

**Final Report for Period:** 09/2007 - 08/2008

**Submitted on:** 12/03/2008

**Principal Investigator:** Sivakumar, Raghupathy .

**Award ID:** 0519733

**Organization:** GA Tech Res Corp - GIT

**Submitted By:**

**Title:**

NetS-NBD: Principles and Design of Unified Transport Layer Solutions for Heterogeneous Wireless Data Networks

**Project Participants**

**Senior Personnel**

**Name:** Sivakumar, Raghupathy

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

**Post-doc**

**Graduate Student**

**Name:** Chang, Tae-Young

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Graduate student working on the project.

**Name:** Zhuang, Zhenyun

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Graduate student working on the project.

**Name:** Tsao, Cheng-lin

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

Graduate student who helped in the simulations and algorithm design.

**Name:** Kakamanu, Sandeep

**Worked for more than 160 Hours:** Yes

**Contribution to Project:**

**Undergraduate Student**

**Technician, Programmer**

**Other Participant**

**Research Experience for Undergraduates**

**Organizational Partners**

**Other Collaborators or Contacts**

## Activities and Findings

### **Research and Education Activities:**

A tremendous amount of research has been done toward improving realtime and non-realtime application performance over wireless data networks. Over the course of the past three years we have focused on developing new transport principles for such applications [13], [27], [28]. Our focus included a variety of applications including enterprise applications, Voice-over-IP (VoIP) applications, web applications, and peer to peer applications [14], [20].

The findings from the work have been included in the graduate level class on wireless networking in the School of Electrical and Computer Engineering in Georgia Tech.

### **Findings: (See PDF version submitted by PI at the end of the report)**

In this project, we attempt to design transport principles for improving the delivery of typical applications over wireless data networks. We broadly classify applications into four categories:

ò Enterprise Applications: Many of today's enterprise applications including CIFS and SMTP are designed and tuned well under the wired networks such as LANs. As they move into the wireless and mobile environment, these applications often experience degraded performance. In this report, we argue that these performance downgrade cannot be solved by traditional transport protocols as application behaviors may render any lower layer optimizations ineffective. We identify a set of such typical application behaviors. Based on the insights gained, we propose a set of transport principles in the form of a solution suite called A3 (application aware acceleration). This solution is application aware, but application transparent. When deployed appropriately, no application modifications is needed. We evaluate the performance of A3 using emulation-based test bed. We also build a proof-of-concept prototype of A3 which can work with real applications.

ò Voice Over IP The expected VoIP call capacity in a one-hop IEEE 802.11b network with G.711 voice codec is about 85 simultaneous calls, but the actual observed capacity is only 5 calls even at the highest rate and under zero loss conditions. In this report, we first provide test bed results for the actual VoIP call capacity in an IEEE 802.11b network; we then perform mathematical analysis for the call capacity in an IEEE 802.11 network to explain the poor performance observed, and identify three dominant components that can be exploited to improve the call capacity. We then present a set of three transport principles that improve the characteristics of the three components, and thereby lead to an increase in the VoIP call capacity of IEEE 802.11 WLANs. Finally, using ns2-based simulations, we evaluate the algorithms and show that performance improvements of up to 300% can be achieved.

ò Web-based Applications Current popular web-browsers simply fetch the entire web-page from the web-server in a greedy fashion. This simple web fetching mechanism employed by browsers is inappropriate for use in low-bandwidth networks, since they cause large response times for users unnecessarily. In this report, we first analyze the reasons that cause large response times by considering several factors including the properties of typical web-pages, the properties of current web-browsers, the interaction of the HTTP and TCP protocols, and the impact of server-side optimization techniques. We then propose transport principles that may be deployed purely on the client-side to reduce the user response time. Through ns2-based simulations, we compare the performance of our solution with that of current browsers and show that the proposed scheme brings significant performance benefits in terms of user-perceived response times.

ò Peer to Peer Applications Over the last few years, peer-to-peer (P2P) data sharing applications have experienced an explosive growth. In this report, we explore the question of what is the performance of mobile users when participating in P2P data sharing? We started the exploration with experiments running on a real BitTorrent test bed. Using insights gained through the aforementioned study, we present a set of new transport principles in the form of deployable solution suite called wireless P2P (wP2P) that addresses the issues using changes only to the P2P application at the mobile host. wP2P uses techniques transparent to the fixed peer, but uniquely relevant to the specific issues pertaining to wireless and mobile hosts functioning in a P2P data network.

While we present the approaches and approaches as stand-alone transparent mechanisms, the realistic long-term adoption strategy for these are as integrated transport layer mechanisms.

### **Training and Development:**

Four graduate students have been funded under this project over the last years. All four of them have developed substantially in terms of

understanding problems users of wireless data networks face, and key mechanisms that help in alleviating the problems. One of the graduate students has graduated and taken up a research engineer position at a wireless service provider.

The PI also teaches the class on wireless networks in the School of ECE in Gatech, and he has included the findings from the research in the curriculum for the class.

### **Outreach Activities:**

The work has resulted in several publications in International conferences/journals.

### **Journal Publications**

Tae-Young Chang, Zhenyun Zhuang, Aravind Velayutham, and RaghupathySivakumar, "WebAccel: Accelerating Web Access for Low-Bandwidth Hosts", Elsevier Computer Networks, p. , vol. , (2008). Accepted,

Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham, "Application-Aware Acceleration for Wireless Data Networks: Design Elements and Prototype Implementation", IEEE Transactions on MobileComputing, p. , vol. , (2008). Submitted,

Zhenyun Zhuang, Sandeep Kakumanu, Yeonsik Jeong, Raghupathy Sivakumar, and Aravind Velayutham, "Mobile Hosts Participating in Peer-to-Peer Data Networks: Challenges and Solutions", ACM/Kluwer Wireless Networks Journal (WINET), p. , vol. , (2008). In preparation,

Yeonsik Jeong, Sandeep Kakumanu, Cheng-Lin Tsao, and RaghupathySivakumar, "VoIP over Wi-Fi Networks: Performance Analysis and Acceleration Algorithms", ACM/Kluwer Mobile Networks and Applications Journal (MONET), p. , vol. , (2008). Submitted,

### **Books or Other One-time Publications**

Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham, "A3: Application-Aware Acceleration for Wireless Data Networks.", (2006). Proceedings, Published  
Collection: ACM International Conference on Mobile Computing and Networking (MOBICOM)  
Bibliography: A: Application-Aware Acceleration for Wireless Data Networks, Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham, ACM International Conference on Mobile Computing and Networking

Yeonsik Jeong, Sandeep Kakumanu, Cheng-Lin Tsao, and Raghupathy Sivakumar, "Improving VoIP Call Capacity over IEEE 802.11 Networks", (2007). Proceedings, Published  
Collection: International Conference on Broadband Communications, Networks and Systems (Broadnets),  
Bibliography: Improving VoIP Call Capacity over IEEE 802.11 Networks, Yeonsik Jeong, Sandeep Kakumanu, Cheng-Lin Tsao, and Raghupathy Sivakumar, International Conference on Broadband Communications and Systems

Tae-Young Chang, Zhenyun Zhuang, Aravind Velayutham, and Raghupathy Sivakumar, "Client-Side Web Acceleration for Low-Bandwidth Hosts", (2007). Proceedings, Published  
Collection: International Conference on Broadband Communications, Networks and Systems (Broadnets)  
Bibliography: Client-Side Web Acceleration for Low-Bandwidth Hosts, Tae-Young Chang, Zhenyun Zhuang, Aravind Velayutham, and Raghupathy Sivakumar, The Fourth International Conference on Broadband Communications and Systems

Zhenyun Zhuang, Sandeep Kakumanu, Yeonsik Jeong, Raghupathy Sivakumar, and Aravind Velayutham, "On the Impact of Mobile Hosts in Peer-to-Peer Data Networks", (2008). Proceedings, Published  
Collection: International Conference on Distributed Computing Systems (ICDCS 2008)  
Bibliography: On the Impact of Mobile Hosts in Peer-to-Peer Data Networks, Zhenyun Zhuang, Sandeep Kakumanu, Yeonsik Jeong, Raghupathy Sivakumar, and Aravind Velayutham, The 28th International Conference on Distributed Computing Systems

Web/Internet SiteOther Specific ProductsContributions**Contributions within Discipline:**

A tremendous amount of research has been done toward improving transport layer performance over wireless data networks. The improved transport layer protocols are typically application unaware. Over the past one year, the project has focused on identifying the first set of principles for improving performance of users in wireless data networks. The findings indicate that the behavior of applications can and do dominate the actual performance experienced. More importantly, for practical applications, application behavior all but completely negates any improvements achievable through better transport layer protocols.

In this context, the project has developed two solution suites:

- (i) An application aware but application-transparent solution called (SA<sup>3</sup>T<sup>\$</sup>) that can be applied to many general applications; and
- (ii) An application aware but application-changing solution called (*Cut-Load*) that is specifically designed for the web application.

**Contributions to Other Disciplines:****Contributions to Human Resource Development:****Contributions to Resources for Research and Education:**

International publications, Curriculum development

**Contributions Beyond Science and Engineering:**Categories for which nothing is reported:

Organizational Partners

Any Web/Internet Site

Any Product

Contributions: To Any Other Disciplines

Contributions: To Any Human Resource Development

Contributions: To Any Beyond Science and Engineering

# NSF CNS-0519733

## Principles and Design of Transport Layer Solutions for Heterogeneous Wireless Data Networks

### Final Report

Faculty: Raghupathy Sivakumar

Students: Tae-Young Chang, Cheng-Lin Tsao, Zhenyun Zhuang, and Sandeep Kakumanu

School of Electrical and Computer Engineering  
Georgia Institute of Technology  
Atlanta, GA 30332

#### I. OVERVIEW

A tremendous amount of research has been done toward improving realtime and non-realtime application performance over wireless data networks. Over the course of the past three years we have focused on developing new transport principles for such applications [13], [27], [28]. Our focus included a variety of applications including *enterprise applications*, *Voice-over-IP (VoIP) applications*, *web applications*, and *peer to peer applications* [14], [20]. The following is a summary of the primary results.

In this project, we attempt to design transport principles for improving the delivery of typical applications over wireless data networks. We broadly classify applications into four categories:

- *Enterprise Applications*: Many of today's enterprise applications including CIFS and SMTP are designed and tuned well under the wired networks such as LANs. As they move into the wireless and mobile environment, these applications often experience degraded performance. In this report, we argue that these performance downgrade cannot be solved by traditional transport protocols as application behaviors may render any lower layer optimizations ineffective. We identify a set of such typical application behaviors. Based on the insights gained, we propose a set of transport principles in the form of a solution suite called  $A^3$  (application aware acceleration). This solution is application aware, but application transparent. When deployed appropriately, no application modifications is needed. We evaluate the performance of  $A^3$  using emulation-based test bed. We also build a proof-of-concept prototype of  $A^3$  which can work with real applications.

- *Voice Over IP* The expected VoIP call capacity in a one-hop IEEE 802.11b network with G.711 voice codec is about 85 simultaneous calls, but the actual observed capacity is only 5 calls even at the highest rate and under zero loss conditions. In this report, we first provide test bed results for the actual VoIP call capacity in an IEEE 802.11b network; we then perform mathematical analysis for the call capacity in an IEEE 802.11 network to explain the poor performance observed, and identify three dominant components that can be exploited to improve the call capacity. We then present a set of three transport principles that improve the characteristics of the three components, and thereby lead to an increase in the VoIP call capacity of IEEE 802.11 WLANs. Finally, using *ns2*-based simulations, we evaluate the algorithms and show that performance improvements of up to 300% can be achieved.

- *Web-based Applications* Current popular web-browsers simply fetch the entire web-page from the web-server in a greedy fashion. This simple web fetching mechanism employed by browsers is inappropriate for use in low-bandwidth networks, since they cause large response times for users unnecessarily. In this report, we first analyze the reasons that cause large response times by considering several factors including the properties of typical web-pages, the properties of current web-browsers, the interaction of the HTTP and TCP protocols, and the impact of server-side optimization techniques. We then propose transport principles that may be deployed purely on the

client-side to reduce the user response time. Through *ns2*-based simulations, we compare the performance of our solution with that of current browsers and show that the proposed scheme brings significant performance benefits in terms of user-perceived response times.

- *Peer to Peer Applications* Over the last few years, peer-to-peer (P2P) data sharing applications have experienced an explosive growth. In this report, we explore the question of *what is the performance of mobile users when participating in P2P data sharing?* We started the exploration with experiments running on a real BitTorrent test bed. Using insights gained through the aforementioned study, we present a set of new transport principles in the form of deployable solution suite called wireless P2P (*wP2P*) that addresses the issues using changes only to the P2P application at the mobile host. *wP2P* uses techniques transparent to the fixed peer, but uniquely relevant to the specific issues pertaining to wireless and mobile hosts functioning in a P2P data network.

While we present the approaches and approaches as stand-alone transparent mechanisms, the realistic long-term adoption strategy for these are as integrated transport layer mechanisms.

## II. STUDENTS SUPPORTED

The graduate students supported under this grant are Mr. Tae-Young Chang, Mr. Cheng-Lin Tsao, Mr. Zhenyun Zhuang, and Mr. Sandeep Kakumanu.

## III. PRINCIPLES FOR ENHANCING ENTERPRIZE APPLICATIONS

### A. Overview

A significant amount of research has been done toward the development of better transport layer protocols that can alleviate the problems Transmission Control Protocol (TCP) exhibits in wireless environments [11], [16], [18], [25]. Such protocols, and several more, have novel and unique design components that are indeed important for tackling the unique characteristics of wireless environments. However, in this work we ask a somewhat orthogonal question in the very context the above protocols were designed for: *How does the application's behavior impact the performance deliverable to wireless users?*

Toward answering this question, we explore the impact of typical wireless characteristics on the performance experienced by the applications for very popularly used enterprize applications including File Transfer Protocol (FTP), the Common Internet File Sharing (CIFS) protocol [3], the Simple Mail Transfer Protocol (SMTP) [8], and the Hyper-Text Transfer Protocol (HTTP) [5]. Through our experiments, we arrive at an impactful result: Except for FTP which has a simple application layer behavior, *for all other applications considered, not only is the performance experienced when using vanilla TCP-NewReno much worse than for FTP, but the applications see negligible or no performance enhancements even when they are made to use the wireless-aware protocols.*

We delve deeper into the above observation and identify several common behavioral characteristics of the applications that fundamentally limit the performance achievable when operating over wireless data networks. Such characteristics stem from the design of the applications, which is typically tailored for operations in substantially higher quality local area network (LANs) environments. Hence, we pose the question: *if application behavior is a major cause for performance degradation as observed through the experiments, what can be done to improve the end-user application performance?*

In answering the above question, we present a new solution called *Application-Aware Acceleration* ( $A^3$ , pronounced as "A-cube"), which is a middleware that offsets the typical behavioral problems of real-life applications through an effective set of principles and design elements.  $A^3$ 's design has five underlying design principles including transaction prediction, prioritized fetching, redundant and aggressive retransmissions, application aware encoding, and infinite buffering. The design principles are derived explicitly with the goal of addressing the aforementioned application layer behavioral problems. We present  $A^3$  as a platform solution requiring entities at both ends of the end-to-end communication, but also describe a variation of  $A^3$  called  $A^3\bullet$  (pronounced as "A-cube dot"), which is a point solution but is not as effective as  $A^3$ . One of the keystone aspects of the  $A^3$  design is that it is *application-aware, but application transparent.*

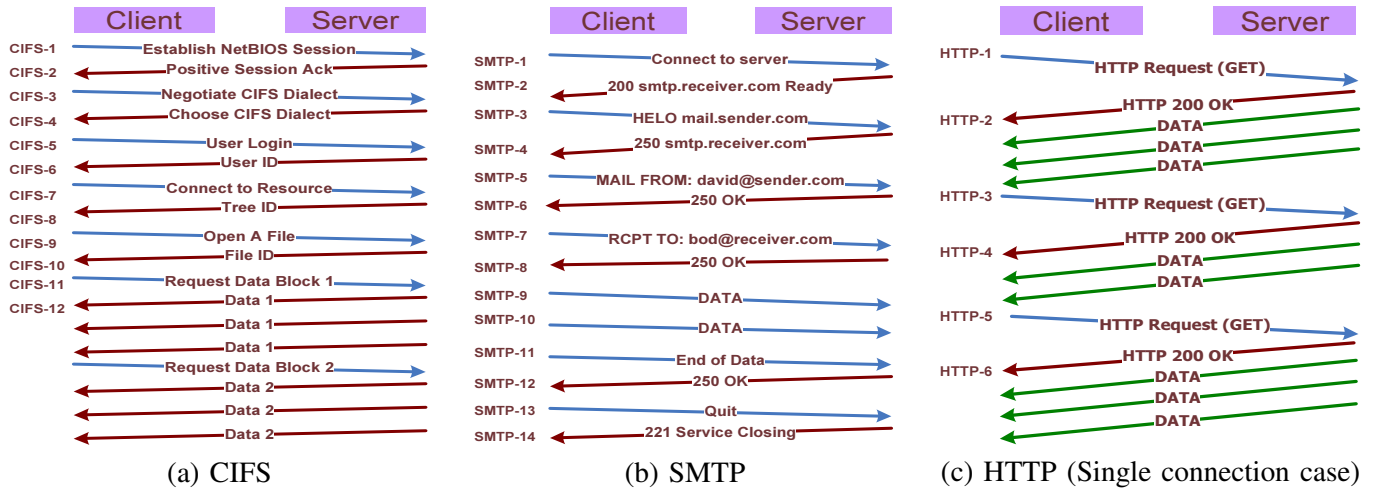


Fig. 1. Application Traffic Patterns

## B. Motivation

The focus of this work is entirely on applications that require reliable and in-sequenced packets delivery. In other words, we consider only applications that are traditionally developed with the assumption of using the TCP transport layer protocol.

1) *Setup*: We use IxChariot [19] to generate accurate application specific traffic patterns. IxChariot is a commercial tool for emulating most real-world applications. It is comprised of the IxChariot console (for control), performance end-points (for traffic generation and reception), and IxProfile (for characterizing performance). We use a combination of a real test-bed and emulation to construct the test-bed for the results presented in the section. Since IxChariot is a software tool that generates actual application traffic, it is hosted on the sender and the receiving machines. The path from the sender to the receiver goes through a node running the Network Simulator (NS2) [26] in emulation mode. The network emulator is configured to represent desired topologies including the different types of wireless technologies. Since we consider wireless LANs (WLAN), wireless WANs (WWAN), and wireless satellite area networks (SAT), we use transport layer protocols proposed in related literature for each of these environments. Specifically, we use TCP-ELN (NewReno with Explicit Loss Notification) [11], WTCP (Wide-area Wireless TCP) [25], and STP (Satellite Transport Protocol) [16] as enhanced transport protocols for WLANs, WWANs, and SATs respectively.

2) *Results*: For FTP, the tailored protocols uniformly show considerable performance improvements. The results illustrate that the design of the enhancement protocols such as TCP-ELN, WTCP, and STP, is sufficient enough to deliver considerable improvements in performance for wireless data networks, *when using FTP as the application*. For applications other than FTP, we observe that *the performance improvements demonstrated by the enhancement protocols for FTP do not carry over to these three applications*. We also observe that the maximum performance improvement delivered by the enhancement protocols is less than 5 % across all scenarios.

While the trend evident from the results discussed above is that the enhanced wireless transport protocols do not provide performance improvements for three very popularly used applications, we argue in the rest of the section that *this is not due to any fundamental limitations of the transport protocols themselves, but due to the specifics of the behavior of the three applications under consideration*.

3) *Impact of Application Behavior*: We now explain the lack of performance improvements when using enhanced wireless transport protocols with applications such as CIFS, SMTP, and HTTP. We use the conceptual application traffic pattern for the three applications in Figure 1 for most of our reasonings [3], [5], [8].

- *Thin session control messages* All three applications, as can be observed in Figure 1, use *thin session control message exchanges* before the actual data transfer occurs, and *thin request messages* during the actual data transfer phase as well. We use the term “thin” to refer to the fact that such messages are almost always contained in a single packet of MSS (maximum segment size).

The observation above has two key consequences: (i) When a loss occurs to a thin message, an entire round trip time (RTT) is taken to recover from such a loss. When the round-trip time is large like in WWANs and SATs, this can result in considerably inflating the overall transaction time for the applications. Note that a loss during the data phase will not have such an adverse impact, as the recovery of that loss can be multiplexed with other new data transmissions, whereas for thin message losses, no other traffic can be sent anyway. (ii) Most protocols, including TCP, rely on the arrival of out-of-order packets to infer packet losses and hence trigger loss recovery. In the case of thin messages, since there are no packets following the lost message, the only means for loss detection is the expiry of the retransmission timer. Retransmission timers typically have coarse minimum values to keep overheads low. TCP, for example, typically uses a minimum Retransmission Time Out (RTO) value of *one second*.<sup>1</sup>

- *Batched data fetches* Another characteristic of the applications, especially CIFS and HTTP, is that although the total amount of data to be fetched can be large, *the data transfer is performed in batches, with each batch including a “request-response” exchange*. CIFS uses its *request-data-block* message to send the batched requests, with each request typically requesting only 16 KB to 32 KB of data. Such a batched fetching of data has two implications to performance: (i) When the size of the requested data is smaller than the Bandwidth Delay Product (BDP), there is a gross under-utilization of the available resources. Hence, when the SAT network has a BDP of 128 KB, and CIFS uses a 16 KB request size, the utilization is only 12.5 %. (ii) Independent of the size of each requested data batch, one *rtt* is spent in sending the next request once the current requested data arrives. When the *RTT* of the path is large like in WWANs and SATs, this can inflate the overall transaction time and hence lower throughput performance.

- *Flow control bottlenecked operations* Flow control is an important function in communication that helps in preventing the source from overwhelming the receiver. In a mobile/wireless setting, flow control can kick in and prove to be the bottleneck for the connection progress due to two reasons: (i) If the application on the mobile device reads slowly or is temporarily halted for some other reason, the receiver buffer fills up and the source is eventually frozen till the buffer empties. (ii) When there are losses in the network, and the receiver buffer size is of the same order as the BDP (which is typically true), flow control can prevent new data transmissions even when techniques such as fast recovery is used due to unavailability of buffer space at the receiver. The dominant effect of flow control is however undesirable in wireless environments because of its resultant low throughput performance.

