Appeared in the Proceedings of the
IEEE International Conference on Communications, ICC'2001, Finland, June 2001

Prefetch Scheduling for Composite Hypermedia

Javed I. Khan and Qingping Tao

Internetworking and Media Communications Research Laboratories
Department of Math & Computer Science, Kent State University, 233 MSB, Kent, OH 44242

***Abstract*- Composite hypermedia documents are fast becoming ubiquitous in web. In this paper we explore a novel segment-based background prefetch technique that utilizes the rendering properties of various individual media elements and their interdependencies. The technique can play a role in accelerating surfing speed in composite hypermedia.**

I. INTRODUCTION

Most web pages are no longer simple HTML with few embedded images. Information catered by modern web servers is increasingly becoming multimedia enriched, active and dynamic. With the increased sophistication of the catered document responsiveness of web systems is experiencing slowdown. Most modern web pages now not only contain a simple parent HTML with few embedded images but also contains embedded entities such as banners, JAVA applets, flash presentations, stock ticks, etc. with varying rendering constraints. A snapshot of a typical *composite hypermedia* page from ABC News© website is given in Fig-1. A summary of the 48 individual elements that makes this rendering is given in Tables -1 and 2.

A number of recent researches has anticipated that prefetching can significantly speedup web response just like it has accelerated hardware systems-- although none seems to address the complex case of composite hypermedia. In one of the pioneering studies, Kroeger et. al. demonstrated that with ample knowledge of future reference a combined caching and prefetching can reduce access latency as much as 60% [KrLM97,WaCR96, JaCa98]. Palpanas and Mendelzon [PaMe99] demonstrated that a k-order Markovian prediction engine could improve response time by a factor of up to 2. These methods used variants of partial matching of context for prediction. Pitkow and Pirolli [PiPi99] compared various methods to predict surfer's path from log traces such as session time, frequency of clicks, Levenshtein Distance analyses, etc. These studies focused on the problem of link transition probability estimation. In most of these works, however paths were ranked simply in-order of the estimated probabilities and once a path is selected the suggestions are to prefetch the entire documents. Such prefetch has been found to cause excessive wasted prefetch and can adversely affect the overall network bandwidth. Recently, Cohen and Kaplan [CoKa00] suggested just doing pre-staging- such as pre-establishing TCP connection to avoid the waste. RealPlayer (release 7 onward, 1999) seems to have implemented a very similar TCP/RTSP-session pre-setup. We have also recently contributed several new ideas. We note, in the web there is no usual limit on links. The degree of branching is very high (in contrast in hardware decision points mostly bifurcate the control flow tree). This is of particular concern in the Internet—which does not own an exclusive bus. In [Khan00, Khan99] we demonstrated that instead of simply ranking candidate hyperlinks in order of transition probabilities-- a ranking order that also considers the loading time can yield much better performance with respect to larger prediction errors. In [KhTa01] we also demonstrated that instead of all or none-- one can only preload an estimated lead segment. The remaining can be loaded in background only when they are requested. The technique can improve responsiveness with simultaneous reduction of wasted prefetch. This paper now extends the results for the case of composite hypermedia. A composite hypermedia offers a number of interesting challenges to web systems. The embedded elements are generally related with intricate rendering constraints and interdependence. Some early works such as AMSTERDAM hypermedia model [HaBR94] has dealt with the representational issues of various forms of hypermedia interdependencies. Indiscriminate and unplanned hypertext like file dump is bound to slow down the rendering just because of their sheer volume and associated delay in active rendering. We provide a technique, which looks into the rendering properties of the individual media elements of the composite hypermedia and provides an efficient scheduling of individuals media segments for partial prefetch based delay minimized rendering.

The following section first briefly presents the model and background material. Then section III explains the scheduling scheme. Finally section IV present the performance result of the proposed mechanics under various media scenarios obtained via statistical simulation.

