# Educational Applications of Multi-Client Synchronization through Improved Web Graph Semantics

Michael Capps Brian Ladd David Stotts Lars Nyland
The University of North Carolina at Chapel Hill <capps, ladd, stotts, nyland>@cs.unc.edu

## Abstract

*The Multi-Head, Multi-Tail, Multi-Client (MMM) Browsing Project is a continuing effort to bring stronger graph semantics to the World Wide Web thereby increasing the Web's usefulness for collaboration and education. The first modified browser [1] provided facilities for links with multiple heads and tails, thereby giving the content author the ability to direct concurrent and synchronized browsing streams. The web author can direct the paths of his readers, and ensure that they visit pages in the correct context and order desired.*

*Our most recent effort [2] was the construction of a layer that filters content between the reader and the Web. This layer allows the ready composition of graph protocols so that Web content can be interpreted according to a variety of graph models. The model of the first project was expanded to be a full colored Petri net, thereby allowing synchronization of multiple browsing streams; the application of the MMM project to education now extends to a full collaborative classroom.*

## Introduction

The Multi-Head, Multi-Tail, Multi-Client (MMM) Browsing Project at UNC is a continuing effort to bring stronger graph semantics to the World Wide Web, thereby increasing its usefulness in a variety of collaborative and programming situations. The first cycle saw the creation of the MMM browser, a modified version of NCSA's Mosaic client which included special handlers for multi-headed multi-tailed hyperlinks. In the first MMM paper [1], we described how these multilinks offered the author both the ability to synchronize his reader's browsing paths, as well as to direct multiple concurrent paths. We explain below how this browser was fashioned, and the changes to HTML that were necessary to support the additional functions.

The second research effort was a generalization of the first MMM project. We realized that it might be possible to develop an interpretive layer in a browser that would read generalized link and node structures and from them devise a graph of a type specified either by author or user. In other words, the author could develop a series of interlinked pages, and the user might browse them as if they were nodes in a Petri net or nodes in a Parallel Finite Automaton. This Semantic Web Graph Layer, which we term the SWGL, implements the graph protocols in a concatenated stream of content filters. This gives us flexible extendibility, in that the next protocol often only requires a new filter to be added to an old protocol, as well as an interesting tool for graph protocol extension, by modifying the composition or order of known filter streams. Later in this paper we explore the architecture of this layer, as well as the further modifications to HTML that were necessary to support the generic protocol interface. Greater detail can be found in [2].

In addition to the graph layer, we concurrently explored a simpler, but more rapidly prototypable, server-side scripting design. Later in this paper we discuss the advantages of each design path and our conclusions.

The most recent directions of the MMM project sprang from a demonstration of the abilities of the SWGL. We originally chose the colored Petri net (CPN) protocol to display the filter-composition architecture of the SWGL, as well as to show the multi-client session manager that was necessary to synchronize browsing behavior of multiple readers. The value of CPNs in collaboration has already been investigated [3]; we now are investigating the application of our CPN collaboration tool to a standard classroom environment.

Our paper begins with a comparison of our system to other CSCW projects. We continue with an introduction to Petri nets, a concept used throughout the paper and our research. The main sections of the paper use that information to explain the three MMM projects. We conclude with what we consider the successes and failings of the MMM research, as well as a hint of possible directions in the project.

## Petri nets

Petri nets are a model of computation used to describe parallel processes, essentially bipartite graphs with place nodes and transition nodes connected by links. Links are constrained to connect either a place to a transition or a transition to a place. Places can contain tokens, which are markers representing the state of the computation. In a traditional monochromatic Petri net, when all of the places leading into a transition contain at least one token, the transition is active. Computation in the Petri net proceeds by firing an active transition, removing one token from each of the incoming places and adding one token to each of the outgoing places from the fired transition.

This model of computation can be applied to browsing a hypertext document [3]. In Web terms: a page is a place, a page is made visible in our browser when it has at least one token, and our specialized Multi-HTML constructs are transitions, complete with the list of links into and out of the transition. Our first work uses a slight simplification of this interpretation, parallel finite automata, that are like Petri nets except that they do not track any more than a single token in a place.