- *Other reasons* While the above discussed reasons are behavioral “acts of commission” by the applications that result in lowered performance, we now discuss two more reasons that can be seen as behavioral “acts of omission”. These are techniques that the applications could have used to address conditions in a wireless environment, but do not. (i) *Non-prioritization of data*: For all three applications considered, no explicit prioritization of data to be fetched is performed, and hence all the data to be fetched are given equal importance. However, for certain applications prioritizing data in a meaningful fashion can have a profound impact on the performance experienced by the end-system or user. For example, consider the case of HTTP used for browsing on a small-screen PDA. When a web page URL request is issued, HTTP fetches all the data for the web page with equal importance. However, the data corresponding to the *visible* portion of the webpage on the PDA’s screen is obviously of more importance and will have a higher impact on the perceived performance by the end-user. Not leveraging such means of prioritizing data hence results in HTTP suffering performance as defined by the original data size and the low bandwidths of the wireless environment. (ii) *Non-use of data reduction techniques*: Finally, another issue is applications not using knowledge specific to their content or behavior to employ effective data reduction techniques. For example, considering the SMTP application, “email vocabulary” of users has evolved over the last couple of decades to be very independent of traditional “writing vocabulary” and “verbal vocabulary” of the users. Hence, it is an interesting question as to whether SMTP can use email vocabulary based techniques to reduce the actual content transferred between SMTP servers, or a SMTP server and a client. Not leveraging such aspects prove to be of more significance in wireless environments where the baseline performance is poor to start with.

<sup>1</sup>While newer Linux releases have lower minimum RTO values, they still are in the order of several hundred ms.

### C. Design

Since we have outlined several behavioral problems with applications, an obvious question to ask is: “*Why not change the applications to address these problems?*” We believe that is indeed one possible solution. Hence, we structure the presentation of the A<sup>3</sup> solution into two distinct components: (i) the key design elements or principles that underlie A<sup>3</sup>; and (ii) the actual realization of the design elements for specific applications in the form of an *optimization middleware that is application-aware, but application transparent*. The design elements generically present strategies to improve application behavior and can be used by application developers to improve performance by incorporating changes to the applications directly. In the rest of this section, we outline the design of five principles in the A<sup>3</sup> solution.

- *Transaction Prediction (TP)* Transaction prediction (TP) is an approach to *deterministically predict* future application data requests to the server, and issue them ahead of time. Note that this is different from techniques such as “prefetching” where content is heuristically fetched to speed up later access but is not guaranteed to be used. In TP, A<sup>3</sup> is fully aware of application semantics and knows exactly what data to fetch and that the data will be used. TP will aid in conditions where the BDP is larger than the default application batch fetch size and where the *RTT* is very large. Under both cases, the overall throughput will improve when TP is used.

- *Redundant and Aggressive Retransmissions (RAR)* Redundant and aggressive retransmissions (RAR) is an approach to protect thin session control and data request messages better from losses. The technique involves recognizing thin application messages, and using a combination of packet level redundancy, and aggressive retransmissions to protect such messages. RAR will help address both issues with thin messages. The redundant transmissions reduce the probability of message losses, and the aggressive retransmissions that operate on tight *RTT* granularity timeouts reduce the loss recovery time. The key challenges in RAR is to recognize thin messages in an application-aware fashion. Note that *only thin messages require RAR because of reasons outlined above*. Regular data messages should not be subjected to RAR both because their loss recovery can be masked in the overall transaction time by performing the recovery simultaneously with other data packet transmissions, and because the overheads of performing RAR will become untenable when applied to large volume messages such as the data.

- *Prioritized Fetching (PF)* Prioritized fetching (PF) is an approach to prioritize subsets of data to be fetched as being more important than others and to fetch the higher priority data *faster* than the lower priority data. A simple approach to achieve the dual-rate fetching is to use default TCP-like congestion control for the high priority data, but use congestion control like in TCP-LP [22] for low priority data. An important consideration in PF is to devise a strategy to prioritize data intelligently and on the fly.

- *Infinite Buffering (IB)* Infinite buffering (IB) is an approach that prevents flow control from throttling the progression of a network connection terminating at the mobile wireless device. IB prevents flow control from impacting performance by providing the sender the impression of an *infinite buffer* at the receiver. Secondary storage is used to realize such an infinite buffer, with the main rationale being that *reading from the secondary storage will be faster than fetching it from the sender over the wireless network when there is space created in the actual connection buffer at a later point*. With typical hard-disk data transfer rates today being at around 250 Mbps [6], the above mentioned rationale is well justified for wireless environments. Note that the trigger for using IB can be both due to application reading slowly or temporarily not reading from the connection buffer, and due to losses on the wireless path.

- *Application-aware Encoding (AE)* Application-aware encoding (AE) is an approach that uses application specific information to better encode or compress data during communication. Traditional compression tools such as *zip* operate on a given content in isolation without any context for the application corresponding to the content. AE, on the other hand, explicitly uses this contextual information to achieve better performance. Note that AE is not a better compression algorithm. However, it is a better way of identifying data-sets that need to be operated on by a given compression algorithm.

## IV. SOLUTION

### A. Deployment Model and Architecture

The A<sup>3</sup> deployment model is shown in Figure 2. Since A<sup>3</sup> is a platform solution, it requires two entities at either end of the communication session that are A<sup>3</sup>-aware. At the mobile device, A<sup>3</sup> is a software module that is

installed in user space. At the server side, while  $A^3$  can be deployed as a software module on all servers, a more elegant solution would be to deploy a packet processing network appliance that processes all content flowing from the servers to the wide-area network. We assume the latter model for our discussions. However, note that  $A^3$  can be deployed in either fashion as it is purely a software solution.

This deployment model will help in any communication between a server behind the  $A^3$  server and the mobile device running the  $A^3$  module. However, if the mobile device communicates with a non- $A^3$  enabled server, two options exist: (i) As we discuss later in the paper,  $A^3$  can be used as a point-solution with lesser effectiveness; or (ii) the  $A^3$  server is brought closer to the mobile device, perhaps within the wireless network provider’s access network. In the rest of the paper, we do not delve into the latter option. However, we do revisit the point-solution mode of operation of  $A^3$ .

We present an  $A^3$  implementation that resides in user-space, and uses the NetFilter utility in Linux for the capturing of traffic outgoing and incoming at the mobile device. NetFilter is a Linux specific packet capture tool that has hooks at multiple points in the Linux kernel. The  $A^3$  hooks are registered at the Local-In and Local-Out stages of the chain of hooks in NetFilter. While our discussions are Linux-centric, our discussions can be mapped on the Windows operating system through the use of the Windows Packet Filtering interface, or wrappers such as PktFilter that are built around the interface. Figure 3(a) shows the  $A^3$  deployment on the mobile device using NetFilter.

The  $A^3$  software architecture is shown in Figure 3(b). Since the design elements in  $A^3$  are to a large extent independent of each other, a simple chaining of the elements in an appropriate fashion results in an integrated  $A^3$  architecture. The specific order in which the elements are chained in the  $A^3$  realization is TP, RAR, PF, IB, and AE. While RAR protects the initial session control exchanges and the data requests, it operates on traffic after TP, given that TP can generate new requests for data. PF manipulates the priority with which different requests are served, and IB ensures that data responses are not throttled by flow control. Finally, AE compresses any data outgoing, and decompresses any data incoming.

### B. Application Overviews

Since we describe the actual operations of the mechanisms in  $A^3$  in the context of one of the three applications, we now briefly comment on the specific message types involved in typical transactions by those applications. We then refer to the specific message types when describing the operations of  $A^3$  subsequently.

Due to lack of space, instead of presenting all message types again, we refer readers back to Figure 1 to observe the message exchanges for the three applications. The labels such as CIFS-x refer to particular message types in CIFS and will be referred to in the  $A^3$  realization descriptions that follow.

CIFS, also sometimes known as Server Message Block (SMB), is a platform independent protocol for file sharing. The typical message exchanges in a CIFS session are as shown in Figure 1(a). Overall, TP manipulates the CIFS-11 message, RAR operates on CIFS-1 through CIFS-11, and IB aids in CIFS-12.

SMTP [8] is Internet’s standard host-to-host mail transport protocol and traditionally operates over TCP. The typical message exchanges in an SMTP session are shown in Figure 1(b). Overall, RAR operates on SMTP-1 through SMTP-8, and SMTP-12 through SMTP-14, IB and AE operates on SMTP-9 and SMTP-10.

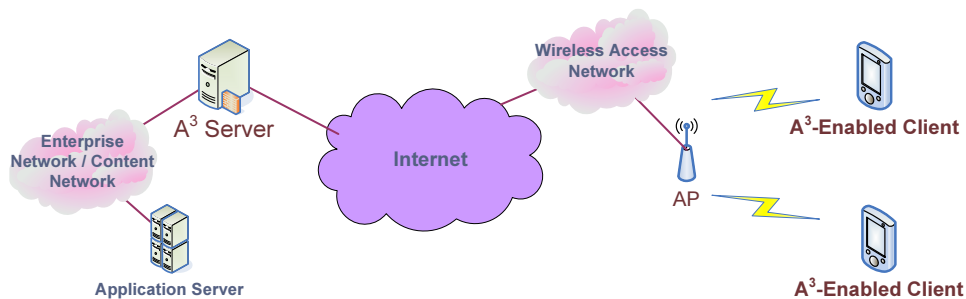


Fig. 2. Deployment Model

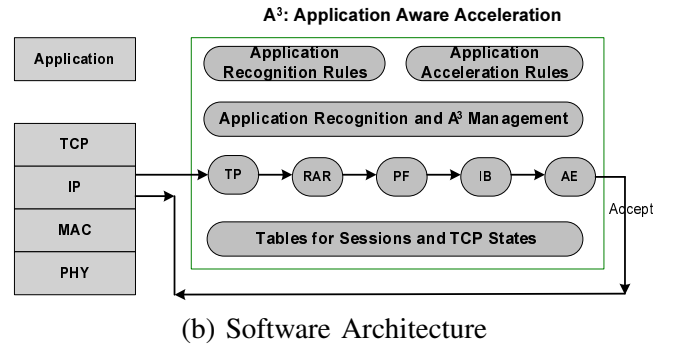
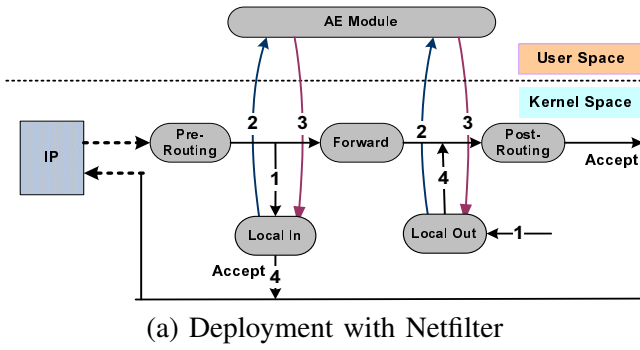


Fig. 3. A<sup>3</sup> Deployment Model with NetFilter and Software Architecture

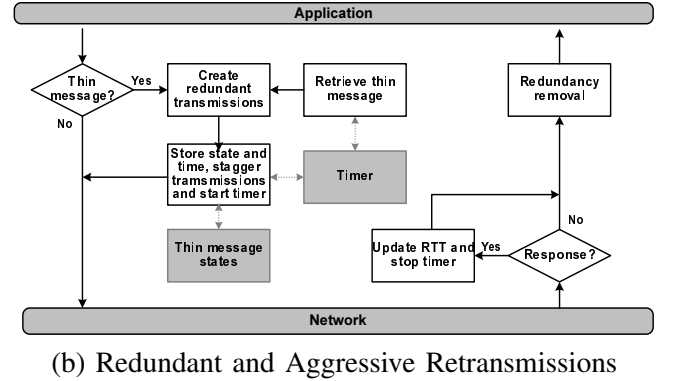
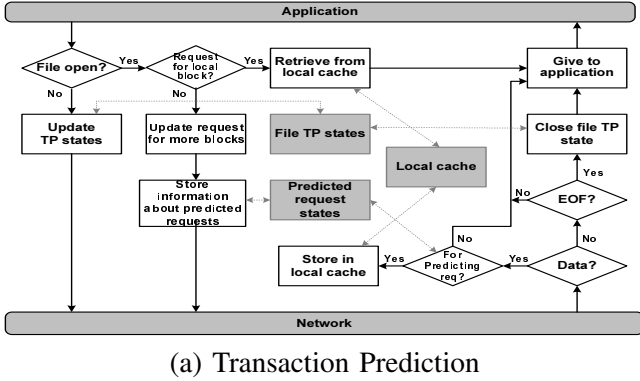


Fig. 4. TP and RAR (Shaded blocks are storage space and timer, white blocks are operations.)

The HTTP message exchange standard are relatively simple, and typically consist of the messages shown in Figure 1(c). A typical HTTP session consists of multiple objects, as well as the main HTML file, and hence appear as a sequence of overlapping exchanges of the above format. Overall, RAR operates on HTTP-1,2,3,5; and PF and IB operate on HTTP-3,5.

### C. A<sup>3</sup> Realization

In the rest of the section, we take one design element at a time, and walk through the algorithmic details of the element with respect to a single application. Note that A<sup>3</sup> is an application-aware solution, and hence its operations will be application specific. Since we describe each element in isolation, we assume that the element resides between the application and the network. In an actual usage of A<sup>3</sup>, the elements will have to be chained as discussed earlier.

1) *Transaction Prediction*: Figure 4(a) shows the flow chart for the implementation of TP for CIFS at the A<sup>3</sup> client. When A<sup>3</sup> receives a message from the application, it checks to see if the message is CIFS-9, and records state for the file transfer in its File-TP-States data structure. It then passes through the message. If the message was a request, TP checks to see if the request is for a locally cached block, or for a new block. If the latter, it updates the request for more blocks, stores information about the predicted requests generated in the Predicted-Request-States data structure, and forwards the requests.

In the reverse direction, when data comes in from the network, TP checks to see if the data is for a predicted request. If yes, it caches the data in secondary storage and updates its state information, and forwards the data to the application otherwise.

The number of additional blocks to request is an interesting design decision. For file transfer scenarios, TP generates requests asking for the entire file. The file size information can be retrieved from the CIFS-10 message. If the incoming message is for an earlier retrieved block, TP retrieves the block from secondary storage, and provides it to the application.

While CIFS servers accept multiple data requests from the same client simultaneously, it is possible that for some applications, the server might not be willing to accept multiple data requests simultaneously. In such an event, the

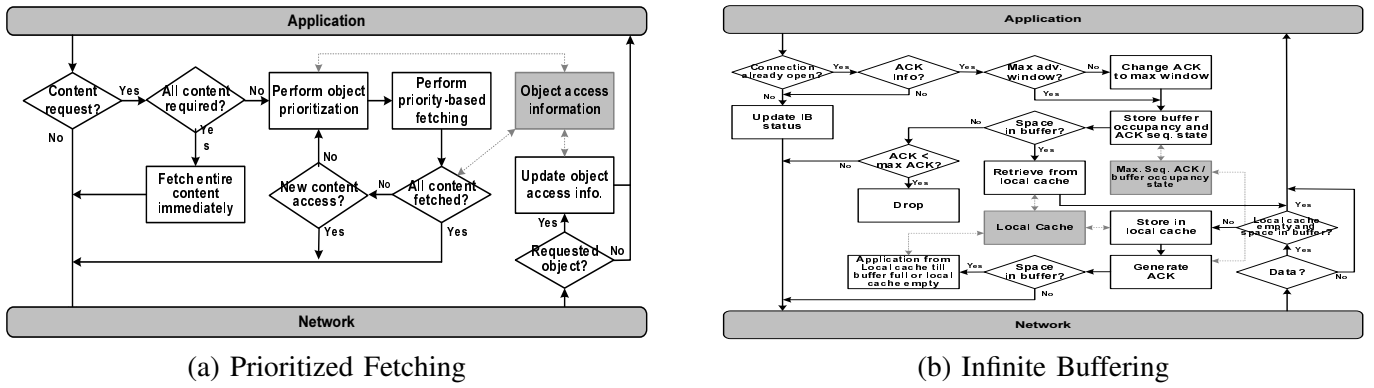


Fig. 5. PF and IB (Shaded blocks are storage space and timer, white blocks are operations.)

$A^3$  server will let only one of the client requests go through to the server at any point in time, and will send the other requests one at a time once the previous requests are served.

2) *Redundant and Aggressive Retransmissions*: Figure 4(b) shows the flow chart for the implementation of RAR for CIFS. When  $A^3$  receives a message from the application, it checks to see if it is a thin message. The way  $A^3$  performs the check is to see if the message is one of the messages between CIFS-1 and CIFS-11. All such messages are interpreted as thin messages.

If the incoming message is not a thin one,  $A^3$  will let it through as-is. Otherwise,  $A^3$  will create redundant copies of the message, note the information about current time, start retransmission alarm, and send out the copies in a staggering fashion. When a response arrives,  $A^3$  checks the timestamp for the corresponding request, and updates its estimated  $RTT$ .  $A^3$  then passes on the message to the application. If the alarm expires for a particular thin message, the message is again subjected to the redundant transmissions.  $A^3$  server is responsible for filtering the successful arrivals of redundant copies of the same message.

The key issues of interest in the RAR implementation are: (i) How many redundant transmissions are performed? Since packet loss rates in wireless data networks rarely exceed 10 %, even a redundancy factor of *two* (two additional copies created) reduces the effective loss-rate to about 0.1 %. Hence,  $A^3$  uses a redundancy factor of two. (ii) How should the redundant messages be staggered? The answer to this question lies in the specific channel characteristics experienced by the mobile device. However, at the same time, the staggered delay should not exceed the round-trip time of the connection, as otherwise the mechanism would lose its significance by unnecessarily delaying the recovery of losses. Hence,  $A^3$  uses a staggering delay of  $\frac{RTT}{10}$  between any two copies of the same message. This ensures that within 20 % of the  $RTT$  duration, all messages are sent out at the mobile device. (iii) How is the aggressive timeout value determined? Note that while the aggressive timeout mechanism will help under conditions when all copies of a message are lost, the total message overhead by such aggressive loss recovery is negligible when compared to the overall size of data transferred by the application. Hence,  $A^3$  uses a timeout value of the  $RTT_{avg} + \alpha$ , where  $\alpha$  is a small guard constant, and  $RTT_{avg}$  is the average  $RTT$  observed so far. This simple setting ensures that the timeout values are tight, and at the same time the mechanism adapts to changes in network characteristics.

3) *Prioritized Fetching*: Figure 5(a) shows the flow chart for the implementation of PF in the context of HTTP. Once again, the key goal in PF for HTTP is to retrieve web objects that are required for the display of the visible portion of the webpage quickly at the expense of the objects on the page that are not visible.

Unlike in the other mechanisms, PF cannot be implemented without some additional interactions with the application itself. Fortunately, browser applications have well defined interfaces for querying state of the browser including the current window focus, scrolling information, etc. Hence, the implementation of PF relies on a separate module called the application state monitor (ASM) that is akin to a browser plug-in to coordinate its operations.

When a message comes in from the application, PF checks to see if the message is a request. If it is not, it is let through. Otherwise, PF checks with the ASM to see if the requested content are immediately required. ASM classifies the objects requested as being of immediate need (*i.e.*, visible portion of webpage) or as those that are

not immediately required. PF then sends out fetch requests immediately for the first category of objects and uses a low-priority fetching mechanism for the remaining objects.

Since  $A^3$  is a platform solution, all PF has to inform the  $A^3$  server is that certain objects are of low priority through  $A^3$ -specific piggybacked information. The  $A^3$  server then de-prioritizes the transmission of those objects in preference to those that are of higher priority. Note that the relative prioritization is used not only between the content of a single end-device, but also across end-devices as well to improve overall system performance. Approaches such as TCP-LP [22] are candidates that can be used for the relative prioritization between TCP flows, although  $A^3$  currently uses a simple priority queuing scheme within the same TCP flow at the  $A^3$  server.

Note that while the ASM might classify objects in a particular fashion, changes in the application (e.g. scrolling down) will result in a re-prioritization of the objects accordingly. Hence, the ASM has the capability of gratuitously informing PF about priority changes. Such changes are immediately notified to the  $A^3$  server through appropriate requests.

4) *Infinite Buffering*: Figure 5(b) shows the flow chart for the implementation of IB in the context of SMTP. IB keeps track of TCP connection status, and monitors all ACKs that are sent out by the TCP connection serving the SMTP application for SMTP-9 and SMTP-10. If the advertised window in the ACK is less than the maximum possible, IB immediately resets the advertised window to the maximum value, and appropriately updates its current knowledge of the connection’s buffer occupancy and maximum in-sequence ACK information.

Hence, IB prevents anything less than the maximum buffer size from being advertised. However, when data packets arrive from the network, IB receives the packets and checks to see if the connection buffer can accommodate more packets. If the condition is true, IB delivers the packets to the application directly. If the disk cache is non-empty, which means the connection buffer is full, the incoming packet is directly added to the cache. In this case, IB generates a proxy ACK back to the server. Then, if the connection buffer has space in it, packets are retrieved from the disk cache and given to the application till the buffer becomes full again. When the connection sends an ACK for a packet already ACKed by IB, IB suppresses the ACK. When the connection state is torn down for the SMTP application, IB resets the state accordingly.

5) *Application-aware Encoding*: Figure 6 shows the flow-chart for the implementation of AE for SMTP. When AE receives data (SMTP-9) from the SMTP application, it uses its application vocabulary table to compress the data, and marks the message as being compressed and forwards it to the network. The marking is done to inform the  $A^3$  server about the need to perform de-compression. Similarly, when incoming data arrives for the SMTP server, and the data is marked as compressed, AE performs the necessary de-compression.