II. ANALYSIS

A. Hyperspace & Transport Model

In this paper we borrow the initial model from [KhTa01] and extend it. Likewise we model the hyperspace in which a reader moves as a graph called *roaming-sphere* $G(V_G,E_G)$. Each node here is a *composite hypermedia* document. The reader moves through a sequence of nodes in this hyperspace called *anchor sequence*. Links has a transition probability p(i,j). The available bandwidth is correspondingly separated into two sub-channels; *feed channel* for loading the streaming segment of the current anchor, the *lead channel* to proactively load the lead segments. Each node thus has two parts-- the *lead segment* and the *stream segment*. We assume that D_{total} is the size of an elementary resource, D_{lead} is the bytes in lead

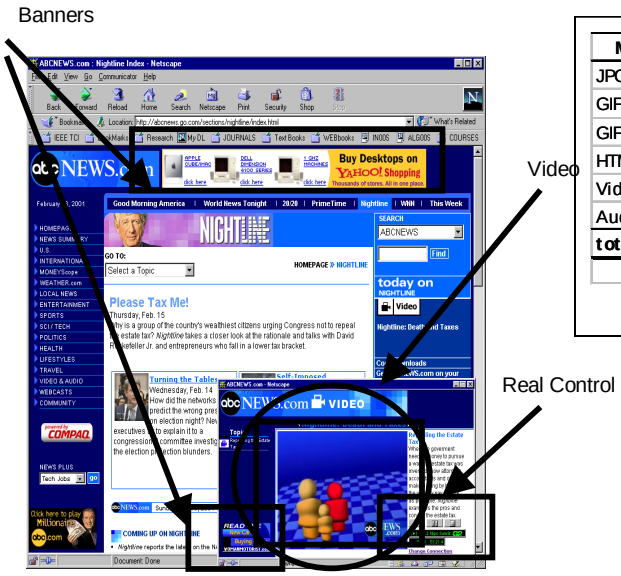

Fig-1 Example Composite Media from ABC News.
The rendering has 48 individual components.

Media Type	Files	KB
JPG images	4	19
GIF graphics	38	26
GIF animations	2	17
HTML	2	69
Video Stream	1	
Audio stream	1	
total	48	131

Table-1

	Comp.	Audio	Video
Codec:		16Kbps	RealVideo
		Voice	G2
minimum (Kbps)	115.8	~8	~83
maximum(Kbps)	256	~67	~219
Encoded/Clip BW	163	32	131
Average (Kbps):	181.4	35.4	146.2
Duration(min:sec)		3:21	3:21

Table-2

Table 1 and 2 Static and Dynamic Properties of the Media

segment and D_{feed} is the bytes to be streamed. We use β to denote the ratio of total bandwidth to that allocated to the lead sub-channel.

B. Composite Document Model

To model a composite media we introduce a set of functions called *rendering rate profiles.* These profiles characterize the data rate function along the presentation time-line of the composite document. A *profile* is intrinsic to the document and is not dependent on the communication network.

Given a composite document N with embedded media elements n^i, we denote individual element's profile as the function $f^i(t)$, where superscript i is the element index. We also introduce the following three common profiles -- (a) *constant rate* (CBR) media, (b) *impulse media*, and (3) *impulse series* media.

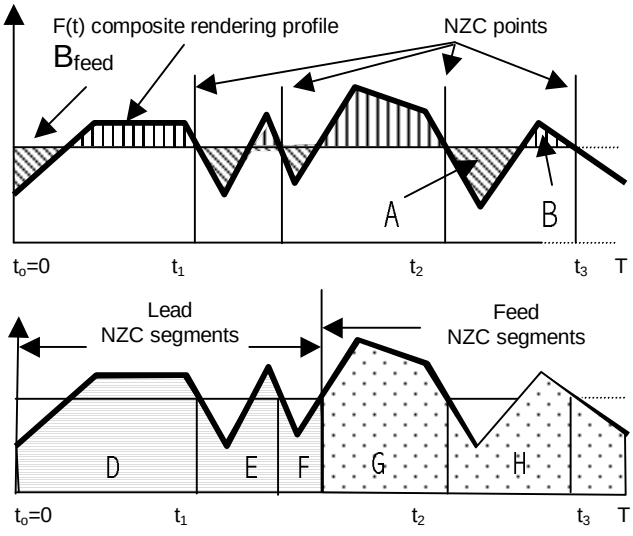

Fig-2 (a) at the top shows how the composite rendering profile can be rate for determining *critical lead mass*. In this example, the determining point is the middle one since mass A is larger than mass B. Fig-2(b) at bottom shows the lead (D,E,F) and feed NZC segments (G,H,I).