In our most recent work, we make use of an extension to monochromatic Petri nets, the colored Petri net (CPN) [7]. Here there are different types of tokens, known as colors, and transitions can have more complicated formulae for activation such as "A token in each of the two input places, but they must not be of the same color." The SWGL interprets each person in a session as using a different-colored token. Hence the author can specify the desired browsing semantics for a group by describing the transition activation rules.

## Multi-Head, Multi-Tail Mosaic

The first phase of research was prompted by two particular problems the current generation of Web protocols fails to handle: concurrent browsing streams and synchronization of browsing streams. Concurrent browsing streams are two or more paths of browsing, all of which are part of the same user's browsing session. Many current browsers permit the reader to follow a link in such a way that a new window is opened with the content at the tail end of the link; they do not, however, permit authors to specify this behavior directly in their documents. Synchronization of multiple browsing streams is necessary to ensure that the reader is able to advance so far in one stream without advancing another, or to advance only along with multiple other streams in lock step. Current Web browsers and protocols fail to address these needs.

Authors of computer-aided instructional material have, traditionally, had great control over the order of presentation of their material. The power of hypermedia, though, is that readers can follow train-of-thought explorations in linked information structures. As CAI applications migrate to the Web, authors find HTML restricts their ability to express orderings more complicated than one browsing thread. Our methods allow authors to integrate some control of presentation order with this traditional browsing freedom. The generalizations of the link we have created allows a mixing of both capabilities.

We encapsulated this extended Web graph model in a novel browsing client, Multi-head Multi-tail Mosaic (MMM). We also created MHTML, an extension of HTML with facilities for expressing multi-head/multi-tail links. The resulting graph semantics are equivalent in a formal context to those of a Parallel Finite Automaton (PFA), as mentioned above. PFA's can be used in similar manner to Petri nets for specifying hyperdocument structure, as explored in the Trellis system [3,4]. Automata have been used to good effect in computer-aided instruction and other fields where control over the traversal of a graph is important. It is useful to note that neither of these two models allows synchronization between the browsing streams of two separate processes. This behavior can be specified with a simple multi-user graph model such as colored Petri nets (CPNs); we explore that avenue in our more recent research.

## Implementation of prototype system

The MMM project's original goal was to test the usefulness of PFA semantics in the World Wide Web setting; the focus was on rapid development of a prototype. To ensure an appropriate sample population, it was necessary to make that prototype acceptable to current Web users. We chose to modify a popular and freely-available browser (NCSA Mosaic), as we were felt users would be more willing to experiment with a newly-downloaded browser than a similarly-acquired server. Less than two thousand lines of code pre-process MHTML into HTML which the standard Mosaic renderer presents to the user. Hooks in the code present the DISPLAY_TEXT field when a pointing device passes over an MHMT anchor, open and close appropriate windows when one is selected, and update the status of MHMT anchors when windows open or close.

## MHTML: extensions to HTML

Two considerations guided our extensions of HTML: ease of use for authors and logic of presentation for unmodified browsers. Using existing constructs as much as possible forwarded both goals. The tendency of browsers to ignore unrecognized tags permitted us to accomplish the second goal fairly easily.

A multi-headed/multi-tailed link is enclosed between a <MHMT> and </MHMT> pair of tags. Between them text, anchors, and our incoming reference (<IREF>) tag are all treated specially by our modified browser. An example link, which leads from a set A of three pages to a set B of two new pages, follows:

```
<MHMT text="Continue to Next Set"
<A HREF="B1.html">New Page 1</A> and
<A HREF="B2.html">New Page 2</A>
<IREF HREF="A2.html">
<IREF HREF="A3.html">
</MHMT>
```

*Figure 1*

This example has two outgoing links (as defined in the <A HREF> tags) and three incoming links (two explicitly defined with <IREF> tags and one from the current page).

So, with the MHMT construct the author has control of what the user sees, where the user is told that he is going (as opposed to the bare URL displayed by most browsers), the pages the user must have already visited (synchronization), and the pages the user will see afterwards (concurrent browsing). Though authoring graphs in this fashion is predictably difficult, we do have a tool for graph visualization and MHTML authoring.