The mechanisms used for the actual creation and manipulation of the vocabulary tables are of importance to AE. In  $A^3$ , the SMTP vocabulary tables are created and maintained purely on a user pair-wise basis. Not only are the table created in this fashion, but the data sets over which the vocabulary tables are created is also restricted to this pair-wise model. In other words, if A is the sender and B is the receiver, A uses its earlier emails to B as the data set on which the A-B vocabulary table is created, and then uses this table for encoding. B, having the data set already (since the emails were sent to B), can exactly recreate the table on its side and hence decode any compressed data. This essentially precludes the need for exchanging tables frequently, and also takes advantage of

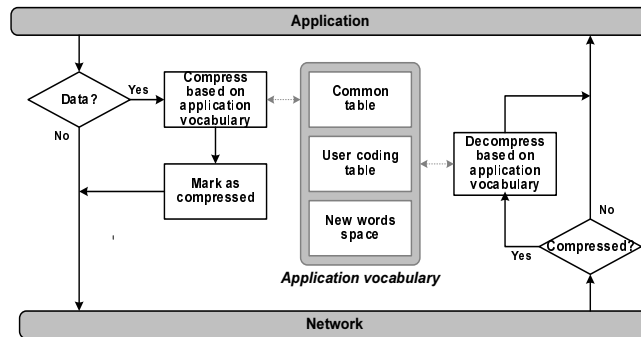


Fig. 6. Application-aware Encoding

changes in vocabulary sets that might occur based on the recipient. Though the tables are created on both sides implicitly and synchronized in most cases, a backup mechanism used to explicitly synchronize the tables is also needed. The synchronization action is triggered by a mismatch of table hashes on both sides, and the hash is sent along each new email and updated when the table changes.

#### D. $A^3$ Point Solution - $A^3\bullet$

While the  $A^3$  deployment model assumed so far is a platform model requiring participation by  $A^3$  enabled devices at both the client and server ends, in this section we describe how  $A^3$  can be used as a point-solution, albeit with somewhat limited capabilities. We refer to the point-solution version of  $A^3$  as  $A^3\bullet$ .

Of the five design elements in  $A^3$ , the only design element for which the platform model is mandatory is the application-aware encoding mechanism. Since compression or encoding is an end-to-end process,  $A^3\bullet$  cannot be used with AE. However, each of the other four principles can be employed with minimal changes in  $A^3\bullet$ .

TP involves the generation of predictive data requests, and hence can be performed in  $A^3\bullet$  as long as the application server can accept multiple simultaneous requests. For CIFS and HTTP, the servers do accept simultaneous requests. IB is purely a flow control avoidance mechanism, and can be realized in  $A^3\bullet$ . RAR involves redundant transmissions of messages, and hence can be implemented in  $A^3\bullet$  as long as application servers are capable of filtering duplicate messages. If the application servers are not capable of doing so (e.g. HTTP servers, which would respond to each request), the redundant transmissions will have to be performed at the granularity of transport layer segments as opposed to application layer messages, since protocols such as TCP provide redundant packet filtering. Finally, PF can be accomplished in  $A^3\bullet$  in terms of classifying requests and treating the requests differently. However, the slow fetching of data not required immediately has to be realized through coarser receiver based mechanisms such as delayed requests as opposed to the best possible strategy of slowing down responses as in  $A^3$ .

#### E. Performance Evaluation

The experimental Setup consists of three desktop machines running the Fedora Core 4 operating system with the Linux 2.6 kernel. All the machines are connected using 100 Mbps LAN. An application-emulator (AppEm) module runs on both the two end machines. The AppEm module is a custom-built user-level module that generates traffic patterns and content for three different application protocols: CIFS, SMTP, and HTTP.

The traffic patterns are modeled based on traffic traces generated by the IxChariot emulator, and documented standards for the application protocols. The AppEm module also generates traffic content based on both input real-life data-sets (for email and Web content) and random data-sets (File transfer)<sup>2</sup>. The traffic patterns shown in Figure 1 are representative of the traffic patterns generated by AppEm.

The system connecting the two end-systems runs the emulators for both  $A^3$  and the wireless network. Both the emulators,  $A^3$ -Em and WNetEm, are implemented within the framework of the ns2 simulator, and ns2 is running in the emulation mode. Running ns2 in its emulation mode allows for the capture and processing of live network traffic. The emulator object in ns2 taps directly into the device driver of the interface cards to capture and inject real packets into the network.

All five  $A^3$  mechanisms are implemented in the  $A^3$ -Em module, and each mechanism can be enabled either independently or in tandem with the other mechanisms. The WNetEm module is used for emulating different wireless network links representing the WLAN, WWAN, and SAT environments.

The primary metrics monitored are throughput, response time (for HTTP) and confidence intervals for the throughput and response time. Each data point is the average of 20 simulation runs and in addition we show the 90 % confidence intervals. The results of the evaluation study are presented in two stages. We first present the results of the performance evaluation of  $A^3$  principles in isolation. Then, we discuss the combined performance improvements delivered by  $A^3$ .

We present the results of the combined effectiveness of all applicable principles for the three applications, CIFS, SMTP and HTTP. We employ RAR, TP, and IB on the CIFS traffic. For SMTP, the RAR, AE and IB principles are used. For HTTP, the  $A^3$  principles applied are RAR, PF and IB. As expected, the throughput of the

<sup>2</sup>While the IxChariot emulator can generate representative traffic traces, it does not allow for specific data sets to be used for the content, and hence the need for the custom built emulator.

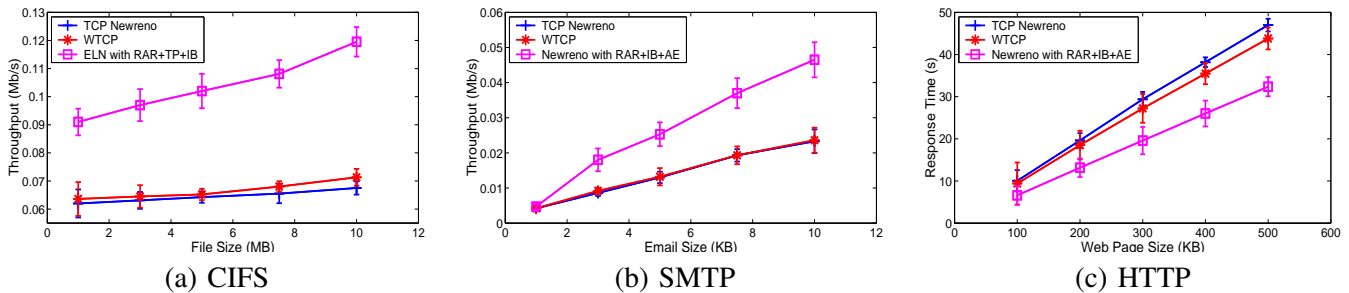


Fig. 7. Integrated  $A^3$  Results in WWAN

applications (CIFS and SMTP) when using the integrated  $A^3$  principles is higher than when any individual principle is employed in isolation, while the response time of HTTP is lower than any individual principle. The results are shown in Figure 7, with  $A^3$  delivering performance improvements of approximately 70 %, 110 %, and 30 % for CIFS, SMTP, and HTTP, respectively.

- *Proof-of-concept Prototype:* The prototype runs on a testbed consisting of five PCs connected in a linear topology. The first four PCs run Fedora Core 5 with 2.6.15 Linux kernel. The fifth PC has dual OS of both Fedora Core 5 and Windows 2000. All machines are equipped with 1 GHz CPU and 256 MB memory. The implementation utilizes NetFilter [7] Utility for Linux platform. The first PC works as the application server for the SMTP (Sendmail server), CIFS (Samba server) and SCP (SCP server). The  $A^3$  server module and client module are installed on the second and fourth PCs, respectively. The third PC works as a WAN emulator, and the fifth PC has the email client, sambaclient, SCP client and Internet Explorer running on it.

The prototype implementation makes use of two netfilter libraries: libnfnetlink (version 0.0.16) and libnetfilter\_queue (version 0.0.12) [7]. The registered hook points that we use are `NF_IP_FORWARD`. For all the hooks, the registered target is the `NF_QUEUE`, which queues packets and allows user-space processing. After appropriate processing, the queued packets will be passed, dropped, or altered by the modules.

## F. Summary

In this work, we motivate the need for application acceleration for wireless-data networks, and present the  $A^3$  solution that is application-aware, but application transparent. Using a combination of principles targeted toward tackling design problems in popular real-world applications,  $A^3$  provides significant improvements in end-user application performance.

## V. PRINCIPLES FOR ENHANCING VOICE OVER IP LIKE REAL-TIME TRAFFIC

### A. Overview

Voice over IP (VoIP) services have been significantly gaining prominence over the last few years because of a number of impressive advantages over their traditional circuit-switched counterparts including but not limited to high bandwidth efficiency, low cost, and flexibility of using various compression strategies. Simultaneously, the use of wireless networks has also grown tremendously over the past years. Wireless LAN (WLAN) solutions, armed with better physical layer technologies, now promise high data rates of more than 100 Mbps (IEEE 802.11n).

In this context, recent efforts have focused on marrying the potential benefits of VoIP and WLANs to provide wireless telephone services. A natural question that then arises is *how well does VoIP perform over WLAN environments?* The answer to this question is counter-intuitive. Even though the WLANs boast very high data rates and a typical VoIP call carries only 128 Kbps of bidirectional data (using G.711 voice codec), the number of VoIP calls supported by these networks is abysmally low. An IEEE 802.11b network for example can support only 5 VoIP calls even at the highest data rate of 11 Mbps, as we will show later. VoIP traffic in WLANs is characterized by its small frame sizes, and IEEE 802.11 MAC is notorious for very poor performance for small frame sizes. For small frames the overheads at the different layers of the network stack themselves pose a significant burden. Added to this IEEE 802.11 MAC protocol exhibits other distributed effects that further degrade the VoIP call capacity.

TABLE I  
ANALYSIS OF MNVC(PHY) IMPROVEMENT USING DIFFERENT SCHEMES.

AA		FA		LA		TS		HC	
# of ACK	MNVC	$k$ (/s)	MNVC	$R$ (Mbps)	MNVC	$T_{\text{DIFS}}$ ( $\mu\text{s}$ )	MNVC	$\sum H(L)$ (Bytes)	MNVC
1	5.1	100	5.1	11	5.1	50	5.1	74	5.1
1/2	6.1	50	9.7	5.5	4.6	10	5.3	48	5.2
1/4	6.7	25	17.4	2	3.4	0	5.4		
1/8	7.1	12.5	28.9	1	2.4				
0	7.5	6.25	43.2						

Toward the goal of improving the VoIP call capacity in wireless networks we first study the reasons behind the inferior performance of VoIP over WLANs. We identify the universal set of components that could be improved to increase VoIP call capacity over such environments. Finally, we select the three dominant components and present algorithms that improve the performance of the components, thereby increasing the VoIP call capacity. While most of the discussions in the report apply to any of the IEEE 802.11 class of WLAN environments, we present all our discussions only in the context of IEEE 802.11b for clarity.

### B. Motivation and Algorithm Design

We can write an integrated equation for the Maximum Number of VoIP Calls (MNVC) in an 802.11 network as,

$$\text{MNVC(PHY)} = \frac{1}{2k \left( T_{\text{DIFS}} + T_{\text{BO}} + T_{\text{SIFS}} + T_{\text{PHY}} + T_{\text{ACK}} + \frac{8(D + \sum_{L=\text{RTP}}^{\text{MAC}} H(L))}{R} \right)}, \quad (1)$$

where  $k$  is the frame rate of codec,  $D$  is the frame size, and  $H(L)$  is the size of header of the layer  $L$ .  $T_{\text{DIFS}}$ ,  $T_{\text{BO}}$ ,  $T_{\text{SIFS}}$ , and  $T_{\text{ACK}}$  are MAC layer overheads of DIFS, backoff time, SIFS, and ACK, respectively and  $T_{\text{PHY}}$  is the transmission time.

Analyzing this equation, we can envisage 5 different schemes to improve the call capacity by leveraging different components in the equation:

- *ACK Aggregation (AA)*: ACK aggregation refers to sending a single ACK for a block of  $n$  frames. ACK aggregation results in the *reduction of*  $T_{\text{ACK}}$ . From TABLE I we see that we can increase the call capacity by up to 2 calls.
- *Frame Aggregation (FA)*: Frame aggregation refers to fusing multiple frames destined to the same end user into a single large frame. TABLE I shows that frame aggregation gives the maximum possible improvement in terms of MNVC. However frame aggregation is associated with an increase in delay of the frames. In section V-B.2 we present an algorithm for frame aggregation that serves as a balance between the MNVC and the expected delay. FA *decreases*  $k$  in Eq. 1.
- *Link Adaptation (LA)*: Link adaptation refers to changing the transmission rate for the data frames. The 802.11b standard specifies 4 different data rates that can be used. The prevailing channel conditions affect the choice of data rate. Generally a high rate is used in conditions of low error and vice-versa. Existing rate adaptation techniques do not consider the size of frames while adapting to different rates. VoIP traffic is characterized by small frame size and hence will suffer lower frame drop rates than normal traffic for the same channel conditions. Hence there is a strong motivation for a rate adaptation technique that takes into account the frame size. LA can *control*  $R$  in Eq. 1 and maintain it optimal for a given channel condition.
- *Time Saving (TS)*: Time saving refers to *reducing*  $T_{\text{DIFS}}$  waiting time between two successive frames. From TABLE I we see that the potential benefits as a result of this technique is less than a single call. The reason for this fact is that the time interval of DIFS is very small in the order of  $50\mu\text{sec}$  and is considerably low when compared to other MAC delays like backoffs and frame transmission times.
- *Header Compression (HC)*: Header compression refers to reducing the size of the various headers like the RTP/UDP/IP headers using techniques either proposed in literature or otherwise. This technique also has limited potential in improving the call capacity in tune of only 0.1 calls. HC *reduces*  $\sum_{L=\text{RTP}}^{\text{MAC}} H(L)$  in Eq. 1.

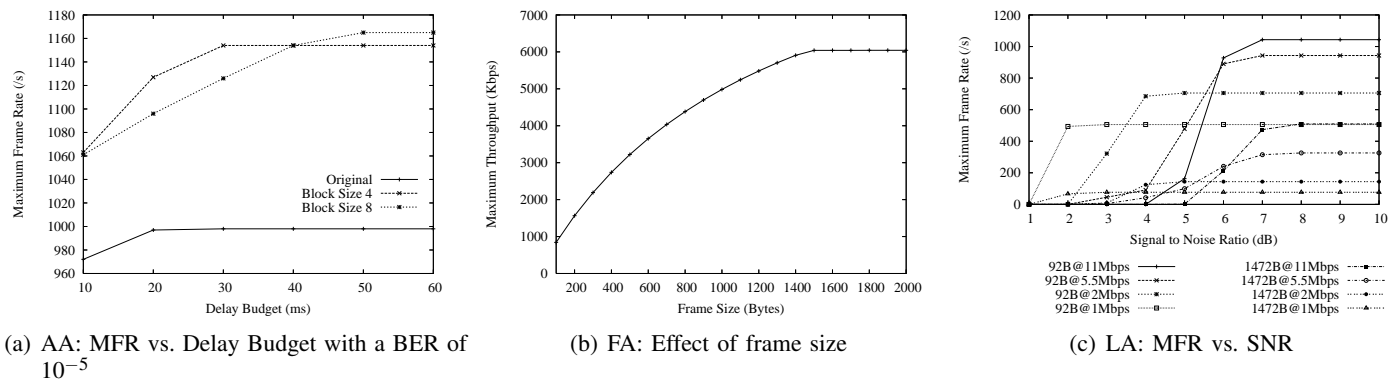


Fig. 8. Motivation results of AA/FA/LA.

In this report we consider AA, FA and LA techniques only because they have a good potential to improve call capacity, while TS and HC provide a maximum of less than one call improvement and hence we ignore them in developing new algorithms.

1) *ACK Aggregation (AA)*: Under saturated conditions the presence of ACK impacts the capacity of an 802.11 network. The 802.11 MAC specifies that the ACK should be sent at the basic rate. Hence even though the size of ACK is very small it eats up considerable amount of the capacity. Eq. 1 captures the impact of the ACK duration on the MNVC that can be supported. Hence one can envisage a scheme wherein ACKs are sent for every  $n$  frames instead of for every frame. We define the set of  $n$  frames as a block and the ACK for every block can contain information about the number of frames received correctly and the number in error. Only the lost frames can be retransmitted. TABLE I summarizes the improvements one can expect by aggregating ACKs. We observe that there is a difference between using a block size of 1 and a block size of 8 but there isn't much difference between 8 and  $\infty$ . This is because of the presence of other terms in the denominator of Eq. 1.

Fig. 8(a) shows the variation in throughput (observed maximum frame rate) by varying the block sizes. We observe that smaller block sizes result in smaller delay for the frames but lower total throughput as well. A larger block size on the other hand has larger delay for the frames but higher overall throughput. This is because when there is an error in a large block size scenario, it takes longer time to retransmit. However the better overall throughput given by a larger block size comes from the higher capacity due to smaller number of ACKs.

Even though the destination delays sending ACK to the source until all the frames belonging to one block are received, the destination can reduce the play buffer delay by sending the uncorrupted frame up to the upper layer whenever it receives a frame.

Based on the above observation that there are cross over points we can think of an algorithm that changes the block size adaptively. We simulate this scheme as a 2.5 layer solution in between the MAC layer and the interface queue. We assume that there is no preset block size and we send a block ACK request from the source once all frames in the present block are sent. Upon receiving the block ACK request, the destination responds with a block ACK containing the required information. The source then starts a new block and retransmits the required frames of the previous block and continues with newer frames. Because of the presence of a block ACK request we can change the size of block at will. We implement a simple adaptive scheme where we increase the block size upon receiving a block ACK with all successes and reduce the size on receiving a block ACK with even a single data loss. The adaptive algorithm chooses the right block size depending on the number of losses in the current block. Thus the adaptive algorithm should give a performance that is the best of both worlds i.e it should have a good delay characteristic as well as high saturated capacity.

2) *Frame Aggregation (FA)*: The MAC throughput in an 802.11 network is limited by the size of the frame. Fig. 8(b) shows the maximum throughput observed in a one-hop single flow scenario using an error-free 802.11b 11 Mbps link (with RTS/CTS mechanism suppressed). We observe that for a small frame size, of say 120 bytes

(the size of a G.711 encoded VoIP frame with RTP, UDP, and IP headers), the maximum throughput is about 931 Kbps, which means that we can get only 7.2 calls from such a network. The reason for this inferior performance is because of the various overheads involved, like the PLCP header, DIFS and SIFS intervals and random backoff. The effect of the overheads decrease as the frame size increases.

An obvious scheme to improve the call capacity will be to aggregate frames. That said, it is a non trivial problem as to how one should go about aggregating frames. A direct consequence of aggregating frames is increased delay. However, there is a delay budget that can be accommodated by the voice codecs and it varies between codecs. Towards this goal of achieving a balance between the number of frames that can be aggregated and the delay budget, we propose an intelligent scheme which scales the number of voice calls that can be accommodated as the delay budget increases. With a delay budget of 100 ms we can get as many as 25 calls at the same time. We also show that such a scheme can also perform very well in the presence of other background traffic.

The core idea behind the algorithm is that we should aggregate frames only when required. Specifically, we aggregate only those frames that have already been received by the underlying queue that precedes the MAC layer in the network stack. The intuition behind such a scheme was the observation that a number of frames were being dropped at the AP because of the queue getting filled up when the number of calls increased beyond 5. Frames are aggregated just before transmitting on the physical layer after all the waiting for DIFS, SIFS and/or backoff times at the MAC layer. We limit the maximum size of the frame that is transmitted to 2304 bytes (maximum allowed frame size in 802.11 standard).

In a purely downstream environment (only AP to client communication) the number of flows that can be accommodated increases as the delay budget allowed increases. This saturates beyond a point because of the maximum frame size limitation. In the upstream case however we do not observe a phenomenal increase in capacity. This can be explained as follows. Each client has only one flow (100 frames per second in our case) and it competes with other nodes for a transmission slot. By the time a client gets a slot to transmit it has typically only one or two frames to aggregate. The AP on the other hand has a number of flows but still contends as one node in network. Hence there will be a number of frames for each flow when it gets to transmit.

To correct this imbalance between downstream and upstream flows we propose an enhancement to the piggy-backing algorithm. We make modifications at both AP and client sides. At the AP we increase the buffer size of the queue to hold up to 500 frames. At the client side we maintain a variable that holds the number of frames aggregated in the previously received frame from the AP. When the client gets a chance to transmit it checks if it has frames at least equal to that variable. If yes, it goes ahead to transmit the aggregated frame. However if it does not have enough frames it goes into a random backoff without increasing the contention window size.

We do not increase the contention window size because we do not want to wait for a very long time because of the delay budget. Since each client now waits for the same number of frames as in downstream it will contend for the channel at large time intervals. The AP on the other hand will contend for each of its client one after the other. Thus the contentions are now balanced between the upstream and downstream nodes depending on the amount of traffic. We refer to this scheme as *enhanced piggybacking*. We study the behavior of this scheme using *ns2* simulations and the results are discussed in the next section.

3) *Link Adaptation (LA)*: It is intuitive that the MNVC at a higher rate would be larger than at a lower rate. However this is true only when we assume that the link is error-free. When the signal-to-noise ratio (SNR) of the channel is low, it would be advisable to transmit at a lower rate. This is because for a given SNR the loss for a lower rate transmission is lower than at a higher rate.

Several schemes have been proposed in literature and some like the ARF [21] have been implemented in practice to adapt the transmission rate depending on the prevailing channel conditions. But all these schemes either adapt to the number of frames in error or use SNR measured at the receiver and communicated to the sender via CTS frame. These schemes do not consider the data frame size for making the decision on the data rate.