Most audio and video plays are generally constant rate media. Parent HTML page, JAVA applets, or embedded foreground/background images generally contribute to the lead segment and is considered impulse media. A dynamic text or banner image requires cycles of bursts for presentation. Similar is the case with any large document (PDF, ps etc.), which are viewed page by page. Additional examples of this class of documents are stock ticks and charts, pushed banners, GIF animations as well as flash and slide shows. We use equation set-1 to model the profiles of these entities.

$$\begin{aligned} f_c(t) &= R^i{}_c \\ f_I(t) &= H^i_I \\ f_S(t) &= \sum_{n=0} h^i_S \delta(t - nt^i_0) \end{aligned} \qquad \ldots(1)$$

The document's *composite rate profile* is the sum of individual functions characterized by the coefficients R, H and h.

Given the above analytical model with *roaming sphere* $G(V_G,E_G)$, the associated *transition probabilities p(i,j)*, and node *profiles* f(t) in this paper we show the optimum prefetch sequence for the media elements of the candidate nodes that will generate minimum expected response lag.

C. Critical Composite Lead Mass

The first question we address is given the rendering profile of a document how much of the document really needs to be prefetched.

We call the amount of data, given a feed bandwidth, that have to be preloaded to avoid rendering delay the *critical lead mass.* The *critical lead mass* for a composite media can be determined by piece-wise integration of the combined rate function performed over a set of intervals defined as *negative zero crossing* (NZC) points where:

$$\sum f(t_i) - B_{feed} = 0 \text{ , and } \frac{d}{dt}\sum f(t_i) \le 0 \cdot \qquad \ldots(2a)$$

Fig-2(a) explains the segments where t_1, t_2 and t_3 are three such points dividing the presentation time T into four segments. The size of the required minimum lead segment is then given by the maximum of the piecewise integrals evaluated in 0-t_i:

$$D_{lead} \geq \max_{j=1}^{N} \left(\int_{\varepsilon}^{t_j+\varepsilon} \sum_i f^i(t)dt - B_{feed} \cdot t_j \right) \quad ...(2b)$$

For zero delay presentation, the system must preload equal or more. Note, if a segment sum is negative in j^{th} interval, than it can reduce the piecewise lead segment of $(j+1)^{th}$ interval, however, the reverse is not true. The quantity ε is a small positive interval for ensuring inclusion of impulses in preceding intervals. Consequently, we look for maximum positive growth in the rate difference function (difference between the *composite rate profile function* and the *feed channel bandwidth)*, however, this can be evaluated only by integrating at NZC points. For example, in the last segment of Fig-2(a), since the negative area A is larger than the positive area B, the integral from 0 to t_2, instead of from 0 to t_3 is the determining interval for its critical lead mass. Note equation-2 is a general solution including VBR rate profiles.

However, easier computation is possible if the composite media is modeled by the profiles given by the equation-1, Accordingly, a composite document with combined continuous media rate R_c, an impulse media size H_I, and a impulse series media rate h_S per time unit t, for a presentation span of T sec, will require a critical lead mass of size:

$$D_{lead} = H + (R_c - B_{freed}) \cdot (T - t) + h\left\lceil \frac{T}{t} \right\rceil + \max\left[t \cdot (R_c - B_{feed}), 0\right] \quad ...(3)$$

The time to prefetch is given by:

$$T_{lead} = \frac{\sum_i D^i{}_{lead}}{B_{preload}} \quad ...(4)$$

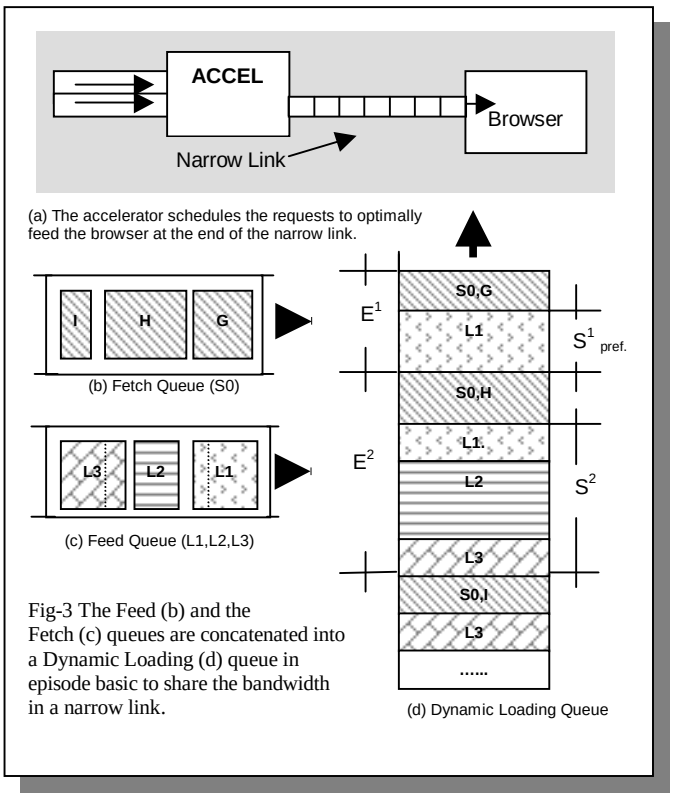

Fig-3 The Feed (b) and the Fetch (c) queues are concatenated into a Dynamic Loading (d) queue in episode basic to share the bandwidth in a narrow link.

D. Prefetch Node Ranking

Let $U=(a_1,a_2,a_3....a_l)$, where $u_i \in G$, is the *anchor sequence*. Let's Γ is the *loading sequence* in which the nodes are loaded in the prefetch cache (Clearly, $U \subseteq \Gamma \subseteq$ {nodes in G}). Let p_i is the estimated probability that a surfer traverses a node n_i in roaming sphere G, and $T_{L,i}$ is the time the node a_i is fetched and $T_{P,i}$ is the time spent by the surfer in that node. Thus, we define an overall penalty function-- the expected cumulative *read-time lag*:

$$T(\Gamma \mid U) = \sum_i^U p_i \max\left\{[T_{L,i} - (T_{L,i-1} + T_{P,i-1})], 0\right\} \quad ...(5)$$

The next objective is to find the loading sequence Γ that will minimize the expected penalty E{T(Γ|U)}. It is important to note that this function optimizes with respect to all probable transitions of U, weighted by their transition probability. Given the above optimization criterion, it can be shown that:

Theorem-1 (Branch Decision)*: Let $A=n_c$ is the current anchor point with direct transition paths to a set of candidate nodes n_1, n_2, n_3,.. n_n, such that T_i is the estimated loading times of node n_i, and $Pr[a_{n+1}=n_i \mid a_n=A]$ is the conditional link transition probability, then the average delay is minimum if the links are prefetched in-order of the highest priority Q_i, where:*

$$Q_i = \log \frac{\Pr[a_{n+1} = n_i \mid a_n = A]}{T_i} \quad ...(6)$$

The proof of this result is given in [Khan99]. For composite media, it states that at any branch point immediately linked nodes should be prefetched in order of the ratio of conditional transition probability and the estimated load times of their lead segments.

III. SEGMENT SCHEDULING

Finally, we show the schedule of data segments and allocation of individual streams within D_{lead} and D_{feed}. We want to avoid a situation where excessive preload of one media can potentially result in sub-critical pre-load for the other.

Given the NZC points of the combined profile, we divide the individual media entities into byte segments $B^i_n(t_j:t_{j+1})$, where i is the media index, and $t_j:t_{j+1}$ is the NGC intervals. We then define a *composite group* as $G_n(j)=\{B^i(t_j:t_{j+1})|$ for all i}. A group contains bytes from all entities those belong to

the same jth NGC segment of the composite document node N_n. Fig-2(b) shows the NZC segments those are within lead segment and the feed segment. The lead mass ends within the 3rd NZC segment. Let the size of these segments are given by $S^j{}_n = size(G_n(j))$. The following rule then applies:

Rule 1: *To ensure critical pre-load all bytes in $G_n(j)$ should be loaded before the bytes in $G_n(j+1)$.*

However, for lead prefetch, there is no requirement of ordering the bytes within a composite group $G_n(j)$, or between the groups from two nodes $G_n(j)$, and $G_m(j)$, when $n \neq m$.