## Multi-Head, Multi-Tail, Multi-Client communications in the web (MMM2)

As previously mentioned, richer graph models allow the author to "program" the browsing behavior they want readers to see by turning the hypertext into a hyperprogram with specific semantics.

Our previous work with Multi-head Multi-tail Mosaic extended the Web graph model to include the programming power and semantics of parallel finite automata (PFA). The second project extended the browsing streams which can be synchronized beyond those of a single user. Authors can now write documents which keep multiple users' browsing paths synchronized, an addition which has obvious utility in instructional as well as other tasks.

Rather than just increase the power of a single extended graph model, this second project also focused on inserting a semantic web graph layer (SWGL) in the browser. The layer provides an API for addition of new graph protocols, which allows a standard web browser to dynamically reinterpret links and nodes within a new model. The author may suggest, and the reader may choose from, multiple graph models such as PFAs, monochromatic Petri nets, CPNs, And/Or graphs, the Dexter Hypermedia model [5], and so forth.

### The Cobweb prototype

Concurrent with the development of the SWGL/proxy server implementation, a prototype was developed, not only to rapidly demonstrate the capabilities to potential users, but also as a means of exploring the capabilities of HTML browsers, http-servers, and long-lived connections between the two. The prototype multi-user system is based on three components: a graph embedded in a set of web pages; a state-server that manages the state of all users; and an agent to relay information between the state-server and the browser. For each graph, there is one set of pages, one state-server, and one agent per user.

The state-server represents a persistent object that maintains the state of the multiple users in the graph. It listens for requests to update the state of the graph, and relays all changes of state to all connected users when any state information changes. The state transition model it uses is extremely simple; it consists of a user request to enter a new state, and can specify whether other users follow along or not.

On behalf of each user, an agent process is run that communicates with the state-server, sending out modified web pages as state information is returned. The agent maintains a long-lived connection with the browser allowing another user's activity to be reflected by the delivery of new web pages. The state information received by the agent not only includes the pages to deliver, but additional information (the user's name, the number of users viewing this page, etc.) that can be used to "edit" the delivered pages. We used the C macro preprocessor for the inclusion/exclusion of contents in the web pages delivered.

The final component is the set of web-pages, whose links represent the edges of the graph. All links in these pages refer to an agent script with parameters to specify state transitions. The pages are standard HTML augmented with C preprocessor editing directives (#if/#endif). There is a well-defined set of symbols used with the C preprocessor, giving the graph author enough capabilities to control how users move together or individually through the graph.

Interaction with the prototype falls into two categories: either the user is actively advancing through the graph by selecting links, or is being advanced through the graph by other users. When a user clicks on a link, it breaks the connection with the current agent, starting a new connection with parameters that describe the transition. The new agent establishes a connection with the state-server, relays the parameters, and awaits a response that tells the agent what pages to
deliver and how to run the macro preprocessor. Once all of this takes place, the agent waits for updated state information from the state-server (caused by another user), and upon receipt, it performs the file editing again, delivering newly modified pages to the user.

### SWGL description

The Semantic Web Graph Layer (SWGL) is an interpretive layer which filters the interaction between the reader and the content of the Web. Currently, the model is implemented as a proxy server; all HTTP requests for HTML content go from the web browser through the proxy, so the SWGL can control at any time the content viewable by the client. Multiple page sets are implemented with the HTML frame construct. The SWGL uses a further-extended version of MHTML [2], which includes a method for specifying protocol types and transition-activation algebras.

As mentioned above, the SWGL filters the interaction between the reader and the Web content in the context of the current graph model. This means nodes and links can be dynamically interpreted as if they were part of a monochromatic Petri net, a parallel finite automata, or even part of the Web graph model. Setting the current graph model is usually handled by the document's author though these settings can be overridden with the SWGL user interface.

This extension of the MMM architecture yields great flexibility for graph theory research. Significant work has already been invested in analyzing graph semantics, and a user-definable, dynamically-switchable interpretive layer offers a significant tool for such research. The SWGL allows us to combine features of existing semantics with others to investigate additional possible protocols.

For this, we take advantage of the fact that the SWGL is implemented as a stream of filters which interpret extended HTML as graph content. Filters can be composed; the exact order and number of filters is the specification of a particular graph layer protocol. Protocol filter streams have been explored in other collaborative tools such as XPEL [6].