There is an inverse relation between SNR and bit error rate (BER) which directly translates to frame error rate (FER) depending on the frame size. Hence for a given BER the FER of a small frame might be acceptable whereas the FER for a large frame size might be unacceptable. Fig. 8(c) illustrates this fact. We observe that for an SNR of 6 dB, a data rate of 11 Mbps might be acceptable for small frame size but unacceptable for large frames. This

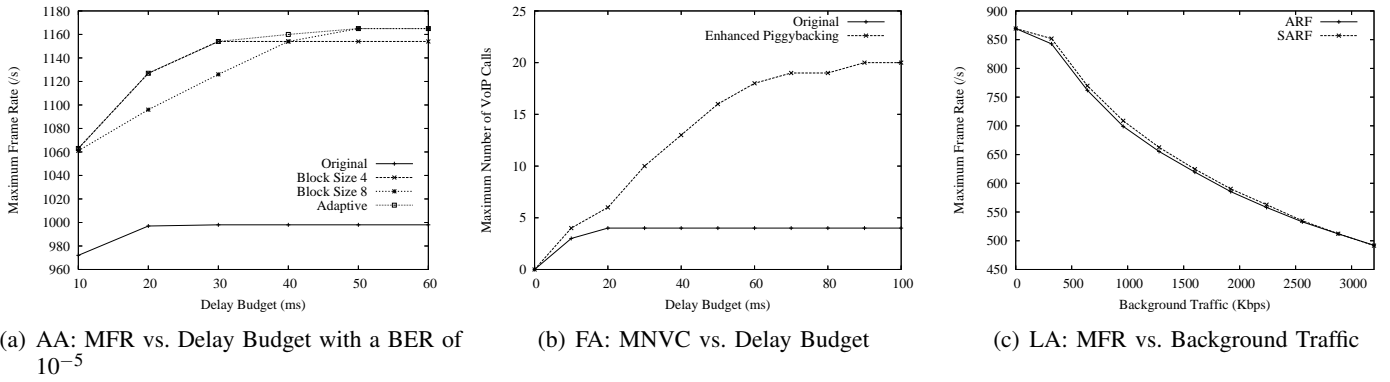


Fig. 9. Simulation results of AA/FA/LA.

gives us a motivation for a size aware rate adaptation technique.

ARF is a popular rate adaptation technique that is widely deployed in many existing WLANs. It is a simple but robust technique that chooses the rate depending on the reception or failure of the successive ACKs for the DATA frames. Briefly the ARF algorithm works as follows: For every ten consecutive ACKs correctly received the rate is increased to the next higher allowed value and upon two consecutive ACK failures the rate is reduced to the immediate lower allowed rate. As is evident from the algorithm this technique does not take into account the frame size during rate adaptation. It is a known fact that ARF does not perform very well under rapidly changing channel conditions because it depends on ten successful ACKs to increase the rate. This problem of ARF worsens when the frames are of varying sizes. VoIP frames are characterized by small frame sizes and when they are interspersed with large frame sized background traffic the ARF will suffer in the sense that frames will be sent at wrong rates.

The core idea behind Size-aware Auto Rate Fallback (SARF) is that if a small frame is in error then there is a high probability of error for a large frame as well. Similarly when a large frame is successful, there is a very high probability of success for small frames as well.

The source maintains two counters (success counter and failure counter) for different frame sizes. These counters are updated (based on the previous conjecture) upon reception or failure of ACKs. When these individual counters reach preset threshold values the rate for the corresponding frame size is modified (either increased or decreased).

The adaptive algorithm is similar to ARF i.e upon successive ACKs received correctly the rate is increased to the next level and upon consecutive ACK failures the rate is reduced. The threshold values are chosen based on some heuristic methods. We observe that even though the MNVC does not improve drastically for SARF the number of frames sent at wrong rate is significantly less than that of ARF. The packets sent at wrong rate may incur either large delay due to lower rate than ideal or much loss due to higher rate than ideal.

### C. Performance Evaluation

We use the open source *ns2* simulator for all our simulations. Unless otherwise stated bi-directional CBR traffic with a frame size of 92 bytes is used for modeling the VoIP traffic. The frame rate of traffic is set at 100 frames per second in either direction i.e AP to client and client to AP. Each call originates from a wired node and has two wired links and one final 802.11b wireless link in an infrastructure mode. In scenarios where background traffic is required we use another CBR traffic with a frame size of 1472 bytes. The rate of the background traffic is varied according to the need.

1) *ACK Aggregation (AA)*: To study the performance of the ACK aggregation scheme we setup bi-directional CBR flows between wired node and client node in the wireless domain. We use MFR as a fine grained metric to study the improvement of this scheme.

Fig. 9(a) shows the performance results of the ACK aggregation algorithm at different BERs. The capacity improvement due to the aggregation of ACKs is evident from the saturated frame rate at higher delay budget. The saturation throughput is higher for a larger block size. This is because for a larger block size smaller number of

ACKs are sent thus making more room for data frames. However when a frame gets lost it will take longer time to get retransmitted if the block size is large. This delay gets worse when the BER is higher. The adaptive algorithm chooses the right block size depending on the number of losses in the current block. Thus the adaptive algorithm gives performance that is the best of both worlds. It has both better delay characteristics as well as better saturated frame rate.

2) *Frame Aggregation (FA)*: To simulate the number of VoIP calls using the enhanced piggybacking scheme we assume an erroneous environment with a BER of  $10^{-5}$ . We setup a number of bi-directional CBR flows with a rate of 100 frames per second in either direction. To study the number of VoIP calls for a given delay budget we increase the number of such bi-directional flows to different clients and stop when the loss at the given delay is more than 2% of the frames. Losses occur due to both erroneous link (BER) and frames exceeding the delay budget.

Fig. 9(b) shows the MNVC that can be accommodated for a given delay budget. We compare the enhanced piggybacking scheme with the original data-ACK scheme. The original scheme peaks at a maximum of 4 calls, which is one smaller than analysis result due to error condition and delay budget. However the enhanced piggybacking scheme gives up to 20 calls for a delay budget of 100 ms and 18 calls for a delay budget of 60 ms. Note that 60 ms is an acceptable wireless delay for good quality of VoIP.

The enhanced piggybacking scheme is able to scale with increasing delay budget. One drawback of the enhanced piggybacking scheme is that as the number of calls increase the delay for every user increases. A suitable scheme at the AP should be employed to limit the number of calls based on a given delay budget. A possible QoS policy can be applied if we can measure the delay for each frame for the leg of transmission from AP to the far end client. This delay information can be used to set the delay budget of the wireless leg of the call. This will be part of our future work.

3) *Link Adaptation (LA)*: We use a single VoIP call and a single background CBR source with large frame size (1472 bytes) to study the efficacy of SARF. We compare the performance of SARF with ARF. We assume a hidden Markov channel model [15] to simulate varying channel SNR conditions.

Fig. 9(c) shows the MNVC for ARF and SARF for varying background traffic. We observe that both SARF and ARF perform similarly. This is because both ARF and SARF are based on similar principles of tracking the channel condition based on success or failure of ACKs. SARF also performs well for other QoS metrics like delay budget but we do not present those details due to lack of space.

#### D. Summary and Future Work

In this report, we analyze the reasons behind the inferior performance of VoIP traffic over wireless LANs. We setup an experimental testbed to serve as a testbed baseline and to justify our analysis. While doing so we propose three algorithms at the MAC layer to improve the observed call capacity namely ACK Aggregation (AA), Frame Aggregation (FA), and Link Adaptation (LA). Extensive ns-2 simulations are performed to show the efficacy of these algorithms. We find that FA can provide capacity improvements in the order of up to 300%. As part of our future work we will complete the analysis and simulation for a variety of codecs and IEEE 802.11 variants. We also plan to implement both software only and standard compliant solution of the proposed algorithms in the real testbed.

We will publish the software-only VoIP acceleration concept in IEEE International Conference on Broadband Communications, Networks, and Systems [20].

## VI. PRINCIPLES FOR ENHANCING WEB ACCESS

### A. Overview

In the past couple of decades, tremendous amount of research has been done on improving web access performance over Internet. In this report, we study the performance of a web-browser under low-bandwidth network conditions, such as in wireless networks. Specifically, we analyze the characteristics of a web-browser with the objective of identifying reasons as to why they might suffer in low-bandwidth conditions. We find that the current fetching model employed by commercial web-browsers is not optimal in bandwidth challenged environments.

We show that the absence of content prioritization and intelligent object fetching mechanisms in current web-browsers leads to increased response times. Browsers, today, do not prioritize the useful data that is viewed by the user over other data in the web-page. With a greedy fetch of the entire content of a web-page, precious bandwidth is wasted and in turn increases user perceived response time. Further, without intelligent object fetching mechanism, the download process of current web-browsers does not utilize the network bandwidth efficiently.

To make this problem even worse, many web-pages have become larger both in pixel- and byte-size with a large number of external objects. For example, `cnn.com` has a main page, which is larger than 3 times the pixel-size of *client area* that is defined as content area within a main browser window, and greater than 300-KB data, with hundreds of external objects. Thus, even with a network with bottleneck bandwidth being 100 Kbps, the time taken to fetch the entire page can be larger than 20 seconds.

In this report, we propose a client-side only solution to address the problem of large response time with current browsers. The solution is based on careful consideration of several factors including the content displayed on the screen viewed by the user, server-side content distribution networks, and the relationship between the HTTP and TCP protocols. The three mechanisms we propose include prioritized fetching, object reordering and connection management. One major advantage of our approach is that it is pure client-side enhancement and does not need any server changes. The proposed solution helps to reduce response time, and is easy to deploy since it only requires client-side installation to current web-browsers.

## B. Motivation

In this section, we describe drawbacks in the conventional web access model in low-bandwidth environments and use them as motivation for designing a new web access scheme.

1) *Screen Contention Problem*: When a user is viewing a screen on a display device, objects for displaying other screens are unnecessary in the sense that they are not visible to the user at this time. However, in conventional web browsers, the process of fetching *necessary* on-screen objects (*i.e.* objects on current screen) may be slowed down due to the competition from the process of fetching *unnecessary* off-screen objects. We refer to the fact that objects from different screens are competing for bandwidth as *screen contention*.

The main reason of screen contention is disparity of cumulative transfer-size among multiple connections. As mentioned earlier, a parsing engine inserts object requests to message queues in a round-robin fashion that considers only fairness in terms of the number of objects per connections. As a result, some connections having only small-size objects may finish transmissions of on-screen objects early and begin to fetch off-screen objects. Under this scenario, different connections may fetch objects on different screens simultaneously.

Another possible reason is directionality in a table structure. When a table is defined in a web page so that one of internal cells has larger vertical pixel-size than the client area in the browser window, a web browser fetches off-screen objects located at the end of this cell first, and then begins to fetch on-screen objects in the next cells. Figure 10 shows an example of the object fetching sequence in Internet Explorer at the `amazon.com` page, which consists of 1 HTML document, 5 javascript, 2 flash, and 65 image objects. In the figure, most off-screen objects are fetched or begin to be fetched before all the 35 on-screen objects are downloaded because of fetching directionality in the table.

We show the effect of the screen contention problem by presenting the simulated object fetching progress in Figure 11. We assume that all the objects are from a single server, and the effect of directionality in a table structure is ignored. In the simulations, the initial screen has 18 objects, *i.e.* objects numbered after 18 are unnecessary data. As observed from the figure, objects from both the current screen and other screens are fetched simultaneously due to the screen contention problem. Since fetching the unnecessary objects consumes some portion of bandwidth, the resulting response time for initial screen is increased unnecessarily. As shown in the figure, image (IMG) objects numbered 20 and 22 are fetched in parallel with other objects on the initial screen. As a result, the response time of initial screen, which is measured after IMG 17 is downloaded, is 18.7 s.

An intuitive solution to screen contention is to prevent unnecessary object fetching. Figure 12 shows an ideal case that contention is eliminated in an ideal browser. As seen from the figure, when the faster connection (Connection 1) completes the downloading of all the objects on the initial screen, it stops fetching and waits for the other connection

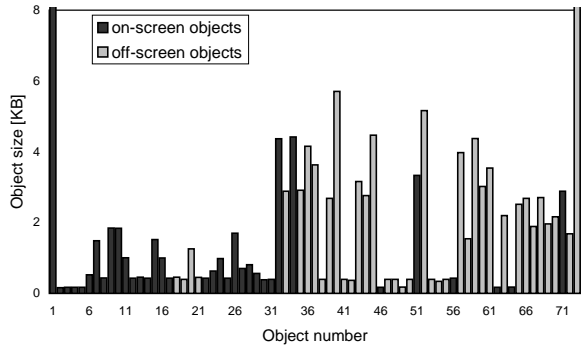


Fig. 10. Object fetching sequence at amazon.com

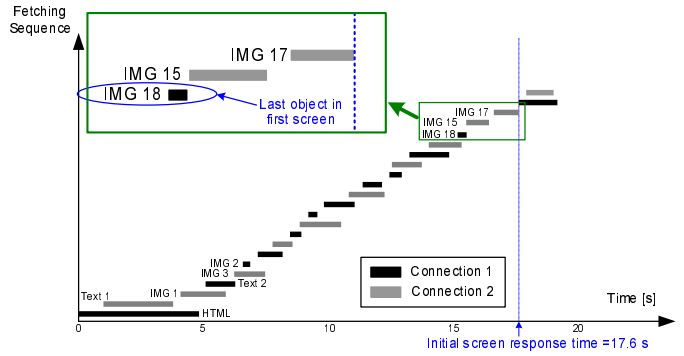


Fig. 11. Fetching without screen contention in ideal web-browsers

(Connection 2) to finish fetching the objects on the initial screen. By doing so, the remaining connection can obtain more bandwidth and in turn reduces the response time for the initial screen by 1.1 s.

2) *Bandwidth Under-utilization Problem:* In HTTP/1.1, a persistent connection consists of a series of request-response transactions. Given this model, [12] shows that the idle time of the network decreases with the increase in the number of simultaneous TCP connections. The authors of [12] also show that there is an optimal number of simultaneous TCP connections (around 6) at which the performance is optimal because of reduced idle time. However, since current web-browsers do not schedule object transfers in a bandwidth-efficient way across multiple TCP connections, current web transfers do not always maintain the optimal number of simultaneous TCP connections. In many cases, only a small number (e.g. 1 or 2) of connections are active at any instant. This results in under-utilization of the access link which we refer to as the *bandwidth under-utilization* problem.

The above-described bandwidth under-utilization problem results in varying levels of performance degradation depending on specific server scenarios. In the case of a single server, bandwidth efficiency is determined by synchronized ending times of transmission among parallel connections. In Figure 12, the last object, IMG 17, is fetched with no other objects, and thus only a single connection uses network bandwidth toward the end. In cases of multiple servers, the user performance is also affected by synchronized ending among connections to different servers.

The solution to the bandwidth under-utilization problem is to schedule the GET requests across the multiple TCP connections such that all the TCP connections are always active during the fetching process. Figure 13 shows the impact of performing ideal scheduling such that there is a pending request in each of the TCP connection. In the figure, when the faster connection finishes fetching all the objects in its request queue and has no more objects to fetch, it takes over the unfulfilled object requests from the queue of the other connection and perform fetching. As a result, both the connections can use bandwidth more efficiently and improve the initial screen response time by 1.6 s when compared to conventional web-browsers.

Significant amount of web content is being served from distributed web-servers. Ease of content update and server load-balancing are some of the benefits of employing multiple web-servers to serve web content. In the scenario of multiple web-servers, different objects belonging to the same web-page are delivered by opening TCP connections to the different servers. Commercial web-browsers do not take into account the size of the objects in scheduling the different object requests. This invariably leads to scenarios where several TCP connections to different web-servers are idle while the other connections are active. The intuitive solution is to schedule the different object requests across the multiple servers such that all the connections are active.

We have identified two issues with conventional web-browsers in bandwidth-limited networks. We observe that contention among objects belonging to different screens within the same web-page can increase user-perceived response time of the initial screen. We also identify that network bandwidth can be under-utilized because of two issues with conventional browsers. observe that in most cases screen contention and bandwidth under-utilization problems affect user performance negatively in a conventional web model, and show how the ideal browser can overcome these problems and achieve significant performance improvement. Based on these observations, in the

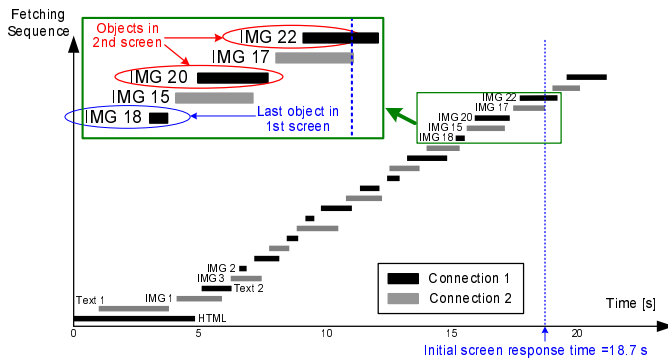


Fig. 12. Fetching with screen contention in conventional web-browsers

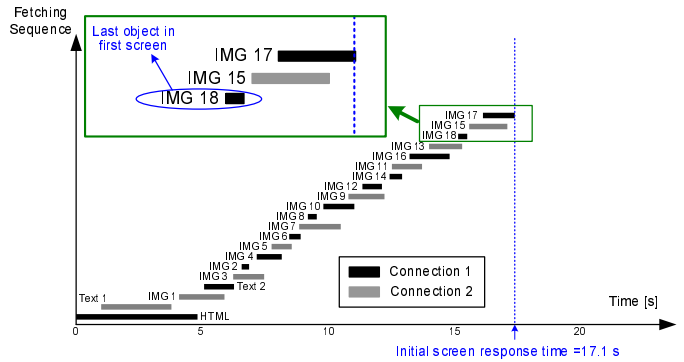


Fig. 13. Fetching with synchronized ending time in ideal web-browsers

next section, we propose a new web access scheme.

### C. Approach

Our proposed solution includes three mechanisms, namely prioritized fetching, objects reordering, and connection management. These three mechanisms complement each other as well as performing optimization with different levels of granularity for web object fetching on the web-browser side. One of the advantages of our solution is easy deployment, since it requires only client-side modification. In fact, the solution can be implemented as nothing more than an add-on to the current web-browsers.

1) *Prioritized Fetching (PF)*: PF differentiates objects from different screens based on the current screen view and allows for downloading only the objects that are required to render the current screen display. As a result, it reduces the response time experienced by web users. The basic operation of PF is as following: (1)When a web access is performed, PF first obtains the initial screen view information in the entire document layout and prioritizes embedded objects according to their locations in the document layout. (2)Then, it performs fetching objects according to their priority levels. (3)When a user scrolls to move to a different view, PF performs the above-mentioned process again for the new screen. PF consists of the following three components.

- *Initial Object Prioritization*: Generally, a web-page consists of various types of embedded objects. PF considers textbased files including HTML, javascript, cascading style sheets, and other layout-related files, as the highest-priority objects since these objects play an important role to construct the overall HTML display layout. On the other hand, for other types objects such as image and multimedia objects, PF assigns different priority levels according to their locations. For simplicity, we consider only IMG objects as representatives of objects that do not affect the document layout.

As mentioned earlier, PF performs location-based prioritization for IMG objects. The detection of pixel-location of an IMG object is possible because most HTML document files defines the pixel-size of image objects and a web-browser can construct the full page layout without downloading these objects. In cases that the HTML document does not specify the pixel-size of an image object that is not fetched yet, PF uses an pre-obtained averaged value based on browsing history.

In order to get the location information of objects, PF scans the document object model (DOM) tree [9]. When it finds an IMG object definition, it searches all the successors in the tree and calculates location offsets from successors to predecessors in a recursive fashion until it reaches the top of the tree. The absolute location in the layout is defined as the sum of all the relative offset. Based on this location information, PF gives highest priority level to objects that are located within the current view in the client area and low priority levels to others.

- *Selective Object Fetching*: For the schemes used to fetch objects of different priority levels, many existing schemes allocate a small portion of bandwidth for low-priority transmissions. However, these schemes cannot be efficiently exploited in PF due to the following reasons: (1) HTTP 1.1 defines persistent connections, which

allows transfer of multiple objects using a single TCP connection. In PF, a priority-based connection doesn't have flexibility to send both high- and low-priority objects. (2) Each TCP connection performs multiple short transmissions. As mentioned earlier, generally IMG objects that account for the half of total bytesize of a web-page have a small average byte-size (a few packets). In these short bursty on-off transmissions, prioritybased connection schemes can't assign a desired portions of bandwidth accurately.

PF uses a delayed-transmission scheme. When information of new objects are extracted from a HTML document and they are prioritized as high level, PF inserts the corresponding request messages into the already-in-use queues. In this scheme, low-priority objects are fetched only after all the higher-priority objects have been downloaded, i.e. after the higher-priority queues become empty.

- *Re-prioritization*: When browsing web-pages, a user may scroll to another view other than the current one before all the on-screen objects in the current screen are fully downloaded. For example, a user may perform *fast scroll* by searching and clicking an internal link to another part in the same page. In these scenarios, the current focus is changed before the downloading of previous screen, and the initial prioritization may not perform efficiently. Thus, a proper mechanism is required to deal with these scenarios.

When the screen focus is moved to a new area, PF removes all the IMG objects that reside in request queues and re-prioritizes them for the newly focused area. For the objects that are currently being downloaded, PF waits for their completeness. The reason for allowing this off-screen fetching is that most web-browsers, as applications above transport layer following HTTP standards, do not have mechanisms to manage disconnections and re-connections. PF thus keeps the currently incoming transfer and only updates the priority levels of the queues involved.

The provided fetching schemes can be different transport protocols, different parameters in the same protocol, or different starting time. As mentioned earlier, in this work, we consider only adjusting starting time to fetch different screens using the currently existing transport protocols.

2) *Objects Reordering (OR)*: For parallel connections to a single server, OR uses balanced ordering of objects to gain benefits in terms of reduced response time. The operations of OR consist of three steps. (1) At the first step, an initial assignment of objects is executed. (2) Then, an optimized ordering of objects is performed by TCP-aware object reordering. (3) After that, it performs dynamic objects rescheduling until all objects are completely fetched. OR consists of the following three components.

- *Initial Objects Assignment*: As we identify in Section II, conventional browsers perform a round-robin assignment to distribute object requests to multiple connections. This size-unaware assignment may cause unsynchronized ending time among different connections, and as a result increases response time. Therefore, OR performs load balancing among connections using byte-based metric rather than simple round robin in order to synchronize their ending time.

Since larger byte-size translates to longer downloading time in web fetching, OR synchronizes ending time of different connections by distributing same amount of objects to every connection. A more accurate way to synchronize the ending time could be one that also considers the number of objects, the preprocessing time for each object, and others.

Performing OR requires the byte-size information of objects. Since this information is normally not included in HTML documents, OR estimates it by considering both the object's pixel-size included in HTML documents and the object formats such as gif and jpeg. Based on this data size information, OR sends object requests through multiple connections in a balanced way. A time with  $\phi$  expiration value is used to strike the balance between amount of objects and increased response time.