Before showing the schedule, we first specify macro request *REQ(URL, segment)*, which is a request for an composite URL segment., and a collection of several micro requests *mREQ (URL/media element, segment)*[1].

We first queued the requests for each element into two virtual queues. The *prefetch queue* $Q_{prefetch}$ contains the fetch requests for lead segments of all the candidate prefetch nodes. The lead segments are ordered in terms of their Q priorities. Within each lead segment, the requests are further divided into *composite groups* in order of their NZC segments. The composite groups are further divided into element groups in order of their element order, which can be domain specific. The *feed queue* Q_{feed} contains fetch segments for the currently rendering node divided into *composite groups* in order of their NZC segments. Let this has q NZC segments. These two queues are then concatenated into a *dynamic loading queue* Q_{DL} in q separate episodes each of duration:

$$T^j = \frac{1}{(1-\beta)\cdot B_{channel}} \cdot S^j_{stream} \quad ...(7a)$$

In each episode, first a full NZC group is picked from the top of the feed queue. This is followed by prefetch bytes from $Q_{prefetch}$ in the amount of:

$$S^j{}_{prefetch} = \frac{\beta}{1-\beta} \cdot S^j_{stream} \quad ...(7b)$$

When any one of the queues depletes, the remainder of the other is directly appended at the end. The exact delivery time of the prefetch segments - as long as it is within the correct NZC segment, does not matter. Since, these are not in need of immediate rendering. We place the feed requests at the top of queue within each NGC segment. Thus, in effect stream data enjoys a little prefetch.

Once, a new anchor node k is traversed, all $G_n(j)$ $j \neq k$ is dropped from Q_{DL}. A New $Q_{prefetch}$ and Q_{feed} are formed and new concatenated segments are appended at the end of Q_{DL}. Consequently, if there is any remaining lead segment of N_k not yet fetched, it receives the 100% of the bandwidth.

The process is illustrated in Fig-3. We can consider a browser is acting at the end of a narrow link. An network embedded *accelerator appliance* unit is at the other end of this narrow pipe. The accelerator is scheduling the fetch operations for the given bandwidth B_{total} of the narrow link-end. Consider the analysis of current anchor node N_0 shown in Fig-2(b). The critical lead mass entails that NZC segments D, E, and part of the third segment F have to be prefetched. The reminder G, and subsequent NZC segments H, I, J, … becomes the feed segment, and is loaded when the user arrives in this node. Let's L1, L2 and L3 are the lead segments of the Q-ordered candidate links. Fig-3(a), 3(b) and 3(c) then shows the content of these three queues.

IV. PERFORMANCE RESULTS

We preferred to characterize the performance with statistically generated data set rather than with server trace. The controllability of parameters is inadequate in trace. Also the web-page composition seems to be changing rapidly. Thus, the results may not be extensible. Consequently, a broad range of hyperspace data with distinct and varying statistical properties were generated, and observations were made that how the proposed method would perform in each of these conditions.

The objective of the first experiment is to observe the impact of the two key proposed ideas—the Q-ordering of the links and the partial prefetching. We were also curious to see how the relative rendering speed of various media types will impact the performance.

First we generated a random set of nodes (composite documents) each with a parent HTML impulse document and a set of embedded CBR media. HTML documents were given fixed sizes (H^i_S). The sizes for the CBR media and the transition probabilities for the links were then generated using normal distributions. We limited the maximum links per node to 21. The underlying algorithm further pruned links with (below ε) low transition frequency. For a given link bandwidth we then varied the *profile parameter* h^i_I as a control variable. Also, since our focus was to track the