As an example of the power of filter composition, consider that the difference between PFAs and Petri nets is that PFAs do not keep track of multiple markings of a given place; additional tokens are simply discarded. Given the specification of a PFA graph layer protocol, the Petri net protocol required only the addition of a "token counting" filter to the filter stream implementing the PFA graph layer protocol. The creation of a multi-colored token handler, as well as a session management system, gave us the CPN protocol in a similar facile manner.

We also provide within the SWGL a generic session manager such that workgroup protocols, like CPNs, have the ability to communicate between users.

## Colored Petri nets in classroom collaboration

In this final section, we focus on a particular extended graph model, colored Petri nets, and explain how it can be used as a protocol for synchronized collaborative exercises, such as are useful in education. We then follow with a more detailed explanation of our implementation of the CPN protocol in the web through presenting a scenario involving a distributed classroom.

We will focus on the use of CPNs to describe the behavior of Computer Presented Reading Assignments (CPRAs). CPRAs are grouped into difficulty levels referred to be colors; for example, a student must pass all assignments at the Silver level before proceeding to the Gold level. Within a level, CPRAs have the following properties: they may be pursued in any order; a student starting an assignment must finish before starting another; students should be able to view the assignment while being tested upon it (tests are "open book"); the teacher should be informed when they complete an assignment (in this case to grade it); a student's progress should not interfere with or be visible to other students.

The following discussion refers to Figure 2, an excerpt of the Gold level from a network implementing CPRAs. Tokens in the network are individually typed for each of the students and there is a teacher-typed token referring to the instructor. We now trace a student's progress through the Gold level.

From the *Enter Gold* page the student can begin the Gold level net by pressing the highlighted *Entering Gold* transition. A student-typed token is placed in the *Gold Introduction* page and each of the invisible *Must Pass* assignment pages; in an attempt to keep this example tractable only two assignments are used. The *Gold Introduction* page is then presented to the student, though the other pages are not. Invisible pages (dotted circles on the diagram) are never presented, though their token marking is taken into account when calculating what transitions are active. On entry, both the *Go Gold 1* and *Go Gold 2* transitions are active since all places on input arcs are marked with the student's token. The presence or absence of other students does not change her view.

When the student chooses *Go Gold 1* the introduction page is replaced with the *Gold 1 Assignment* page. At this point it might be worthwhile to note one of the tools our system provides the instructor: as a privileged user of the session manager, the teacher can check where any or all of the students are at any time. Therefore the teacher has available information like the time each student is spending on a particular assignment as well as the level and number of that assignment.

When the student finishes with the *Gold 1 Assignment* she can progress to the *Gold 1 Test*; the assignment page is remarked as well, putting two windows on the student's screen so she can review the reading while answering the test questions as desired above. Both the assignment and test pages are unmarked when the student progresses through the *Grade Gold 1* transition.

*Grade Gold 1* also generates a teacher-typed token in the *Award Grade Gold 1* page. This brings up a new window on the teacher's workstation, presenting him with the student's answers to the test and the choice of passing or failing the student. It is possible to automate the grading so a process running as a proxy for the teacher determines pass-fail, but for this example of collaboration it was desirable to keep the human teacher in the loop.

If the teacher passes the student the invisible page *Passed Gold 1* is marked with the student's type of token; if the student fails, the invisible *Must Pass Gold 1* page is marked (as it was on entry to the Gold level). In either case the student progresses to the *Gold Introduction* page where she can choose from the assignments she still needs to pass.

The *Finish Gold* transition is active when the student has passed both assignments at the Gold level and is at the *Gold Introduction* page. The *Passed Gold* assignment pages are not truly necessary in the CPN model, since we already have pages to represent