- *Dynamic Objects Rescheduling*: Irrespective of initially balanced assignment of objects among connections, due to dynamic behavior of connections, the total fetching time of different connections may still vary significantly. If due to some reasons one connection is delayed, and the other connection is idle, it is possible to reschedule the objects from the busy connection to the idle one, and thus reduce response time even more. Dynamic Objects Rescheduling runs in an on-demand fashion during the fetching process in order to deal with the dynamic nature of the connections.
- *TCP-aware Objects Reordering*: When initial objects are assigned to connections, TCP-aware reordering of

fetching sequence can minimize the adverse effect of slow-start in TCP connection setup and increase fetching speed.

TCP slow start can kick in at any time during the downloading process. However, since HTTP and upper layer is unaware of each others status information, there is no way to take advantage of them without some other cross-layer mechanisms. Thus, the TCP-aware objects reordering scheme only makes use of the slow start phase in the beginning of a TCP connection. For this rescheduling to take place, appropriate ordering of objects is required. Intuitively, with small objects being put at the end of connection, it is more likely to reschedule objects among connections, and thus reduce response time. Also, ordering objects from big to small also makes rescheduling easily to perform, since small objects can be rescheduled in a finer granularity as the fetching process going on.

OR orders the fetching sequence in a *rats-elephants-rats* fashion. First, all the objects assigned to one connection are sorted according to their data size. After that, from smallest one, all objects are inserted from two ends of the queue in round-robin way, and the resulting ordering is a small-to-big-to-small order.

3) *Connection Management (CM)*: CM addresses the bandwidth under-utilization problem when fetching objects from multiple servers by controlling the numbers of connections a browser can open to different servers. By adjusting the number of connections for each server, CM effectively synchronizes the ending time of downloads in the connections. As a result of the improved bandwidth utilization, the response time is reduced. CM consists of the following two components.

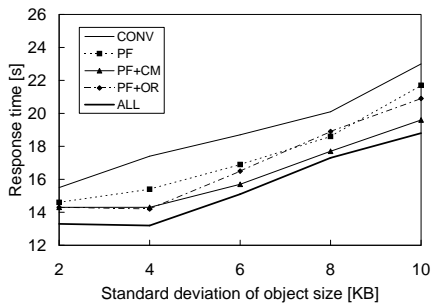
- *Per-Connection Load Estimation*: In order to estimate the ending time of downloading, CM uses the byte-size information that OR converted earlier. The intuition of CM is to assign more connections to servers with larger data size, while assign less connections to servers with smaller data size. To maintain friendliness to current browsers and compatibility to published standards, the total number of connections in our mechanism is maintained the same as in today's popular browsers. By doing so, CM behaves *friendly* to them. To achieve this purpose, whenever it assigns one more connection to some server, one less connection should be deducted from some other server. Furthermore, CM limits the maximum number of connections assigned to a server.
- *Dynamic Connection Assignment*: When fetching a HTML document from a server, a browser fetches and parses the contents. Whenever OR detects new object information, it estimates the byte-size of this object and starts a timer with  $\delta$  expiration value. The setting of this timer requires careful consideration. On one hand, CM needs to collect some amount of object samples in order to achieve improvements. Thereby the  $\delta$  should not be too small. On the other hand, CM should not delay object fetching significantly to avoid increasing response time adversely, and thus the expiration value should not be very large such as dozens of ms. After the expiration of this timer, CM performs the initial assignment based on object information collected so far. During the process of fetching objects, it keeps recording the object information on how much data already received. This information will be used again to adjust the number of connections

#### D. Key Results

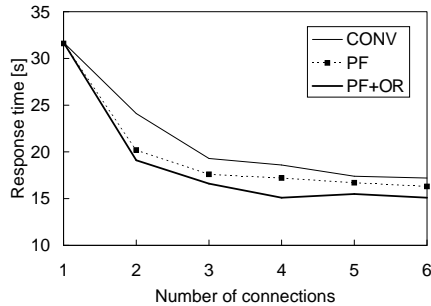
In order to evaluate the performances, we use *ns2* simulator [26]. We assume that the local DNS server has all the required domain information. Response time for the initial screen is used as the primary metric for comparing performances. In this section, we compare five schemes including conventional (CONV), PF only (PF), PF with OR (PF+OR), PF with CM (PF+CM), and all integrated (ALL) schemes.

1) *Impact of Object Characteristics*: Figure 14 shows the initial response time of conventional web-browsers, and our proposed solution when the standard deviation of individual object size and the number of objects in the first screen vary. As shown in the figure, when used in combination, the three proposed mechanisms can reduce up to 30% of response time compared to current browsers.

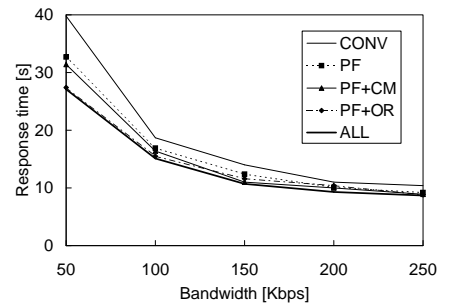
Figure 14(a) show that as the variance of object size increase, the performance of both the conventional model and our scheme shows worse performance. For conventional web-browsers, the reason is obvious, since larger variance can be translated to reduced bandwidth utilization. Our mechanisms can alleviate this problem, and thus reduce the initial response time. However, since the problem still exists, and becomes more severe when variance of object size increases, the performance degradation is still expected.



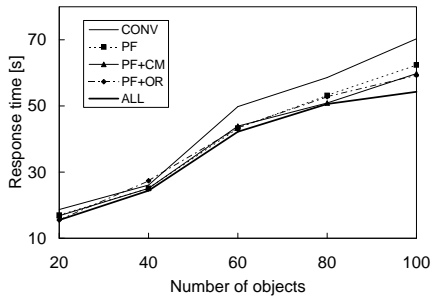
(a) Impact of variance of object size



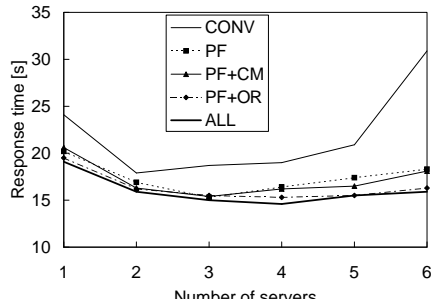
(a) Impact of number of connections



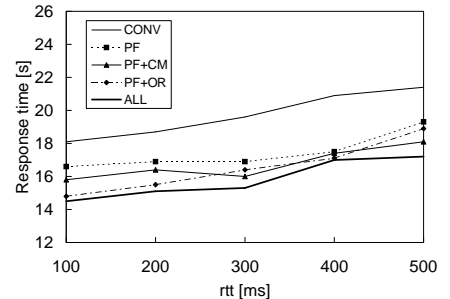
(a) Impact of bandwidth



(b) Impact of number of objects



(b) Impact of number of servers



(b) Impact of round-trip time

Fig. 14. Impact of object characteristics

Fig. 15. Impact of numbers of connections and servers

Fig. 16. Impact of network characteristics

Figure 14(b) shows the performance differences between conventional web-browsers and proposed ones when the total number of objects increases. Two trends are shown in the figure. As more objects are included in a web-page, first, larger response time is expected; second, the response time reduced by the proposed solution is larger since all of the three mechanisms can gain more benefits.

2) *Impact of Number of Connections and Servers*: Figure 15(a) shows the impact of number of connections to a single server, and it can be seen that up to 20% of response time can be reduced by using our solution. In the figure, as the number of connections to a single server increases, both the conventional and our solution has smaller response time. However, as this number exceeds 4, there are no obvious performance improvements with more connections. This result is consistent with the results presented in other works [12].

Figure 15(b) shows how the initial response time varies as the number of servers for a web-page increases. Two observations can be made from the figure. Increasing number of servers does not necessarily always result in better performance for both conventional browsers and proposed ones. Second, with more servers, our solution can achieve more improvements compared to conventional web-browsers.

3) *Impact of Network Characteristics*: Figure 16(a) shows how the initial response time changes under varying bottleneck bandwidth. As shown in the figure, our solution brings more performance improvement for smaller bandwidth. It is because of the fact that smaller bandwidth makes the screen contention problem more severe, and thus our solution can reduce response time more by alleviating this problem.

Figure 16(b) shows how the initial response time is affected by the rtt values. Since the major effects of rtt come from the request-response behavior of HTTP protocols (i.e. each object is fetched upon the request from the web client, and thus takes at least one rtt to fetch one object) and our solution can alleviate this effect by removing some of these rtt's required, our solution sees better performance. As shown in the figure, around 20% performance improvement is achieved by our solution under the rtt values considered in the evaluation.

## E. Summary

In this report, we first explore the reasons why conventional web access models are not appropriate for low-bandwidth hosts. We identify two reasons, screen contentions and bandwidth under-utilization, which results in large user-perceived response time. To address this problem, we propose a new web access scheme for low-bandwidth hosts such as wireless clients. The proposed scheme uses an intelligent mix of prioritized fetching, object reordering, and connection management. Using simulations with the web parameters obtained from the top web sites, we evaluate the performance of our scheme and prove its benefits over conventional web access models.

We will publish the client-side web acceleration concept in IEEE International Conference on Broadband Communications, Networks, and Systems [14].

## VII. PRINCIPLES FOR ENHANCING P2P TRAFFIC

### A. Overview

Over the last few years, peer-to-peer (P2P) data sharing applications have experienced an explosive growth. By the end of 2005, a staggering 60% of the Internet data traffic constituted of P2P file sharing [2]. While copyright concerns had earlier brought down popular P2P applications such as Napster, several content owners and providers have of late started embracing what is being seen as a technology that has come to stay [1].

With P2P data sharing applications securing a dominant position in the Internet landscape, an interesting question is to ask is: *what is the performance of mobile users when participating in P2P data sharing?* The question is significant because of two reasons: (a) as with any Internet application with emerging or established popularity, wireless and mobile users are increasingly adopting P2P data sharing applications on devices such as laptops and PDAs [17]; And (b) there are several efforts underway to deliver P2P data sharing as the next killer application for mobile devices [23], [24]. Initial instantiations of such efforts focus on sharing of ring-tones and music files, but are expected to evolve into other types of content including video.

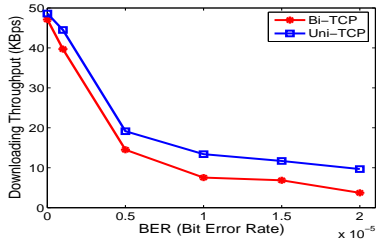
Thus, in this work, we first investigate the following question: *what is the performance of a mobile user in a wireless environment using a P2P data sharing application?* As a corollary, we also investigate the following question: *what is the performance of a fixed peer in a P2P network when using a mobile host as a corresponding peer?* In answering the above questions, we find that several of the fundamental design principles and peculiarities used in P2P data sharing applications are inconsistent with the key limiting characteristics of typical wireless and mobile environments.

Using insights gained through the aforementioned study, we present a deployable solution suite called wireless P2P (*wP2P*) that addresses the issues using changes only to the P2P application at the mobile host. *wP2P* uses techniques transparent to the fixed peer, but uniquely relevant to the specific issues pertaining to wireless and mobile hosts functioning in a P2P data network. While we elaborate on the specifics of the solutions later in the paper, *wP2P* uses a combination of multiple proposed techniques in tandem including age based manipulation, incentive aware operations, and mobility aware fetches. We evaluate *wP2P* with an implemented prototype, and show that significant performance improvements can be achieved for mobile hosts and fixed peers when using *wP2P* at the mobile hosts.

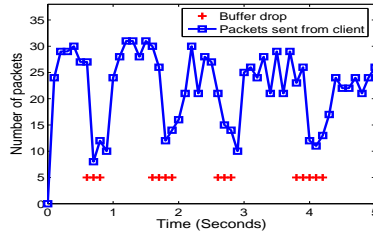
### B. Motivation

1) *Scope:* This work is entirely focused on P2P data sharing networks. Data sharing P2P networks are primarily used for sharing files containing audio (e.g. mp3 files), video (e.g. mpeg2 files), or data (e.g. linux distributions). While we identify characteristics of P2P networks that are generically applicable to all four of the above networks, we use BitTorrent as the primary example for all discussions, experiments, and trials. However, as necessary we also step back and investigate relevance of our discussions and interpretations for the other networks as well. We believe that this choice of BitTorrent as the key representative is justifiable from multiple standpoints including its dominance in terms of traffic carried, and its relative sophistication.

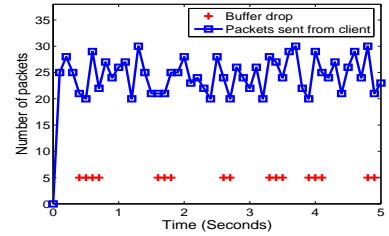
While mobile users with any type of wireless access can participate in P2P networks, the access technology typically used is wireless LANs (WLANs). This is both because of the higher bandwidths available and the relatively lower or no cost models associated with such networks. Hence, we consider WLANs for the wireless environment



(a) Throughput comparison



(b) Uni-directional: Packets



(c) Bi-directional: Packets

Fig. 17. Impact of Bi-Directional TCP

in this work. Similarly, while other mobile devices such as PDAs and IP enabled cellphones are fair game for assuming membership in P2P networks, we primarily consider laptops as the mobile device in this work.

We consider throughput performance as the main metric in our evaluation and optimization considerations. The focus is more on the question: *what is the performance of a mobile host when it participates in a P2P network?* In addition, we also consider a corollary question: *what is the performance of a fixed peer when it uses a mobile host as a peer to download data from?*

2) *Backgrounds:* BitTorrent, like other P2P data sharing protocols, uses peers that have downloaded a certain content as the sources for the content subsequently for other peers that need the same content. We now outline some of the key elements of the BitTorrent protocol relevant to the focus of this work.

- *Torrent, tracker, seeds and leeches:* (i) Any peer that wants to share a file through the BitTorrent network creates a “.torrent” file that consists of some meta data information and the address of the *tracker* that will act as the directory server for the file. (ii) A tracker maintains all current peers that have a specific file either in its entirety or in parts. When it receives a request from a client for a specific file, it furnishes the client with the addresses of the peers associated with the file. The list of peer addresses is updated periodically. (iii) All peers downloading a specific file form a swarm, which consists of seeds (peers having the entire file) and leeches (only in parts).

- *Tit-for-Tat and Rarest-first:* A client uploads to peers in the swarm and downloads from other peers. (i) A tit-for-tat incentive policy is used to control the upload rate of one peer to another peer based on the download rate the peer enjoys from that other peer. (ii) BitTorrent peers do not fetch parts of the file in sequence. Instead, each peer picks the *rarest* of the blocks (in terms of the number of peers in the swarm that have the block) preferably to download. This ensures that the rarest blocks of a file are propagated in the swarm faster, reducing bottlenecks at the few peers that have the block and increasing the availability of those blocks if the peers that have them shutdown.

### C. Test bed & Methodology

We perform experiments on a BitTorrent network to study the performance of a mobile host. We identify several design characteristics of P2P data networks that are incompatible with typical characteristics of a wireless environment. While we present all the discussions in the context of BitTorrent, we revisit the implications of the discussions on the other P2P networks at the end of the section. The testbed consists of six locally controlled BitTorrent peers which run popular BitTorrent clients: three run the Azureus client 2.3.0.4 on Linux, and the other three peers run the BitSpirit 3.2.215 client on Windows.

1) *Bi-directional TCP:* Peers in BitTorrent upload and download at the same time, and because of the tit-for-tat policy used by BitTorrent, several of the uploads are to peers from which downloads are being done. Hence, *it is common for data to be exchanged simultaneously between peers in both directions.* Given that BitTorrent (or for that matter the other P2P applications) uses TCP, and that TCP is inherently designed to be a bi-directional protocol, BitTorrent uses TCP in its true bi-directional mode. In other words, a single TCP connection is used to transfer data in both directions between the peers.

While TCP is designed to be a bi-directional transport protocol, few extensive studies of its behavior have been performed. More importantly, in the context of this work, very little is understood about the behavior of bi-directional TCP in a wireless environment. In this context, we identify two issues with the use of bi-directional

TCP by BitTorrent centered around the behavior of TCP with respect to *ACK piggybacking* and *fast-retransmit* under bi-directional conditions.

- *Piggybacking*: When bi-directional TCP is used, ACKs in the reverse path are almost always piggybacked on the data packets being sent in the reverse direction. In a wireless environment, where random errors rates can be non-trivial, this potentially has an adverse impact on the connection performance. More specifically, when ACKs are piggybacked on data packets, the effective “length” of the ACKs is longer than if they were sent as *pure* ACKs (non-piggybacked). Hence, for the same bit error rate (BER) in a wireless environment, the effective packet error rate (PER) for the ACK traffic is larger. This in turn results in more number of ACK packets being lost on the reverse path just because they were piggybacked.

While it is true that TCP uses cumulative ACKs, and hence is relatively robust to ACK losses, there still is a negative impact in terms of the overall throughput enjoyed by a connection in the presence of higher number of ACK losses. More importantly, in a P2P network peers typically have a large number of TCP connections ongoing even for a single swarm (BitTorrent trackers typically provide addresses of 50 peers in response to a request, but the overall swarm size can grow to numbers greater than 500), resulting in the average congestion window size of a TCP connection to be relatively small. And, it is for connections with small congestion window sizes that a higher ACK loss rate can result in a non-trivial degradation in throughput. In other words, the download rate for a TCP connection from a particular peer will be smaller just because of ACKs being piggybacked in the reverse direction.

We set up two peers which hold different portion of shared data, and obtain the throughput values when either uni-TCP or bi-TCP is used. Specifically, when two peers hold different data, they exchange data over bi-TCP. When only one peer has the data the other needs, data are downloaded using uni-TCP. Figure 17 (a) presents the 5-run averaged results of the download rate experienced by a peer under varying conditions of bit error rate on the wireless leg. Note that bi-directional connections will in general suffer in throughput when compared to uni-directional connections because of the self-contention between packets being sent in the upstream and downstream directions. However, that difference is captured by the data-point at BER=0.

- *Fast Retransmit*: The second issue with BitTorrent using bi-directional TCP occurs during congestion. TCP’s fast retransmit mechanism is based on the arrival of DUPACKs (duplicate ACKs) at the sender. When bi-directional TCP is used, TCP specifications stipulate that DUPACKs should not be piggybacked. The reason for this stipulation is that otherwise the sender has no way of knowing whether piggybacked DUPACKs are due of losses, or due to the receiver sending multiple data packets between two data packet arrivals (and hence piggybacking the same ACK sequence number on those data packets).

Hence, a TCP receiver will send *only* pure ACKs when DUPACKs have to be transmitted due to losses. While this design has no issues in a wired environment, it results in problems when performed as-is in a WLAN environment. This is because, when congestion has occurred in the WLAN resulting in a packet drop, the DUPACKs sent back will be explicitly de-coupled from the data stream in the reverse path, thus *actually increasing the number of packets in transit on the wireless leg*. While it is true that the TCP sender, when it receives the DUPACKs, will reduce its number of outstanding packets in the network by half, *the new transmissions of pure ACKs essentially offsets the decrease in the number data-packets*. Hence, as far as the wireless leg is concerned, the number of packets in transit stays approximately the same even after a congestion event!

Figures 17 (b, c) show the number of packets sent on the wireless leg with time for two BitTorrent connections of a typical run, one uni-directional and one bi-directional respectively. The congestion events are indicated in the same figure (buffer drop) as well. It can be observed that while the number of packets on the wireless leg in general decreases after congestion events for the uni-directional connection, this is not true for the bi-directional TCP connection. This “mis-behavior” can potentially result in performance degradation for the connections due to deviation from the intended behavior of TCP’s congestion control.

2) *Uploads based Incentives*: The tit-for-tat mechanism in BitTorrent encourages higher rates for uploads to enjoy better download rates. In a wired environment, it can be shown that peers enjoy their best download rates when their upload rates are high. We count the aggregate download rate of five simultaneous as a function of the upload rate limit in a wired setting. The upload rate limit is set as a fraction of the physical link capacity. The network is Comcast Cable High-speed Internet with 4Mbps downloading rate and 384Kbps upload rate. We observe

that the download rate is an increasing function of the upload rate limit.

However, for a wireless environment, the relationship between the enjoyed download rate and the upload rate limit changes. The aggregate download rate of the same five tasks is obtained in Georgia Tech Wireless LAN. As we observed, while the download rates initially increase with higher upload rates, beyond a much smaller upload rate (than 80%) the download rates actually drop. This is due to the *shared* channel nature of the wireless link, where the uploads and downloads are contending for the same wireless channel bandwidth. This is in contrast to a wired setting where the uploads and downloads do not share the same bandwidth resources. The same figure also demonstrates that shutting down the upload rates to zero is not a solution either as the tit-for-tat strategy of BitTorrent will kick-in punishing the peer with low download rates.

3) *Incentives and Mobility*: The tit-for-tat mechanism in BitTorrent is associated with a unique identifier for the peer called the *peer-id*. The *peer-id* is typically constructed as either a function of the IP address of the host and a random value, or simply as a function of a mobile host specific random value. The *peer-id* is regenerated every time fetch tasks are reinitiated. Thus when a mobile host experiences a hand-off and receives a new IP address, the ongoing tasks are terminated and the tasks are re-initiated<sup>3</sup> thus generating a new *peer-id*. However, since the peers track the *goodness* of corresponding peers based on the *peer-id*, this results in the mobile peer losing all the credit it has built with its corresponding peers.

We measure the effect of incentives on the download size for a 100MB file as a function of time. Under the no mobility scenario, we observe that the download size is lower when there is no upload traffic. This is the normal incentive behavior. However when we introduce mobility (IP address changes periodically) we see that the incentive mechanism is rendered ineffective. Not only is the actual download size lower than the no-mobility case, there is marginal difference between the download rates with or without uploading. This is because every time the IP address changes, tasks are re-initiated and thus the host acts as a new peer without any previous incentives. Thus, the mobility of a peer can have an adverse impact on the incentive mechanism of a P2P network.

4) *Server Mobility*: Peer address updates in BitTorrent happen at the granularity of tens of minutes when a peer goes back to the tracker for an updated set of peers to use in its swarm. Even within that period, the decision on whether to use a peer or not for downloads is adjusted at the granularity of tens of seconds. Note that this is understandable in a wired environment where disconnections of peers may not occur frequently. However, when a mobile peer undergoes a hand-off or simply gets disconnected, the fixed peers who were clients with respect to the mobile peer will continue to try to reach the mobile peer till either the peer selection algorithm chooses an alternate peer or the tracker provides alternate addresses or alternate peers (in the order of several minutes).