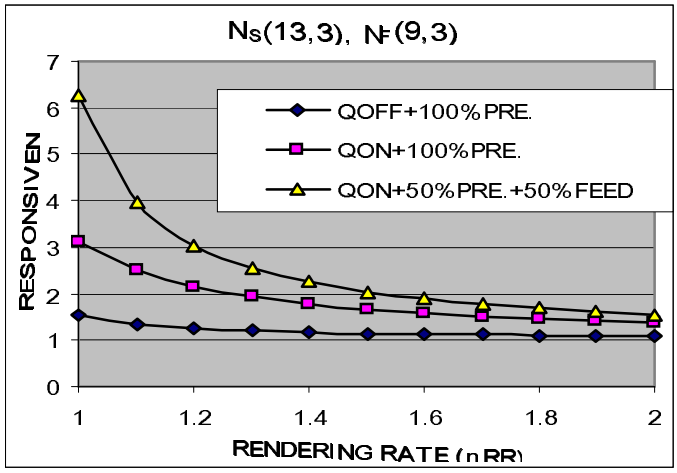

Fig 4

[1] HTTP 1.1 [FGMF97] conditional range GET mechanism (If-Range header, Range and Content-range) can be used for implementing the micro requests.

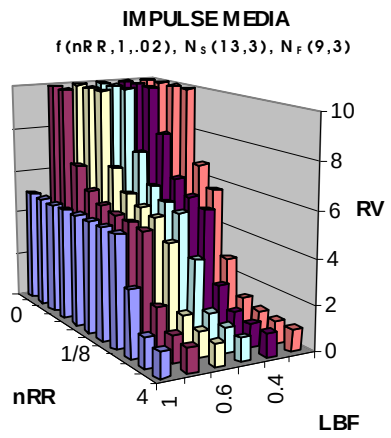

Fig-5(a)

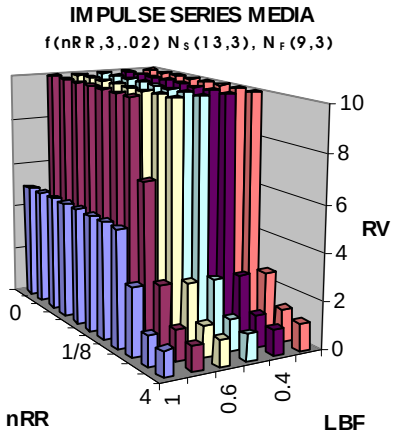

Fig-5(b)

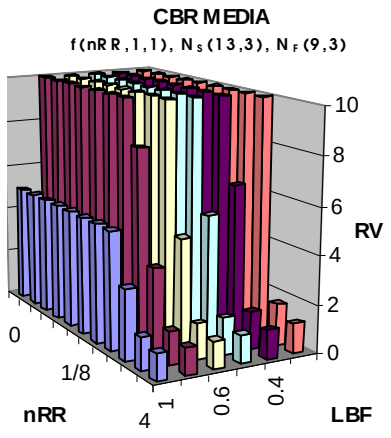

Fig-5(c)

performance only due to prefetching, we disallowed caching. With each move to a new anchor, all nodes became prefetchable again. We then let 21 sets of users walk through all probable chains of anchor points for the given schedule, not just one user group walking through the most probable one. We then observed the expected *responsiveness* (ratio of cumulative lag time experienced with active prefetching to that without any prefetch) for all groups.

Fig-4 plots the *responsiveness* (RV) (y-axis) for various *normalized rendering rates* (ratio of *network bandwidth* to the *rendering rate nRR* of the media) (x-axis). The top curve (C3) shows the performance with the proposed lead-only prefetch scheme. (QON+50%prefetch+50%feed.) For comparison, the bottom curve (Q-OFF+100%prefetch) shows the base case—with simple frequency based ordered prefetching. The dramatic speedup in system's responsiveness is apparent. How much was the contribution of ordering? The middle curve shows (Q-ON+100%prefetch) the performance improvement only with Q based link ordering. In the base scheme the maximum improvement in system's responsiveness we observed (at nRR=.7) is about 1.5 times. It improved to 3.6 times with Q-based link ordering and to +17 times with active lead only prefetching. Faster rendering media is always more challenging. In the other extreme for fast rendering media (nRR=1.7) the base scheme provided almost no speedup (1.12), while the Q-ordered partial prefetch was still able to improve the system's responsiveness by 1.47 times.

The next experiment we performed was to test how the proposed scheme fares against various media types? Besides, the variation in profiles, average rendering rate is also critical. Generally the text media is read only at a fractional rate of the line bandwidth, while video renders' much faster[2].