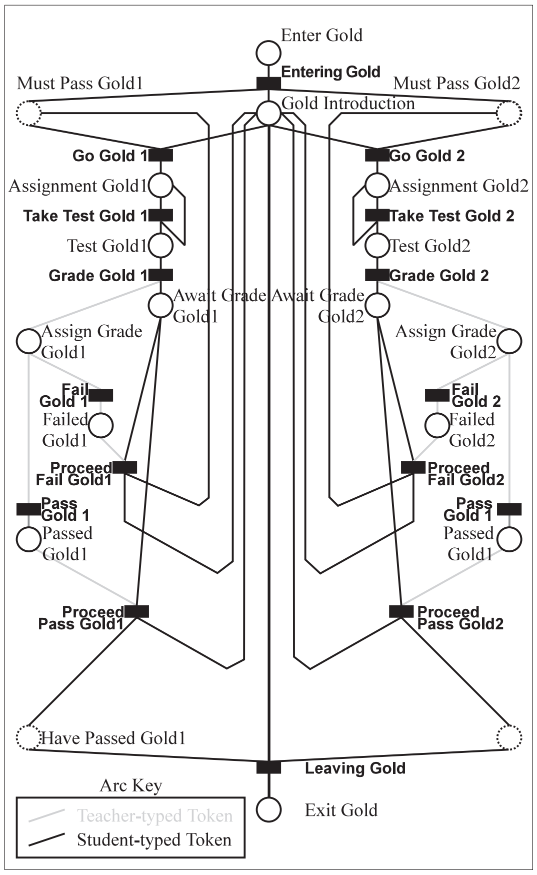

*Figure 2*

each page not being completed. CPNs have the ability to place algebraic formulae on each transition to state an activation condition based on the exact type of tokens available. Our example does not include link algebras because of the added complexity, but note that at least in this case, links that specified "not-student-token" in the *Must Pass Gold* assignment pages would indeed simplify the net.

While it adds a level of complexity not needed for this example we have also developed a network representing an adaptive learning environment where the student can fail, low pass, pass, or high pass the exam. The contents of assignment 2 can then be changed based on the grade on assignment 1 (or instead of the same assignment 1 the student can get another assignment at the appropriate level so answers cannot be memorized). It is also possible to have a high pass mark more than one assignment or to tailor assignments to the types of learning the student has shown the need for.

## Scenario - behind the scenes

The anticipated hardware and software implementation of the CPRA scenario is based on that in ARPA educational research: a single high-performance workstation for the instructor, connected to individual student computers by a local network. We foresee the Session Manager running as a server process on the instructor's workstation with a copy of the SWGL and a browser running on each student machine.

The Session Manger is started and stopped by the instructor, who is then super-user for the Session Manager. This means the teacher can add and delete students, review where students are in the network, and shut the system down, taking a snapshot of the marking of the network for restoration at restart.

Student requests for pages from their browser are passed to the SWGL proxy on their computer where the legality of the request is checked in terms of the current graph protocol. The CPRA example assumes that the protocol is CPN, though the SWGL was designed to support multiple graph protocols.

As an example of SWGL operation, consider what happens when the student passes through the *Grade Gold 1* transition (again see Figure 2). The marking in the network changes, placing a student-type token in the *Await Grading Gold 1* page and a teacher-type token in the *Grading Gold 1* page. The new teacher-type token is communicated to the SWGL on the instructor's machine; a new window will appear with the student's answers to the test. Since they only have one input, both the *Pass Gold 1* and the *Fail Gold 1* transitions are active. The teacher selects one of them, for example *Pass Gold 1*.

Now a teacher-type token is placed in the *Teacher Pass* page. This change must be communicated to the student's SWGL because the *Proceed - Pass Gold 1* transition must be made active. In the current implementation the SWGL keeps a long-lived connection to the browser using the Netscape standard for handling multi-part MIME messages. Through this connection, the SWGL can replace the contents (in which the transition would not be an available link) with the new contents (with the transition properly highlighted).

Hopefully it is clear from the current and preceding sections that the CPN model implemented using the SWGL can handle concurrency (assignment and test shown simultaneously), single-user and multi-user synchronization, having one user's browsing pattern affects another's, or the content of active pages. These capabilities are distributed among several filters which are composed to provide the desired semantics.

## Future work

We look forward to improving the current research in two ways: the number of different protocols supported, and the number of users who have tried our system.

Many interesting filters which would have useful applications in the classroom environment have been considered for the SWGL. For instance, it would be possible to provide 'Net-sitter content-filtering based on site addresses or on-the-fly content analysis; we envision an instructor choosing to lock-out most non-local sites during work periods, then releasing the restrictions during free time. We also are researching the usefulness of timed-links, which have a window for activation and can even fire themselves after a certain interval.