5) *Rarest-first Fetches*: BitTorrent employs a rarest first fetching paradigm. This results in any snapshot of the downloaded content for a file not having any significant “in-sequence” data from the head of the file till a large percentage of the file download is completed. Many media formats, on the other hand, allow for partial playback of content provided the partial information is in sequence. For example, for an MPEG file of a 2 hour video, the download of the first 30 minutes worth of the video will still allow for a playback of that part of the video. We measure the *playable* fraction of two files being downloaded with increasing fraction of the actual downloads using rarest-fetch. The piece length is the default value of 256 KB, and the results are averaged over 10 runs. It can be observed that until a large percentage of the whole file download is complete, a significant percentage of the file still remains unplayable.

#### D. Design

We now outline some key design aspects of the proposed *wP2P* solution that are targeted to address the limitations of existing P2P data networks identified. One key property of all the principles we present is that they are *mobile host only changes* that do not require any support from the fixed peers, and are fully backward compatible with already existing versions of the P2P (BitTorrent) protocols. Also, we present any specific implementation details with respect to the BitTorrent protocol. However, all the solutions presented are *purely local to the mobile host* and backward compatible to all existing BitTorrent P2P client applications on fixed peers.

<sup>3</sup>We assume here that mobile IP is not used to handle mobility due to its slow deployment.

1) *Age-based Manipulation*: The bi-directional TCP problem arises because of specific quirks of the TCP design and how they relate to the wireless environment. However, bi-directional TCP's performance otherwise is desirable since it eliminates ACK overheads under normal conditions. In other words, the solution to the problems with bi-directional TCP is not to switch back to dual unidirectional TCP connections as that would render the overall performance worse than when using bi-directional TCP as pure ACKs in both directions consume precious bandwidth resources.

In this context, the *Age-based Manipulation* (AM) design principle of *wP2P* involves the adaptive manipulation of the bi-directional TCP connections for better performance. Essentially, recalling the discussion on ACK loss rates, an argument can be made that TCP's throughput performance is vulnerable to ACK losses *only when the congestion window is small*. At larger congestion windows, the higher ACK loss rates do impact progress, but not significantly. Hence, under age-based manipulation, explicit conversion of piggybacked ACKs to pure ACKs is performed *when the connection congestion window (cwnd) is small*, and piggybacked ACKs are let through as-is when the congestion window is larger than a threshold.

The AM component constantly monitors the congestion window of the TCP connection and if the current connection congestion window is less than a specified threshold value  $\gamma$ , the connection status is set to *YOUNG*. Otherwise the connection status is set to *MATURE*. The AM component also maintains state about the TCP connection and captures TCP packets transmitted by the mobile host. If the connection status is *YOUNG*, the AM module conveys any new ACK information piggybacked on DATA packets transmitted by the mobile host as separate pure ACKs. This achieves better robustness for the ACKs given a finite error rate in the wireless channel. The captures and manipulates TCP packets in *wP2P* can be achieved using the WinpkFilter [10] framework that acts transparently to the existing protocol stack of the network.

2) *Incentive-Aware Operations*: The problem of failure of incentives stems from the two distinct conditions of the self contention in a wireless link and mobility related identity loss. The *Incentive-Aware operations* (IA) principle in *wP2P* addresses both problems. Essentially, one technique under incentive aware operations in *wP2P* involves the adaptation of the upload rate in order to find the smallest upload rate possible to achieve the maximum download rate. While this value for the upload rate is trivial to determine in a wired setting, a more sophisticated algorithm is required in a wireless environment.

Since a wireless host uses a shared channel, the upload and download traffic contend with each other. In order to strike the optimal balance between the two competing issues (incentives and self-contention) *wP2P* performs a Linear Increase History-based Decrease (LIHD) algorithm that adapts the uploading rate to an optimal value. The intuition behind the LIHD algorithm is that while increasing the upload rate, it is better to be conservative so that the mobile host does not upload more than necessary. At the same time while reducing the uploads it is desirable to be aggressive. LIHD hence increases upload rates linearly when there is a positive correlation between the uploads and downloads, while decreasing the upload rates with increasing aggressiveness when decreasing the uploads does not cause a decrease in the downloads.

The Incentive-Aware (IA) component monitors upload and download rates achieved by the P2P application. It uses window-averaged throughput to determine the upload rate control of the P2P application. It controls the upload rate in a way as to optimize the downloads achieved by the mobile host. If the download rate in the current time window is greater than the download rate achieved in the preceding time window, then the IA component increments the upload-rate counter. On the other hand, if the download rate in the current time window is less than the previously recorded download rate, then the upload rate counter is decremented by a value proportional to the number of consecutive cutdowns of the upload rate. This procedure achieves the optimal trade-off between incentive driven P2P downloads and wireless contention of the uplink and downlink transmissions.

Another technique *wP2P* uses that falls under this design principle is identity retention across hand-offs and within the same swarm. The rationale for generating uniquely different peer-ids in BitTorrent is to be able to identify and distinguish between clients with the same IP address (say, if the clients are behind a NAT), but at the same time confine the benefits of incentives accumulated by a peer to only that swarm in which the peer contributed. Since the typical scenario for task initiation in wired environments is when a peer wants to download another data file, generating a new peer-id is reasonable. However, in mobile environments task re-initiations can occur just because IP addresses have changed. *wP2P*, in this context, performs identity retention within a swarm,

whereby even when task re-initiation is performed, as long as it is for a swarm the mobile peer was a member of before, the old peer-id is retained. This enables the mobile peer to leverage its previously accumulated incentives. Thus, IA component stores the peer ID of the mobile host when the application is started and when there is IP layer handoff, the IA component restores the stored peer ID to maintain incentives.

3) *Mobility-Aware Fetching*: We observed how in a mobile peer (as a client) mobility can impact the performance of downloads in BitTorrent. *wP2P* uses a *Mobility-Aware Fetching* (MA) principle to deal with the problems associated with mobility. This principle explicitly controls how data is fetched. The mechanism is that of *exponentially increasing altruism* or *exponentially decreasing selfishness*. Essentially, a mobile peer fetches blocks in sequence with a probability  $p_s$  ( $=1 - p_r$ ), and fetches the rarest-first block with a probability  $p_r$ . This probability  $P_r$  is a function of the network stability of the mobile host as measured by the amount of time elapsed since the last network disconnection of the mobile host (or the start of the download). During the initial phases of the download, the mobile peer uses a small value (say, 20%) for  $p_r$ , and exponentially increases  $p_r$  as it downloads increasing fractions of the total file.

The rationale for this design is as follows: during the initial stages of downloads, if the mobile host gets disconnected, there is no benefit due to the rarest-fetch mechanism *either for the mobile host (in terms of playability) or to the P2P network (in terms of availability)*. Hence, it is more desirable to fetch sequentially. However, as the mobile host stays connected for a longer period of time, *its utility to the P2P network* has more stability and hence it is more meaningful to have available rare blocks. Furthermore, if the mobile host now gets disconnected, the user still has a considerable portion of the data in-sequence for playback. Thus, This mobility-aware adaptive content fetching achieves a more desired tradeoff between *sequential content availability for disconnected usage of content* and *usability of content for other peers to download*.

## E. Key Results

1) *Implementation*: We use a prototype implementation of *wP2P* that is built as an enhancement to the CTorrent client version 1.2 [4], which is a lightweight C++ implementation of BitTorrent protocol with about 10K lines of code. All the three components of *wP2P* are implemented by either modifying the source code of CTorrent or adding a separate module which works with a packet filtering utility widely available in Linux distributions [7].

- *Network Setup*: We use two wireless clients on a popular BitTorrent network to compare the performances of *wP2P* with a default version of BitTorrent. One client runs the modified CTorrent version and the other runs plain vanilla version of the CTorrent (we call this as the default client). The clients are connected to the Internet through ns-2 based wireless emulators. We use ns-2 emulation to study the impact of various issues of wireless environments. We emulate random wireless losses using random bit errors. We emulate mobility by changing the IP addresses of the clients using the “ifup/ifdown” commands in Linux. We also monitor the bandwidth consumed at each client to enforce bandwidth limitations.

- *Age-based Manipulation*: The operations of this component require the determination of the connection’s age. The determination is based on the measurement of current congestion window. Since the information of congestion window typically is not available to the application itself, the realization of this component has to obtain such information from certain networking entities. Specifically, we choose Netfilter utility to assist the implementation of the component partly due to its wide deployment in Linux distributions. A module in the user space keeps track of the amount of data sent by the remote peer in every round trip time (rtt), and uses the current value as an estimate of that peer’s TCP congestion window for the next rtt. We chose a congestion widow of size 9k bytes (approximately 6 full packets) as an indicator of the age of the flow.

- *Incentive-Aware Operations*: This component employs two techniques: identity retention and Linear Increase History based Decrease (LIHD) rate control. For the first technique a static peer ID is used in lieu of a randomly generated peer ID every time the IP address changes. For the second technique we modify CTorrent’s in-built capability to control upload and download limits. The default CTorrent client allows users to specify the upload and downloads limits. We modify it to allow the adaptive LIHD rate control algorithm described in the previous Section. We use bandwidth monitors to check the current upload and download rates for the algorithm.

- *Mobility-Aware Fetching*: The basic CTorrent client does not implement the rarest first fetching algorithm commonly used by BitTorrent clients. Hence we first implement the rarest first fetching algorithm for the default

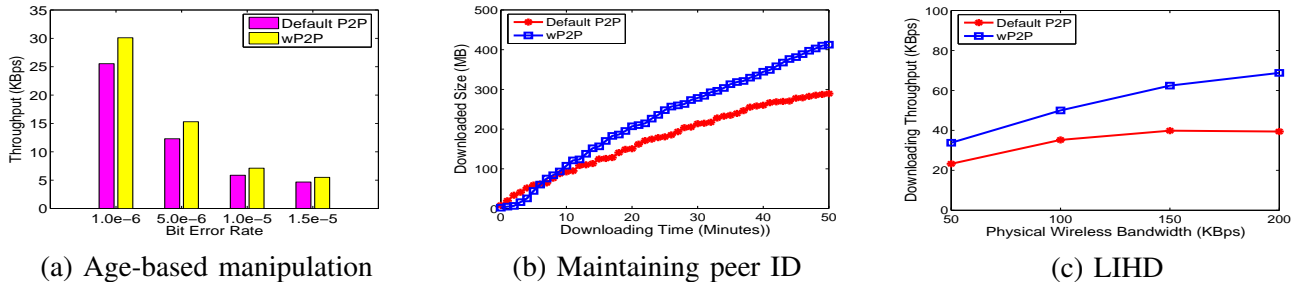


Fig. 18. Age-based manipulation (a), Incentive-aware operations (b, c)

client. We then modify this algorithm to include sequence information and awareness of download progress. Specifically, at the start of the download higher probability is given for low sequence numbered pieces as opposed to high sequence numbered ones. As the download progresses, the default rarest first search algorithm gains prominence.

2) *Age-based Manipulation*: *wP2P* addresses the problems of Bi-directional TCP in wireless environments using the AM component. We study the impact of this component under varying random loss conditions emulated by varying the BERs ranging from  $1e-6$  to  $1.5e-5$ . Initially we setup a single seed for a 100MB file and use the two CTorrent clients as leeches. We allow each of the leech to initially download about 50% from the seed (and the other leech). We then remove the seed so that any further data transfer is just between the two leech peers. We compare the download rates observed by the two leech peers for five runs and show the averaged results in Figure 18(a). We observe that *wP2P* outperforms the default CTorrent under all bit error rates because decoupling ACKs result in smaller ACK losses for the connection, and in turn, larger throughput. Specifically, with all the four BER values, *wP2P* achieves about 20% more throughput.

3) *Incentive-Aware Operations*: As discussed in the previous Section identity retention and LIHD rate control address the issue of failure or loss of incentives. To evaluate identity retention we use the two CTorrent clients to simultaneously download a Fedora-7-KDE-Live-i686.iso (<http://torrent.fedoraproject.org/>) image, a 688MB file shared among more than two hundreds peers when our experiments were conducted. The IP addresses of the two clients are changed every one minute to emulate mobility. Figure 18(b) shows the total downloaded size of a typical run for these two peers. The downloaded size is plotted as a function of time. For the default client whenever incentives are lost the download rate is reset to the initial nominal value unlike in *wP2P* where the incentives are maintained. Hence we find a higher downloads for *wP2P* compared to the default for the same time. After 50 minutes of download we observe that *wP2P* downloaded about 100MB more than the default.

To evaluate LIHD we vary the bandwidth of the wireless emulator from 50KBps to 200KBps. Figure 18(c) shows the averaged results over 10 runs for the case when  $\alpha = \beta = 10KBps$ . We observe that, initially as the available bandwidth increases both *wP2P* and the default client show increased download throughput, but beyond a certain point the default client starts losing achieved throughput. With a bottleneck bandwidth of 200KBps we observe that *wP2P* outperforms the default by as much as 70%.

4) *Mobility-Aware Fetching*: Figures 19 show the results of the Mobility-Aware Fetching technique for different file sizes of the content being downloaded and compare them against the default rarest first fetch algorithm. The results are averaged over 20 runs. In these experiments we set the value of  $p_r$  to be equal to the downloaded percentage of file. We observe that MF can achieve significantly better performance compared to the default. For instance, for a 5MB file, when 50% of the data has been downloaded, MF can result in about 30% of playable content while the default rarest first technique can achieve only about 5%. The improvements are even more prominent when the file size increases.

## F. Summary

In this work we investigated the issues with using mobile hosts as peers in a P2P network. We identified several insights into the issues such hosts face using a real-life BitTorrent P2P data network. We proposed a solution called

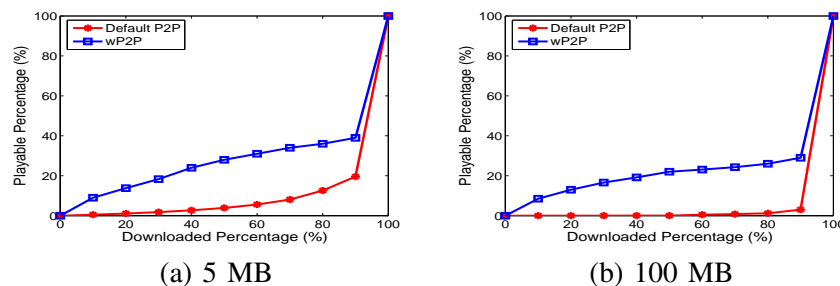


Fig. 19. Mobility-aware Fetching

wP2P that significantly improves performance and also evaluated the solution using a real life implementation as an extension of the CTorrent client.

## VIII. SUMMARY

In this report, we have presented new principles for enhancing real-time and non-realtime traffic, and more specifically VoIP and Web applications. The principles result in significant improvements to end-user experience. While we have presented the principles as stand alone transparent approaches, we believe that a realistic long-term strategy is for them to be absorbed into transport layer protocols.

## IX. LIST OF PUBLICATIONS

- *A<sup>3</sup>: Application-Aware Acceleration for Wireless Data Networks*, Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham, ACM International Conference on Mobile Computing and Networking (MOBICOM) Los Angeles, CA, USA, September 24-29, 2006.
- *Improving VoIP Call Capacity over IEEE 802.11 Networks*, Yeonsik Jeong, Sandeep Kakumanu, Cheng-Lin Tsao, and Raghupathy Sivakumar, The fourth International Conference on Broadband Communications, Networks and Systems (Broadnets), Raleigh, NC, USA, September 2007.
- *Client-Side Web Acceleration for Low-Bandwidth Hosts*, Tae-Young Chang, Zhenyun Zhuang, Aravind Velayutham, and Raghupathy Sivakumar, The fourth International Conference on Broadband Communications, Networks and Systems (Broadnets), Raleigh, NC, USA, September 2007.
- *WebAccel: Accelerating Web Access for Low-Bandwidth Hosts*, Tae-Young Chang, Zhenyun Zhuang, Aravind Velayutham, and Raghupathy Sivakumar, to appear in Elsevier Computer Networks, 2008.
- *On the Impact of Mobile Hosts in Peer-to-Peer Data Networks*, Zhenyun Zhuang, Sandeep Kakumanu, Yeonsik Jeong, Raghupathy Sivakumar, and Aravind Velayutham, The 28th International Conference on Distributed Computing Systems (ICDCS 2008), Beijing, China, June, 2008.
- *Application-Aware Acceleration for Wireless Data Networks: Design Elements and Prototype Implementation*, Zhenyun Zhuang, Tae-Young Chang, Raghupathy Sivakumar, and Aravind Velayutham, submitted to IEEE Transactions on Mobile Computing, 2008.
- *Mobile Hosts Participating in Peer-to-Peer Data Networks: Challenges and Solutions*, Zhenyun Zhuang, Sandeep Kakumanu, Yeonsik Jeong, Raghupathy Sivakumar, and Aravind Velayutham, submitted to ACM/Kluwer Wireless Networks Journal (WINET), 2008.
- *VoIP over Wi-Fi Networks: Performance Analysis and Acceleration Algorithms*, Yeonsik Jeong, Sandeep Kakumanu, Cheng-Lin Tsao, and Raghupathy Sivakumar, submitted to ACM/Kluwer Mobile Networks and Applications Journal (MONET), 2008.

## REFERENCES

- [1] Bittorrent. <http://www.bittorrent.org/>.
- [2] CacheLogic Report: Peer-to-Peer in 2005. <http://www.cachelogic.com/home/pages/research/p2p2005.php>.
- [3] CIFS: A common internet file system. <http://www.microsoft.com/mind/1196/cifs.asp>.
- [4] Enhanced ctorrent, a lightweight c++ implementation. <http://www.rahul.net/dholmes/ctorrent/>.
- [5] Hypertext transfer protocol- http/1.1. <http://www.ietf.org/rfc/rfc2616.txt>.

- [6] Linux magazine. <http://www.linux-magazine.com/issue/15/>.
- [7] Netfilter project. <http://www.netfilter.org/>.
- [8] *Simple mail transfer protocol*. <http://www.ietf.org/rfc/rfc0821.txt>.
- [9] *W3C Document Object Model*. <http://www.w3.org/DOM/>.
- [10] Winpkfilter. <http://www.ntkernel.com/>.
- [11] H. Balakrishnan and R. Katz. Explicit loss notification and wireless web performance. In *Proceedings of IEEE GLOBECOM*, Sydney, Australia, Nov. 1998.
- [12] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt. Performance optimizations for wireless wide-area networks: Comparative study and experimental evaluation. In *MobiCom '04: Proceedings of the 12th annual international conference on Mobile computing and networking*. ACM, 2004.
- [13] T.-Y. Chang, A. Velayutham, and R. Sivakumar. Cut-load: Application-unaware content partitioning for web-based information access in wireless data networks. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2006.
- [14] T.-Y. Chang, Z. Zhuang, A. Velayutham, and R. Sivakumar. Client-side web acceleration for low-bandwidth hosts. In *Proceedings of IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, 2007.
- [15] D. Dwyer and V. Bharghavan. A mobility-aware file system for partially connected operation. *SIGOPS Oper. Syst. Rev.*, 31(1):24–30, 1997.
- [16] T. Henderson and R. Katz. Transport protocols for Internet-compatible satellite networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 17(2):345–359, Feb. 1999.
- [17] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *MobiCom '04: Proceedings of the 12th annual international conference on Mobile computing and networking*, 2004.
- [18] H. Hsieh, K. Kim, Y. Zhu, and R. Sivakumar. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. In *MobiCom '03: Proceedings of the 12th annual international conference on Mobile computing and networking*, 2003.
- [19] IXIA. <http://www.ixiacom.com/>.
- [20] Y. Jeong, S. Kakumanu, C.-L. Tsao, and R. Sivakumar. Improving voip call capacity over ieee 802.11 networks. In *Proceedings of IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, 2007.
- [21] A. Kamerman and L. Monteban. WaveLAN-II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 2(3):118–133, 1997.
- [22] A. Kuzmanovic and E. W. Knightly. Tcp-lp: low-priority service via end-point congestion control. *IEEE/ACM Trans. Netw.*, 14(4):739–752, 2006.
- [23] Ringtonia, <http://www.ringtonia.com/>. *Melodeo's mobile phone P2P to launch*.
- [24] Roadcasting, <http://www.roadcasting.org/>. *A new type of radio*.
- [25] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. In *MobiCom '99: Proceedings of the 12th annual international conference on Mobile computing and networking*, Seattle, WA, USA, Aug. 1999.
- [26] The Network Simulator. ns-2. <http://www.isi.edu/nsnam/ns>.
- [27] R. Vedantham, S. Kakumanu, S. Lakshmanan, and R. Sivakumar. Component based channel assignment in single radio, multi-channel ad hoc networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, New York, NY, USA, 2006. ACM.
- [28] Z. Zhuang, T.-Y. Chang, R. Sivakumar, and A. Velayutham. A<sup>3</sup>: application-aware acceleration for wireless data networks. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 194–205, New York, NY, USA, 2006. ACM.

# NSF CNS-0519733

## Principles and Design of Unified Transport Layer Solutions for Heterogeneous Wireless Data Networks

### Annual Report 2007

Faculty: Raghupathy Sivakumar  
Students: Tae-Young Chang, Chen-Lin Tsao, and Zhenyun Zhuang

School of Electrical and Computer Engineering  
Georgia Institute of Technology  
Atlanta, GA 30332

#### I. OVERVIEW

A tremendous amount of research has been done toward improving realtime and non-realtime application performance over wireless data networks. Over the past one year, we have focused on developing new principles for transporting such content [1]–[3], and specifically Voice-over-IP (VoIP) and HTTP applications, over wireless data networks [4], [5].

In this context, we have developed two solution suites: (i) VoIP capacity enhancement over wireless data networks using the principles of packet aggregation, size dependent PER calculation, and adaptive ACK blocks; and (ii) client-side web enhancement over wireless data networks using the principle of fetch-what-affects-user-experience more aggressively.