[2] Human ASCII text reading speed is about .1 to .5 kbps, which translates into nRR= .01~ .002 on a 56Kbps line. On the other extreme, typical video streams are generally served at close to line speed matched encoding with nRR ≈1.0. Prefetch might enable playing of 2-4 times larger stream. 4Mbps to .1 video can appear as nRR=16 to .4 to a 250Kbps DSL line.

For this experiment we correspondingly varied the normalized rendering rate over a much larger range from .04 to all the way 4.0 (log plotted in x-axis). We were also interested to see the impact of various lead bandwidth allocation. Y-axis plots *lead bandwidth factor* (LBF). To model impulse media we assumed that the entire rendering data have to be prefetched within a small fraction of its reading time (duty k=.02). For impulse series media we used duty k=.02 and number of cycles n=3. The 3-D plots of Fig-5(a), 5(b) and 5(c) respectively show the performance for the cases of (i) impulse, (ii) impulse series and (iii) CBR media for various nRR and LBF. As it can be observed in Fig-5(a) that slow rendering media can be accelerated about 5 times with l00% lead bandwidth (the leftmost series with LBF=1.0). In contrast, with just 30% channels assigned to feed sub-channel (LBF =0.7) the situation improves quite drastically resulting in almost zero lag (RV=+10 zone) surfing. Fig-7(b) and (c) show that the region of zero delay surfing further extends to even nRR ≈ 0.25 if the media types are impulse series or CBR.

This dramatic improvement shown by the proposed scheme is not unexpected. The reason can be tracked by observing the background line loads. Fig-6 plots the quantity *load factor*-- the ratio of the actual bytes those were

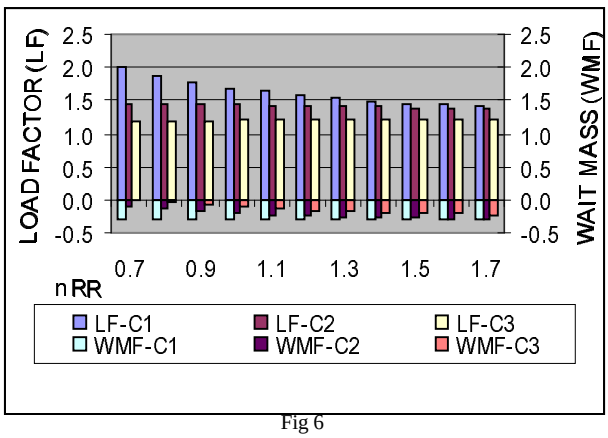

Fig 6

prefetched to the bytes actually read (size of *anchor sequence*) for the three cases shown in Fig-4. As can be noted that streaming enabled prefetching reduced the background line load to almost near one (10% excess) in large part of the graph. In contrast, the base scheme prefetched almost twice the data than read. Any lag is caused by the un-prefetched part of the lead mass called *wait mass*. The objective of the prefetch algorithm is to avoid it's buildup. The 2nd y-axis of Fig-6 (bottom bars), shows the *wait mass* (WMF as a factor of total bytes to fetch) which almost disappeared in our case.

To summarize, the above results show how careful scheduling of the data-bytes per node, in effect allowed information from more candidate nodes to be pre-fetched, given the same bandwidth and local storage. Results suggest a little detail knowledge about the rendering profile and arrangement of media elements inside a composite document can be prudently utilized for a considerable reduction in hard prefetch. Any prefetch is a 'gamble'. The larger the part of the documents that can be left for 'deterministic' streaming, the less is the waste.

V. CONCLUSIONS & CURRENT WORK

The paper's principle focus is on the **composite hypermedia scheduling**. We have presented the technique and optimization algorithms used for stream segmentation backed by analytical model and rigorous statistical simulation. However, there are many open issues in prefetching. Below are some references to other pertinent works. As indicated, the estimation of link transition probability, whether it is from conventional access log [PiPi99], or from explicit message exchange [Duch99] is essential. Also important will be efficient mechanics for partial document transfer. References [GrRB99, GrVe99] have studied requirements for the RTSP and XML extensions for fragment based communication. Streaming issues for continuous media can be found in [JuLC00, SeRT99]. In this paper we have used the rate profiles for representing rendering constraints, and demonstrated the potential pay-off. Further research should be conducted for media elements with more complex forms of temporal inter-dependencies of composite hypermedia. Some interesting research on representation can be found in the AMSTERDAM Hypermedia model in [HaBR94]. Embedded active scripts also offers scope of pre-staging.