Another filter of interest is one that keeps track of browsing path and timings on using the Web. Such a filter will be a necessary part of user studies with MMM in a real classroom. Another prelude to user studies is tuning the SWGL implementation for performance. We envision optimization strategies for browser-to-SWGL-to-WWW communications, as well as examining how the SWGL can best do caching for the attached browser.

We hope to have hardware in place soon to permit the testing of MMM as a classroom support system. In addition to trying to find an elementary education setting, we are investigating candidate college settings. One stumbling block is the development of appropriate curricula; we are actively looking for ways to incorporate existing resources.

## Conclusions

Providing a semantic layer between the browser and the Web permits MMM to support complex, cooperative tasks on the Web. Concurrent browsing streams can be split and synchronized between multiple users of the system under author control. Providing this power to authors is useful in many contexts including education (as described above), business process workflows, and other process control situations.

Formalizing the interaction between various browsing streams permits multi-user browsing; it also permits the application of prior research in parallel and distributed computing (and the reuse of found solutions) to the enhanced Web. As it becomes apparent that the simplicity of the Web must give way to greater functionality, MMM provides a flexible, extensible, solution running atop current standards.

Current information about the MMM project is available at http://www.cs.unc.edu/~stotts/MMM/.

## References

1. Ladd, B., M. Capps, D. Stotts, R. Furuta. "Multi-Head Multi-Tail Mosaic: Adding Parrallel Finite Automata Semantics to the Web." *Proceedings of the 4th World Wide Web Conference*, Boston, MA, Decemeber 1995, pp. 443-440.
2. Capps, M., B. Ladd, D. Stotts. "Enchanced Graph Models in the Web: Multi-client, Multi-head, Multi-tail Browsing." to appear in *Proceedings of the 5th Annual World Wide Web Conference*, Paris, France, May 1996.
3. P.D. Stotts and R. Furuta. "Petri Net Based Hypertext: Document Structure with Browsing Semantics." *ACM Trans. on Information Systems,* vol. 7, no. 1, January 1989, pp. 3-29.
4. R. Furuta and P. D. Stotts, "Interpreted Collaboration Protocols and their use in Groupware Prototyping," *Proc. of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*, Research Triangle Park, NC, October 1994, pp. 121-131.
5. F.G. Halasz, and M. Schwartz. "The Dexter hypertext reference model," *Communications of the ACM* 37,2 (Feb. 1994), pp. 30-39.
6. J. Menges. "The X Engine Library: A C++ Library for Constructing X Pseudo-servers," *Proc. of the 7th Annual X Technical Conference*, Boston, MA, January 1993.
7. J.L. Peterson. *Petri Net Theory and the Modeling Of Systems*, Prentice-Hall, Englewood Cliffs, N.J.1981.
8. H. Shen and P. Dewan. "Access Control for Collaborative Environments." *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, November 1992, pp. 51-58.
9. Stefik et al. "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings." *Communications of the ACM*, January 1987, Vol. 30, No. 1, pp. 32-47.
10. S. Sarin and I. Greif. "Computer-Based Real-Time Conferencing Systems." *IEEE Computer Magazine*, October 1985, Vol. 18, No. 10, pp. 33-49.
11. C. Ellis, S. Gibbs, and G. Rein. "Groupware: Some Issues and Experiences." *Communications of the ACM*, January 1991, Vol. 34, No. 1, pp. 38-58.
12. D. Tatar, G. Foster, and D. Bobrow. "Design for Conversation: Lessons from Cognoter." *International Journal of Man-Machine Studies*, Vol. 34, No. 2, pp. 596-608.
13. P. Dewan and R. Choudhary. "A High-Level and Flexible Framework for Implementing Multi-User User-Interfaces." *ACM Transations on Information Systems.*, October 1992, Vol. 10, No. 4, pp. 345-380.
14. Ralph Hill et al. "The *Rendezvous* Architecture and Language for Constructing Multiuser Applications." *ACM Transactions of Computer Human Interaction*, June 1994, Vol. 1, No. 2, pp. 81-125.