The expected VoIP call capacity in a one-hop IEEE 802.11b network with G.711 voice codec is about 85 simultaneous calls, but the actual observed capacity is only 5 calls even at the highest rate and under zero loss conditions. In this report, we analyze the reasons behind this inferior performance of VoIP traffic, and present software-only VoIP enhancement algorithms at the medium access control layer to improve the observed call capacity. Finally, using *ns2*-based simulations, we evaluate the algorithms and show that performance improvements of up to 300% can be achieved.

Current popular web-browsers simply fetch the entire web-page from the web-server in a greedy fashion. This simple web fetching mechanism employed by browsers is inappropriate for use in low-bandwidth networks, since they cause large response times for users unnecessarily. In this report, we first analyze the reasons that cause large response times by considering several factors including the properties of typical web-pages, the properties of current web-browsers, the interaction of the HTTP and TCP protocols, and the impact of server-side optimization techniques. We then propose client-side web enhancement mechanisms to reduce the user response time. Through *ns2*-based simulations, we compare the performance of our solution with that of current browsers and show that the proposed scheme brings significant performance benefits in terms of user-perceived response times.

While we present the approaches and approaches as stand-alone transparent mechanisms, the realistic long-term adoption strategy for these are as transport layer mechanisms.

#### II. STUDENTS SUPPORTED

The graduate students supported under this grant are Mr. Tae-Young Chang, Mr. Chen-Lin Tsao, and Mr. Zhenyun Zhuang.

### III. PRINCIPLES FOR ENHANCING VOICE OVER IP LIKE REAL-TIME TRAFFIC

#### A. Overview

Voice over IP (VoIP) services have been significantly gaining prominence over the last few years because of a number of impressive advantages over their traditional circuit-switched counterparts including but not limited to high bandwidth efficiency, low cost, and flexibility of using various compression strategies. Simultaneously, the use of wireless networks has also grown tremendously over the past years. Wireless LAN (WLAN) solutions, armed with better physical layer technologies, now promise high data rates of more than 100 Mbps (IEEE 802.11n).

In this context, recent efforts have focused on marrying the potential benefits of VoIP and WLANs to provide wireless telephone services. A natural question that then arises is *how well does VoIP perform over WLAN environments?* The answer to this question is counter-intuitive. Even though the WLANs boast very high data rates and a typical VoIP call carries only 128 Kbps of bidirectional data (using G.711 voice codec), the number of VoIP calls supported by these networks is abysmally low. An IEEE 802.11b network for example can support only 5 VoIP calls even at the highest data rate of 11 Mbps, as we will show later. VoIP traffic in WLANs is characterized by its small frame sizes, and IEEE 802.11 MAC is notorious for very poor performance for small frame sizes. For small frames the overheads at the different layers of the network stack themselves pose a significant burden. Added to this IEEE 802.11 MAC protocol exhibits other distributed effects that further degrade the VoIP call capacity.

Toward the goal of improving the VoIP call capacity in wireless networks we first study the reasons behind the inferior performance of VoIP over WLANs. We identify the universal set of components that could be improved to increase VoIP call capacity over such environments. Finally, we select the three dominant components and present algorithms that improve the performance of the components, thereby increasing the VoIP call capacity. While most of the discussions in the report apply to any of the IEEE 802.11 class of WLAN environments, we present all our discussions only in the context of IEEE 802.11b for clarity.

#### B. Motivation and Algorithm Design

We can write an integrated equation for the Maximum Number of VoIP Calls (MNVC) in an 802.11 network as,

$$\text{MNVC}(\text{PHY}) = \frac{1}{2k \left( T_{\text{DIFS}} + T_{\text{BO}} + T_{\text{SIFS}} + T_{\text{PHY}} + T_{\text{ACK}} + \frac{8(D + \sum_{L=\text{RTP}}^{\text{MAC}} H(L))}{R} \right)}, \quad (1)$$

where  $k$  is the frame rate of codec,  $D$  is the frame size, and  $H(L)$  is the size of header of the layer  $L$ .  $T_{\text{DIFS}}$ ,  $T_{\text{BO}}$ ,  $T_{\text{SIFS}}$ , and  $T_{\text{ACK}}$  are MAC layer overheads of DIFS, backoff time, SIFS, and ACK, respectively and  $T_{\text{PHY}}$  is the transmission time.

Analyzing this equation, we can envisage 5 different schemes to improve the call capacity by leveraging different components in the equation:

- *ACK Aggregation (AA)*: ACK aggregation refers to sending a single ACK for a block of  $n$  frames. ACK aggregation results in the *reduction* of  $T_{\text{ACK}}$ . From TABLE I we see that we can increase the call capacity by up to 2 calls.
- *Frame Aggregation (FA)*: Frame aggregation refers to fusing multiple frames destined to the same end user into a single large frame. TABLE I shows that frame aggregation gives the maximum possible improvement in terms of MNVC. However frame aggregation is associated with an increase in delay of the frames. In section III-B.2 we present an algorithm for frame aggregation that serves as a balance between the MNVC and the expected delay. FA *decreases*  $k$  in Eq. 1.
- *Link Adaptation (LA)*: Link adaptation refers to changing the transmission rate for the data frames. The 802.11b standard specifies 4 different data rates that can be used. The prevailing channel conditions affect the choice of data rate. Generally a high rate is used in conditions of low error and vice-versa. Existing rate adaptation techniques do not consider the size of frames while adapting to different rates. VoIP traffic is characterized by small frame size and hence will suffer lower frame drop rates than normal traffic for the same channel conditions. Hence there is a strong motivation for a rate adaptation technique that takes into account the frame size. LA can *control*  $R$  in Eq. 1 and maintain it optimal for a given channel condition.

TABLE I  
ANALYSIS OF MNVC(PHY) IMPROVEMENT USING DIFFERENT SCHEMES.

AA		FA		LA		TS		HC	
# of ACK	MNVC	$k$ (/s)	MNVC	$R$ (Mbps)	MNVC	$T_{DIFS}$ ( $\mu$ s)	MNVC	$\Sigma H(L)$ (Bytes)	MNVC
1	5.1	100	5.1	11	5.1	50	5.1	74	5.1
1/2	6.1	50	9.7	5.5	4.6	10	5.3	48	5.2
1/4	6.7	25	17.4	2	3.4	0	5.4		
1/8	7.1	12.5	28.9	1	2.4				
0	7.5	6.25	43.2						

- *Time Saving (TS)*: Time saving refers to *reducing*  $T_{DIFS}$  waiting time between two successive frames. From TABLE I we see that the potential benefits as a result of this technique is less than a single call. The reason for this fact is that the time interval of DIFS is very small in the order of  $50\mu sec$  and is considerably low when compared to other MAC delays like backoffs and frame transmission times.
- *Header Compression (HC)*: Header compression refers to reducing the size of the various headers like the RTP/UDP/IP headers using techniques either proposed in literature or otherwise. This technique also has limited potential in improving the call capacity in tune of only 0.1 calls. HC *reduces*  $\sum_{L=RTP}^{MAC} H(L)$  in Eq. 1.

In this report we consider AA, FA and LA techniques only because they have a good potential to improve call capacity, while TS and HC provide a maximum of less than one call improvement and hence we ignore them in developing new algorithms.

1) *ACK Aggregation (AA)*: Under saturated conditions the presence of ACK impacts the capacity of an 802.11 network. The 802.11 MAC specifies that the ACK should be sent at the basic rate. Hence even though the size of ACK is very small it eats up considerable amount of the capacity. Eq. 1 captures the impact of the ACK duration on the MNVC that can be supported. Hence one can envisage a scheme wherein ACKs are sent for every  $n$  frames instead of for every frame. We define the set of  $n$  frames as a block and the ACK for every block can contain information about the number of frames received correctly and the number in error. Only the lost frames can be retransmitted. TABLE I summarizes the improvements one can expect by aggregating ACKs. We observe that there is a difference between using a block size of 1 and a block size of 8 but there isn't much difference between 8 and  $\infty$ . This is because of the presence of other terms in the denominator of Eq. 1.

Fig. 1(a) shows the variation in throughput (observed maximum frame rate) by varying the block sizes. We observe that smaller block sizes result in smaller delay for the frames but lower total throughput as well. A larger block size on the other hand has larger delay for the frames but higher overall throughput. This is because when there is an error in a large block size scenario, it takes longer time to retransmit. However the better overall throughput given by a larger block size comes from the higher capacity due to smaller number of ACKs.

Even though the destination delays sending ACK to the source until all the frames belonging to one block are received, the destination can reduce the play buffer delay by sending the uncorrupted frame up to the upper layer whenever it receives a frame.

Based on the above observation that there are cross over points we can think of an algorithm that changes the block size adaptively. We simulate this scheme as a 2.5 layer solution in between the MAC layer and the interface queue. We assume that there is no preset block size and we send a block ACK request from the source once all frames in the present block are sent. Upon receiving the block ACK request, the destination responds with a block ACK containing the required information. The source then starts a new block and retransmits the required frames of the previous block and continues with newer frames. Because of the presence of a block ACK request we can change the size of block at will. We implement a simple adaptive scheme where we increase the block size upon receiving a block ACK with all successes and reduce the size on receiving a block ACK with even a single data loss. The adaptive algorithm chooses the right block size depending on the number of losses in the current block. Thus the adaptive algorithm should give a performance that is the best of both worlds i.e it should have a good delay characteristic as well as high saturated capacity.

2) *Frame Aggregation (FA)*: The MAC throughput in an 802.11 network is limited by the size of the frame. Fig. 1(b) shows the maximum throughput observed in a one-hop single flow scenario using an error-free 802.11b

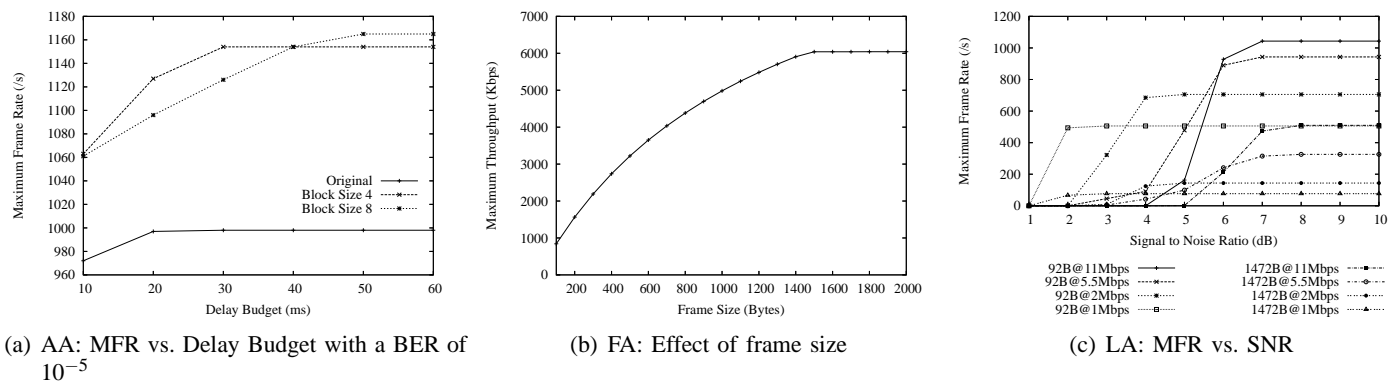


Fig. 1. Motivation results of AA/FA/LA.

11 Mbps link (with RTS/CTS mechanism suppressed). We observe that for a small frame size, of say 120 bytes (the size of a G.711 encoded VoIP frame with RTP, UDP, and IP headers), the maximum throughput is about 931 Kbps, which means that we can get only 7.2 calls from such a network. The reason for this inferior performance is because of the various overheads involved, like the PLCP header, DIFS and SIFS intervals and random backoff. The effect of the overheads decrease as the frame size increases.

An obvious scheme to improve the call capacity will be to aggregate frames. That said, it is a non trivial problem as to how one should go about aggregating frames. A direct consequence of aggregating frames is increased delay. However, there is a delay budget that can be accommodated by the voice codecs and it varies between codecs. Towards this goal of achieving a balance between the number of frames that can be aggregated and the delay budget, we propose an intelligent scheme which scales the number of voice calls that can be accommodated as the delay budget increases. With a delay budget of 100 ms we can get as many as 25 calls at the same time. We also show that such a scheme can also perform very well in the presence of other background traffic.

The core idea behind the algorithm is that we should aggregate frames only when required. Specifically, we aggregate only those frames that have already been received by the underlying queue that precedes the MAC layer in the network stack. The intuition behind such a scheme was the observation that a number of frames were being dropped at the AP because of the queue getting filled up when the number of calls increased beyond 5. Frames are aggregated just before transmitting on the physical layer after all the waiting for DIFS, SIFS and/or backoff times at the MAC layer. We limit the maximum size of the frame that is transmitted to 2304 bytes (maximum allowed frame size in 802.11 standard).

In a purely downstream environment (only AP to client communication) the number of flows that can be accommodated increases as the delay budget allowed increases. This saturates beyond a point because of the maximum frame size limitation. In the upstream case however we do not observe a phenomenal increase in capacity. This can be explained as follows. Each client has only one flow (100 frames per second in our case) and it competes with other nodes for a transmission slot. By the time a client gets a slot to transmit it has typically only one or two frames to aggregate. The AP on the other hand has a number of flows but still contends as one node in network. Hence there will be a number of frames for each flow when it gets to transmit.

To correct this imbalance between downstream and upstream flows we propose an enhancement to the piggy-backing algorithm. We make modifications at both AP and client sides. At the AP we increase the buffer size of the queue to hold up to 500 frames. At the client side we maintain a variable that holds the number of frames aggregated in the previously received frame from the AP. When the client gets a chance to transmit it checks if it has frames at least equal to that variable. If yes, it goes ahead to transmit the aggregated frame. However if it does not have enough frames it goes into a random backoff without increasing the contention window size.

We do not increase the contention window size because we do not want to wait for a very long time because of the delay budget. Since each client now waits for the same number of frames as in downstream it will contend for the channel at large time intervals. The AP on the other hand will contend for each of its client one after the other. Thus the contentions are now balanced between the upstream and downstream nodes depending on the

amount of traffic. We refer to this scheme as *enhanced piggybacking*. We study the behavior of this scheme using *ns2* simulations and the results are discussed in the next section.

3) *Link Adaptation (LA)*: It is intuitive that the MNVC at a higher rate would be larger than at a lower rate. However this is true only when we assume that the link is error-free. When the signal-to-noise ratio (SNR) of the channel is low, it would be advisable to transmit at a lower rate. This is because for a given SNR the loss for a lower rate transmission is lower than at a higher rate.

Several schemes have been proposed in literature and some like the ARF [6] have been implemented in practice to adapt the transmission rate depending on the prevailing channel conditions. But all these schemes either adapt to the number of frames in error or use SNR measured at the receiver and communicated to the sender via CTS frame. These schemes do not consider the data frame size for making the decision on the data rate.

There is an inverse relation between SNR and bit error rate (BER) which directly translates to frame error rate (FER) depending on the frame size. Hence for a given BER the FER of a small frame might be acceptable whereas the FER for a large frame size might be unacceptable. Fig. 1(c) illustrates this fact. We observe that for an SNR of 6 dB, a data rate of 11 Mbps might be acceptable for small frame size but unacceptable for large frames. This gives us a motivation for a size aware rate adaptation technique.

ARF is a popular rate adaptation technique that is widely deployed in many existing WLANs. It is a simple but robust technique that chooses the rate depending on the reception or failure of the successive ACKs for the DATA frames. Briefly the ARF algorithm works as follows: For every ten consecutive ACKs correctly received the rate is increased to the next higher allowed value and upon two consecutive ACK failures the rate is reduced to the immediate lower allowed rate. As is evident from the algorithm this technique does not take into account the frame size during rate adaptation. It is a known fact that ARF does not perform very well under rapidly changing channel conditions because it depends on ten successful ACKs to increase the rate. This problem of ARF worsens when the frames are of varying sizes. VoIP frames are characterized by small frame sizes and when they are interspersed with large frame sized background traffic the ARF will suffer in the sense that frames will be sent at wrong rates.

The core idea behind Size-aware Auto Rate Fallback (SARF) is that if a small frame is in error then there is a high probability of error for a large frame as well. Similarly when a large frame is successful, there is a very high probability of success for small frames as well.

The source maintains two counters (success counter and failure counter) for different frame sizes. These counters are updated (based on the previous conjecture) upon reception or failure of ACKs. When these individual counters reach preset threshold values the rate for the corresponding frame size is modified (either increased or decreased).

The adaptive algorithm is similar to ARF i.e upon successive ACKs received correctly the rate is increased to the next level and upon consecutive ACK failures the rate is reduced. The threshold values are chosen based on some heuristic methods. We observe that even though the MNVC does not improve drastically for SARF the number of frames sent at wrong rate is significantly less than that of ARF. The packets sent at wrong rate may incur either large delay due to lower rate than ideal or much loss due to higher rate than ideal.

### C. Performance Evaluation

We use the open source *ns2* simulator [7] for all our simulations. Unless otherwise stated bi-directional CBR traffic with a frame size of 92 bytes is used for modeling the VoIP traffic. The frame rate of traffic is set at 100 frames per second in either direction i.e AP to client and client to AP. Each call originates from a wired node and has two wired links and one final 802.11b wireless link in an infrastructure mode. In scenarios where background traffic is required we use another CBR traffic with a frame size of 1472 bytes. The rate of the background traffic is varied according to the need.

1) *ACK Aggregation (AA)*: To study the performance of the ACK aggregation scheme we setup bi-directional CBR flows between wired node and client node in the wireless domain. We use MFR as a fine grained metric to study the improvement of this scheme.

Fig. 2(a) shows the performance results of the ACK aggregation algorithm at different BERs. The capacity improvement due to the aggregation of ACKs is evident from the saturated frame rate at higher delay budget. The saturation throughput is higher for a larger block size. This is because for a larger block size smaller number of

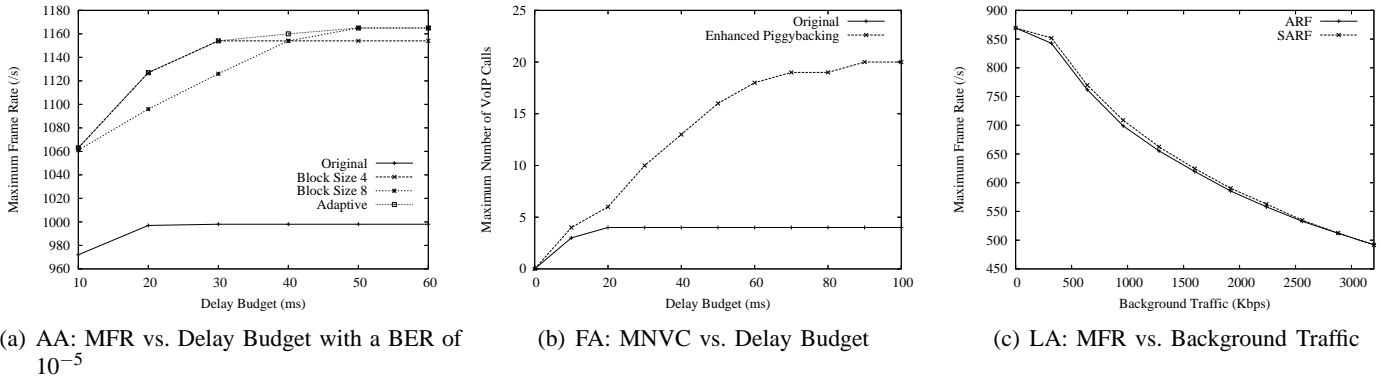


Fig. 2. Simulation results of AA/FA/LA.

ACKs are sent thus making more room for data frames. However when a frame gets lost it will take longer time to get retransmitted if the block size is large. This delay gets worse when the BER is higher. The adaptive algorithm chooses the right block size depending on the number of losses in the current block. Thus the adaptive algorithm gives performance that is the best of both worlds. It has both better delay characteristics as well as better saturated frame rate.

2) *Frame Aggregation (FA)*: To simulate the number of VoIP calls using the enhanced piggybacking scheme we assume an erroneous environment with a BER of  $10^{-5}$ . We setup a number of bi-directional CBR flows with a rate of 100 frames per second in either direction. To study the number of VoIP calls for a given delay budget we increase the number of such bi-directional flows to different clients and stop when the loss at the given delay is more than 2% of the frames. Losses occur due to both erroneous link (BER) and frames exceeding the delay budget.

Fig. 2(b) shows the MNVC that can be accommodated for a given delay budget. We compare the enhanced piggybacking scheme with the original data-ACK scheme. The original scheme peaks at a maximum of 4 calls, which is one smaller than analysis result due to error condition and delay budget. However the enhanced piggybacking scheme gives up to 20 calls for a delay budget of 100 ms and 18 calls for a delay budget of 60 ms. Note that 60 ms is an acceptable wireless delay for good quality of VoIP.

The enhanced piggybacking scheme is able to scale with increasing delay budget. One drawback of the enhanced piggybacking scheme is that as the number of calls increase the delay for every user increases. A suitable scheme at the AP should be employed to limit the number of calls based on a given delay budget. A possible QoS policy can be applied if we can measure the delay for each frame for the leg of transmission from AP to the far end client. This delay information can be used to set the delay budget of the wireless leg of the call. This will be part of our future work.

3) *Link Adaptation (LA)*: We use a single VoIP call and a single background CBR source with large frame size (1472 bytes) to study the efficacy of SARF. We compare the performance of SARF with ARF. We assume a hidden Markov channel model [8] to simulate varying channel SNR conditions.

Fig. 2(c) shows the MNVC for ARF and SARF for varying background traffic. We observe that both SARF and ARF perform similarly. This is because both ARF and SARF are based on similar principles of tracking the channel condition based on success or failure of ACKs. SARF also performs well for other QoS metrics like delay budget but we do not present those details due to lack of space.

#### D. Summary and Future Work

In this report, we analyze the reasons behind the inferior performance of VoIP traffic over wireless LANs. We setup an experimental testbed to serve as a testbed baseline and to justify our analysis. While doing so we propose three algorithms at the MAC layer to improve the observed call capacity namely ACK Aggregation (AA), Frame Aggregation (FA), and Link Adaptation (LA). Extensive ns-2 simulations are performed to show the efficacy of

these algorithms. We find that FA can provide capacity improvements in the order of up to 300%. As part of our future work we will complete the analysis and simulation for a variety of codecs and IEEE 802.11 variants. We also plan to implement both software only and standard compliant solution of the proposed algorithms in the real testbed.