As a part of our ongoing research, we are currently investigating an active net deployable accelerator module, which can be dynamically launched as an active appliance inside network.

VI. ACKNOWLEDGMENTS

The work is currently being funded by DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.

VII. REFERENCES

[CoKa00] E. Cohen and H. Kaplan. Prefetching the Means for Document Transfer: A New Approach for Reducing Web Latency. Procs. of the IEEE INFOCOM 2000, Tel-Aviv, Israel, March 2000.

[CrBa98] M. Crovella, P. Barford, The Network Effects of prefetching, Proc. Of IEEE INFOCOM 1998, San Francisco, USA, 1998.

[Duch99] D. Duchamp. Prefetching Hyperlinks. Proceedings of the USENIX Symposium on Internet Technologies and Systems, Colorado, USA, October 1999. [Http://www.usenix.org/events/usits99].

[FaCJ99] Li Fan, Pei Cao, and Quinn Jacobson. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. Procs. of the ACM SIGMETRICS' 99, Atlanta, Georgia, May 1999.

[FGMF97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk & T. Berners-Lee, Hypertext Transfer Protocol HTTP/1.1, RFC 2068, January 1997.

[GrRB99] S. Gruber, J. Rexford, and A. Basso, Design considerations for an RTSP-based prefix caching proxy service for multimedia streams, Tech. Rep. 990907-01, AT&T Labs - Research, September 1999.

[GrVe99] Grosso, Paul, Daniel Veillard, XML Fragment Interchange, W3C Working Draft 1999 June 30, [Retrieved from: http://www.w3.org/1999/06/WD-xml-fragment-19990630.html]

[JaCa98] Q. Jacobson, Pei Cao, Potential and Limits of Web Prefetching Between Low-Bandwidth Clients and Proxies, 3rd International WWW Caching Workshop, Manchester, England, June 15-17 1998.

[JuLC00] J. Jung, D. Lee, and K. Chon, Proactive Web Caching with Cumulative Prefetching for Large Multimedia Data. Procs. of the 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000.

[Khan99] Javed I. Khan, Ordering Prefetch in Trees, Sequences and Graphs, Technical Report 1999-12-03, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet]

[Khan00] Javed I. Khan, Active Streaming in Transport Delay Minimization, Workshop on Scalable Web Services, Toronto, August 2000, pp95-102.

[KrLM97] T. Kroeger, D. D. E. Long & J. Mogul, Exploring the Bounds of Web Latency Reduction from Caching and Prefetching, Proc. Of USENIX Symposium on Internet Technology and Systems, Monterey, December 1997, pp-319-328.

[NLAN99] NLANR, Proxy cache log traces, December 1999, ftp://ircache.nlanr.net/Traces/.

[PaMe99] T. Palpanas and A. Mendelzon,, Web Prefetching Using Partial Match Prediction, WWW Caching Workshop, San Diego, CA, March 1999

[PiPi99a] P. Pirolli and J. E. Pitkow, Distributions of surfers' paths through the World Wide Web: Empirical characterizations, Jounral of World Wide Web, 1999, v.1-2, pp29-45

[SeRT99] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In Proceedings of the IEEE INFOCOM' 99 Conference, 1999.

[WaCr96] Z. Wang and J. Crowcroft, Prefetching in the World Wide Web. in procs. of IEEE Global Internet, London, UK, 1996.

[KhTa01] Javed I. Khan and Qingping Tao, Partial Prefetch for Faster Surfing in Composite Hypermedia, Proc. of USENIX Symposium on Internet Technology and Systems, San Francisco, March 2001, (in press).

[HaVR94] Linda Hardman, D.C.A. Bulterman, G. V. Rossum, The Amstrerdam Hypermedia Model, Communications of the ACM, February 1994, v.37, no.2, pp-50-62.