] M. Brauer, P. Durusau, G. Edwards, D. Faure, T. Magliery, B. Radius, and D. Vogelheim, "Open Document Format for Office Applications (OpenDocument) v1.0," `http://docs.oasis-open.org/office/v1.0`.

[9] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay, "Back to the Future – The Story of Squeak, A Practical Smalltalk Written in Itself," in *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 1997, pp. 318–326.

[10] A. Raab, "Tweak," `http://tweak.impara.de`.

[11] J. H. Maloney and R. B. Smith, "Directness and Liveness in the Morphic User Interface Construction Environment," in *ACM Symposium on User Interface and Software Technology (UIST)*, 1995, pp. 21–28.

[12] J. Maloney, *Squeak: Open Personal Computing and Multimedia*. Prentice Hall, 2002, ch. 2: An Introduction to Morphic: The Squeak User Interface Framework, pp. 39–68.

[13] Y. Ohshima, "Kedama: A GUI-based Interactive Massively Parallel Particle Programming System," in *Visual Languages and Human Centric Computing*, 2005, pp. 91–98.

[14] Y. Ohshima, K. Wakita, and M. Sassa, "A Report on Porting the Programming Environment Squeak to SHARP Zaurus and its Evaluation (in Japanese)," *Transaction of Information Processing Society of Japan: Programming*, vol. 41, no. SIG-9 (PRO 8), pp. 62–77, November 2000.

[15] D. A. Smith, A. C. Kay, A. Raab, and D. P. Reed, "Croquet - A Collaboration System Architecture." in *In proceedings of First Conference on Creating, Connecting and Collaborating through Computing (C5)*, 2003, pp. 2–9.

[16] D. P. Reed, "Naming and synchronization in a decentralized computer system," Ph.D. dissertation, Massachusetts Institute of Technology, 1978, (available as Technical Report: TR-205).

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, and E. Schooler, "SIP: Session Initiation Protocol," iETF RFC 3261.

[18] C. Worth and K. Packard, "Xr: Cross-device rendering for vector graphics," (presented at the 2003 Ottawa Linux Symposium).