We will publish the software-only VoIP acceleration concept in IEEE International Conference on Broadband Communications, Networks, and Systems [4].

#### IV. PRINCIPLES FOR ENHANCING WEB ACCESS

##### A. Overview

In the past couple of decades, tremendous amount of research has been done on improving web access performance over Internet. In this report, we study the performance of a web-browser under low-bandwidth network conditions, such as in wireless networks. Specifically, we analyze the characteristics of a web-browser with the objective of identifying reasons as to why they might suffer in low-bandwidth conditions. We find that the current fetching model employed by commercial web-browsers is not optimal in bandwidth challenged environments.

We show that the absence of content prioritization and intelligent object fetching mechanisms in current web-browsers leads to increased response times. Browsers, today, do not prioritize the useful data that is viewed by the user over other data in the web-page. With a greedy fetch of the entire content of a web-page, precious bandwidth is wasted and in turn increases user perceived response time. Further, without intelligent object fetching mechanism, the download process of current web-browsers does not utilize the network bandwidth efficiently.

In this report, we propose a client-side only solution to address the problem of large response time with current browsers. The solution is based on careful consideration of several factors including the content displayed on the screen viewed by the user, server-side content distribution networks, and the relationship between the HTTP and TCP protocols. The three mechanisms we propose include prioritized fetching, object reordering and connection management. One major advantage of our approach is that it is pure client-side enhancement and does not need any server changes. The proposed solution helps to reduce response time, and is easy to deploy since it only requires client-side installation to current web-browsers.

##### B. Motivation

In this section, we describe drawbacks in the conventional web access model in low-bandwidth environments and use them as motivation for designing a new web access scheme.

*1) Screen Contention Problem:* When a user is viewing a screen on a display device, objects for displaying other screens are unnecessary in the sense that they are not visible to the user at this time. However, in conventional web browsers, the process of fetching *necessary* on-screen objects (*i.e.* objects on current screen) may be slowed down due to the competition from the process of fetching *unnecessary* off-screen objects. We refer to the fact that objects from different screens are competing for bandwidth as *screen contention*.

The main reason of screen contention is disparity of cumulative transfer-size among multiple connections. As mentioned earlier, a parsing engine inserts object requests to message queues in a round-robin fashion that considers only fairness in terms of the number of objects per connections. As a result, some connections having only small-size objects may finish transmissions of on-screen objects early and begin to fetch off-screen objects. Under this scenario, different connections may fetch objects on different screens simultaneously.

Another possible reason is directionality in a table structure. When a table is defined in a web page so that one of internal cells has larger vertical pixel-size than the client area in the browser window, a web browser fetches off-screen objects located at the end of this cell first, and then begins to fetch on-screen objects in the next cells. Figure 3 shows an example of the object fetching sequence in Internet Explorer at the `amazon.com` page, which consists of 1 HTML document, 5 javascript, 2 flash, and 65 image objects. In the figure, most off-screen objects are fetched or begin to be fetched before all the 35 on-screen objects are downloaded because of fetching directionality in the table.

We show the effect of the screen contention problem by presenting the simulated object fetching progress in Figure 4. We assume that all the objects are from a single server, and the effect of directionality in a table structure

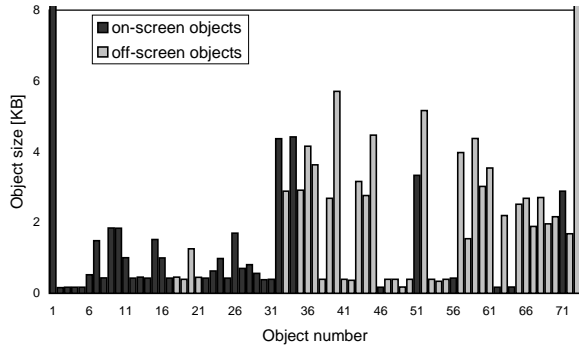


Fig. 3. Object fetching sequence at amazon.com

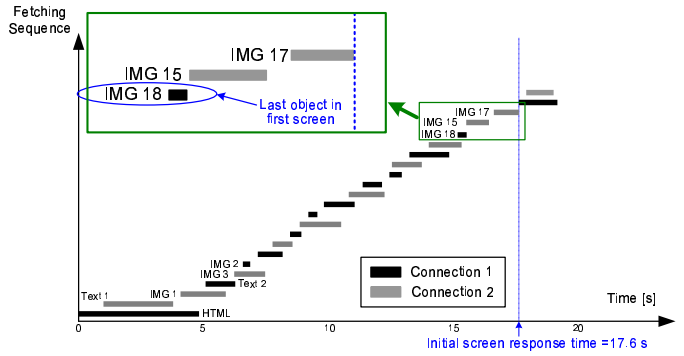


Fig. 4. Fetching without screen contention in ideal web-browsers

is ignored. In the simulations, the initial screen has 18 objects, *i.e.* objects numbered after 18 are unnecessary data. As observed from the figure, objects from both the current screen and other screens are fetched simultaneously due to the screen contention problem. Since fetching the unnecessary objects consumes some portion of bandwidth, the resulting response time for initial screen is increased unnecessarily. As shown in the figure, image (IMG) objects numbered 20 and 22 are fetched in parallel with other objects on the initial screen. As a result, the response time of initial screen, which is measured after IMG 17 is downloaded, is 18.7 s.

An intuitive solution to screen contention is to prevent unnecessary object fetching. Figure 5 shows an ideal case that contention is eliminated in an ideal browser. As seen from the figure, when the faster connection (Connection 1) completes the downloading of all the objects on the initial screen, it stops fetching and waits for the other connection (Connection 2) to finish fetching the objects on the initial screen. By doing so, the remaining connection can obtain more bandwidth and in turn reduces the response time for the initial screen by 1.1 s.

2) *Bandwidth Under-utilization Problem:* In HTTP/1.1, a persistent connection consists of a series of request-response transactions. Given this model, [9] shows that the idle time of the network decreases with the increase in the number of simultaneous TCP connections. The authors of [9] also show that there is an optimal number of simultaneous TCP connections (around 6) at which the performance is optimal because of reduced idle time. However, since current web-browsers do not schedule object transfers in a bandwidth-efficient way across multiple TCP connections, current web transfers do not always maintain the optimal number of simultaneous TCP connections. In many cases, only a small number (e.g. 1 or 2) of connections are active at any instant. This results in under-utilization of the access link which we refer to as the *bandwidth under-utilization* problem.

The above-described bandwidth under-utilization problem results in varying levels of performance degradation depending on specific server scenarios. In the case of a single server, bandwidth efficiency is determined by synchronized ending times of transmission among parallel connections. In Figure 5, the last object, IMG 17, is fetched with no other objects, and thus only a single connection uses network bandwidth toward the end. In cases of multiple servers, the user performance is also affected by synchronized ending among connections to different servers.

The solution to the bandwidth under-utilization problem is to schedule the GET requests across the multiple TCP connections such that all the TCP connections are always active during the fetching process. Figure 6 shows the impact of performing ideal scheduling such that there is a pending request in each of the TCP connection. In the figure, when the faster connection finishes fetching all the objects in its request queue and has no more objects to fetch, it takes over the unfulfilled object requests from the queue of the other connection and perform fetching. As a result, both the connections can use bandwidth more efficiently and improve the initial screen response time by 1.6 s when compared to conventional web-browsers.

Significant amount of web content is being served from distributed web-servers. Ease of content update and server load-balancing are some of the benefits of employing multiple web-servers to serve web content. In the scenario of multiple web-servers, different objects belonging to the same web-page are delivered by opening TCP connections to the different servers. Commercial web-browsers do not take into account the size of the objects in scheduling the

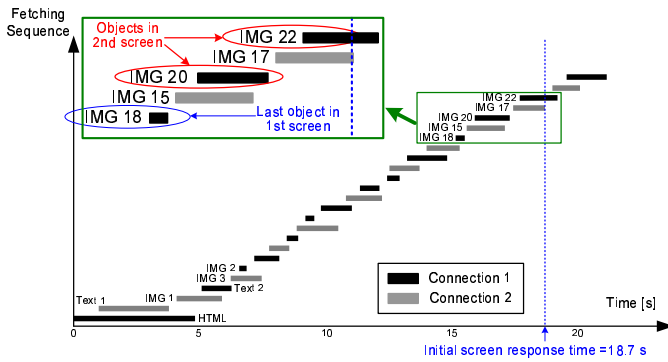


Fig. 5. Fetching with screen contention in conventional web-browsers

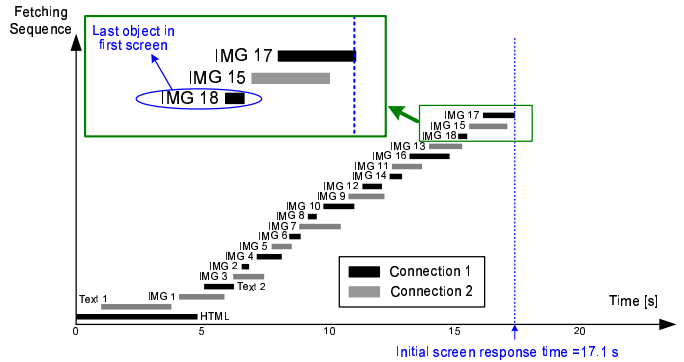


Fig. 6. Fetching with synchronized ending time in ideal web-browsers

different object requests. This invariably leads to scenarios where several TCP connections to different web-servers are idle while the other connections are active. The intuitive solution is to schedule the different object requests across the multiple servers such that all the connections are active.

We have identified two issues with conventional web-browsers in bandwidth-limited networks. We observe that contention among objects belonging to different screens within the same web-page can increase user-perceived response time of the initial screen. We also identify that network bandwidth can be under-utilized because of two issues with conventional browsers. observe that in most cases screen contention and bandwidth under-utilization problems affect user performance negatively in a conventional web model, and show how the ideal browser can overcome these problems and achieve significant performance improvement. Based on these observations, in the next section, we propose a new web access scheme.

### C. Approach

Our proposed solution includes three mechanisms, namely prioritized fetching, objects reordering, and connection management. These three mechanisms complement each other as well as performing optimization with different levels of granularity for web object fetching on the web-browser side. One of the advantages of our solution is easy deployment, since it requires only client-side modification. In fact, the solution can be implemented as nothing more than an add-on to the current web-browsers.

1) *Prioritized Fetching (PF)*: PF differentiates objects from different screens based on the current screen view and allows for downloading only the objects that are required to render the current screen display. As a result, it reduces the response time experienced by web users. The basic operation of PF is as following: (1)When a web access is performed, PF first obtains the initial screen view information in the entire document layout and prioritizes embedded objects according to their locations in the document layout. (2)Then, it performs fetching objects according to their priority levels. (3)When a user scrolls to move to a different view, PF performs the above-mentioned process again for the new screen. PF consists of the following three components.

- *Initial Object Prioritization*: In order to get the location information of objects, PF scans the document object model (DOM) tree [10]. When it finds an IMG object definition, it searches all the successors in the tree and calculates location offsets from successors to predecessors in a recursive fashion until it reaches the top of the tree. The absolute location in the layout is defined as the sum of all the relative offset. Based on this location information, PF gives highest priority level to objects that are located within the current view in the client area and low priority levels to others.
- *Selective Object Fetching*: PF uses a delayed-transmission scheme. When information of new objects are extracted from a HTML document and they are prioritized as high level, PF inserts the corresponding request messages into the already-in-use queues. In this scheme, low-priority objects are fetched only after all the higher-priority objects have been downloaded, i.e. after the higher-priority queues become empty.

- *Re-prioritization*: When the screen focus is moved to a new area, PF removes all the IMG objects that reside in request queues and re-prioritizes them for the newly focused area. For the objects that are currently being downloaded, PF waits for their completeness. The reason for allowing this off-screen fetching is that most web-browsers, as applications above transport layer following HTTP standards, do not have mechanisms to manage disconnections and re-connections. PF thus keeps the currently incoming transfer and only updates the priority levels of the queues involved.

2) *Objects Reordering (OR)*: For parallel connections to a single server, OR uses balanced ordering of objects to gain benefits in terms of reduced response time. The operations of OR consist of three steps. (1) At the first step, an initial assignment of objects is executed. (2) Then, an optimized ordering of objects is performed by TCP-aware object reordering. (3) After that, it performs dynamic objects rescheduling until all objects are completely fetched. OR consists of the following three components.

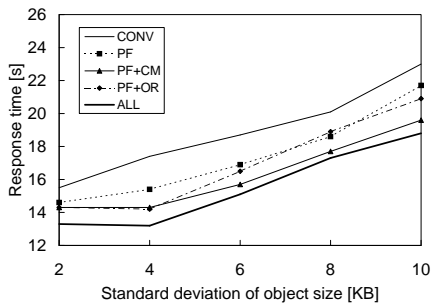
- *Initial Objects Assignment*: Performing OR requires the byte-size information of objects. Since this information is normally not included in HTML documents, OR estimates it by considering both the object's pixel-size included in HTML documents and the object formats such as gif and jpeg. Based on this data size information, OR sends object requests through multiple connections in a balanced way. A time with  $\phi$  expiration value is used to strike the balance between amount of objects and increased response time.
- *Dynamic Objects Rescheduling*: Irrespective of initially balanced assignment of objects among connections, due to dynamic behavior of connections, the total fetching time of different connections may still vary significantly. If due to some reasons one connection is delayed, and the other connection is idle, it is possible to reschedule the objects from the busy connection to the idle one, and thus reduce response time even more. Dynamic Objects Rescheduling runs in an on-demand fashion during the fetching process in order to deal with the dynamic nature of the connections.
- *TCP-aware Objects Reordering*: OR orders the fetching sequence in a *rats-elephants-rats* fashion. First, all the objects assigned to one connection are sorted according to their data size. After that, from smallest one, all objects are inserted from two ends of the queue in round-robin way, and the resulting ordering is a small-to-big-to-small order.

3) *Connection Management (CM)*: CM addresses the bandwidth under-utilization problem when fetching objects from multiple servers by controlling the numbers of connections a browser can open to different servers. By adjusting the number of connections for each server, CM effectively synchronizes the ending time of downloads in the connections. As a result of the improved bandwidth utilization, the response time is reduced. CM consists of the following two components.

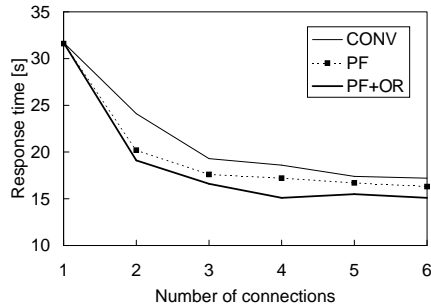
- *Per-Connection Load Estimation*: In order to estimate the ending time of downloading, CM uses the byte-size information that OR converted earlier. The intuition of CM is to assign more connections to servers with larger data size, while assign less connections to servers with smaller data size. To maintain friendliness to current browsers and compatibility to published standards, the total number of connections in our mechanism is maintained the same as in today's popular browsers. By doing so, CM behaves *friendly* to them. To achieve this purpose, whenever it assigns one more connection to some server, one less connection should be deducted from some other server. Furthermore, CM limits the maximum number of connections assigned to a server.
- *Dynamic Connection Assignment*: Whenever OR detects new object information, it estimates the byte-size of this object and starts a timer with  $\delta$  expiration value. After the expiration of this timer, CM performs the initial assignment based on object information collected so far. During the process of fetching objects, it keeps recording the object information on how much data already received. This information will be used again to adjust the number of connections

#### D. Key Results

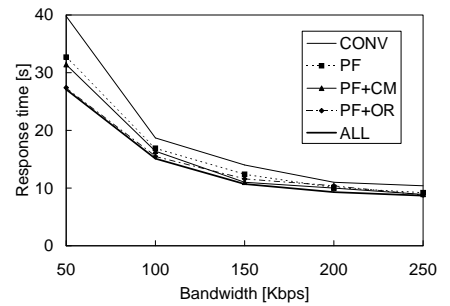
In order to evaluate the performances, we use *ns2* simulator [7]. We assume that the local DNS server has all the required domain information. Response time for the initial screen is used as the primary metric for comparing performances. In this section, we compare five schemes including conventional (CONV), PF only (PF), PF with OR (PF+OR), PF with CM (PF+CM), and all integrated (ALL) schemes.



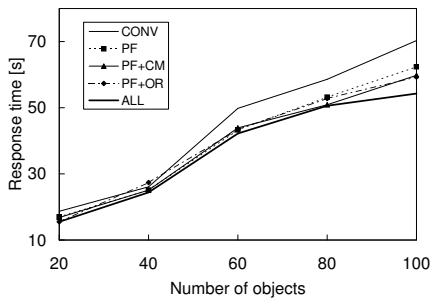
(a) Impact of variance of object size



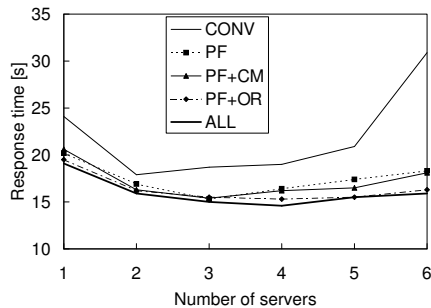
(a) Impact of number of connections



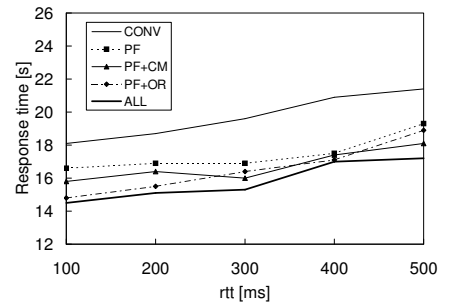
(a) Impact of bandwidth



(b) Impact of number of objects



(b) Impact of number of servers



(b) Impact of round-trip time

Fig. 7. Impact of object characteristics

Fig. 8. Impact of numbers of connections and servers

Fig. 9. Impact of network characteristics

1) *Impact of Object Characteristics:* Figure 7 shows the initial response time of conventional web-browsers, and our proposed solution when the standard deviation of individual object size and the number of objects in the first screen vary. As shown in the figure, when used in combination, the three proposed mechanisms can reduce up to 30% of response time compared to current browsers.

Figure 7(a) show that as the variance of object size increase, the performance of both the conventional model and our scheme shows worse performance. For conventional web-browsers, the reason is obvious, since larger variance can be translated to reduced bandwidth utilization. Our mechanisms can alleviate this problem, and thus reduce the initial response time. However, since the problem still exists, and becomes more severe when variance of object size increases, the performance degradation is still expected.

Figure 7(b) shows the performance differences between conventional web-browsers and proposed ones when the total number of objects increases. Two trends are shown in the figure. As more objects are included in a web-page, first, larger response time is expected; second, the response time reduced by the proposed solution is larger since all of the three mechanisms can gain more benefits.

2) *Impact of Number of Connections and Servers:* Figure 8(a) shows the impact of number of connections to a single server, and it can be seen that up to 20% of response time can be reduced by using our solution. In the figure, as the number of connections to a single server increases, both the conventional and our solution has smaller response time. However, as this number exceeds 4, there are no obvious performance improvements with more connections. This result is consistent with the results presented in other works [9].

Figure 8(b) shows how the initial response time varies as the number of servers for a web-page increases. Two observations can be made from the figure. Increasing number of servers does not necessarily always result in better performance for both conventional browsers and proposed ones. Second, with more servers, our solution can achieve more improvements compared to conventional web-browsers.

3) *Impact of Network Characteristics:* Figure 9(a) shows how the initial response time changes under varying bottleneck bandwidth. As shown in the figure, our solution brings more performance improvement for smaller

bandwidth. It is because of the fact that smaller bandwidth makes the screen contention problem more severe, and thus our solution can reduce response time more by alleviating this problem.

Figure 9(b) shows how the initial response time is affected by the rtt values. Since the major effects of rtt come from the request-response behavior of HTTP protocols (i.e. each object is fetched upon the request from the web client, and thus takes at least one rtt to fetch one object) and our solution can alleviate this effect by removing some of these rtt's required, our solution sees better performance. As shown in the figure, around 20% performance improvement is achieved by our solution under the rtt values considered in the evaluation.

### E. Summary

In this report, we first explore the reasons why conventional web access models are not appropriate for low-bandwidth hosts. We identify two reasons, screen contentions and bandwidth under-utilization, which results in large user-perceived response time. To address this problem, we propose a new web access scheme for low-bandwidth hosts such as wireless clients. The proposed scheme uses an intelligent mix of prioritized fetching, object reordering, and connection management. Using simulations with the web parameters obtained from the top web sites, we evaluate the performance of our scheme and prove its benefits over conventional web access models.

We will publish the client-side web acceleration concept in IEEE International Conference on Broadband Communications, Networks, and Systems [5].

## V. SUMMARY AND FUTURE WORK

In this report, we have presented new principles for enhancing real-time and non-realtime traffic, and more specifically VoIP and Web applications. The principles result in significant improvements to end-user experience. While we have presented the principles as standalone transparent approaches, we believe that a realistic long-term strategy is for them to be absorbed into transport layer protocols. Over the next one year, we will focus on the following three aspects of the proposed research: (i) continue on the prototyping of the developed principles; (ii) integrate work done thus far into unified principles for transport layer protocols; and (iii) focus on other application categories such as peer-to-peer applications.

## REFERENCES

- [1] Z. Zhuang, T.-Y. Chang, R. Sivakumar, and A. Velayutham. "A3: Application-Aware Acceleration for Wireless Data Networks," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, USA, September 2006.
- [2] T.-Y. Chang, A. Velayutham, and R. Sivakumar. "Cut-load: Application-unaware content partitioning for web-based information access in wireless data networks," In *Proceedings of IEEE International Conference on Communications (ICC)*, 2006.
- [3] R. Vedantham, S. Kakamanu, S. Lakshmanan, and R. Sivakumar, "Component Based Channel Assignment in Single Radio, Multi-channel Ad Hoc Networks," in *Proceedings of ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, USA, September 2006.
- [4] Y. Jeong, S. Kakumanu, C.-L. Tsao, and R. Sivakumar, "Improving VoIP Call Capacity over IEEE 802.11 Networks," to appear in *Proceedings of IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, Raleigh, NC, USA, September 2007.
- [5] T.-Y. Chang, Z. Zhuang, A. Velayutham, and R. Sivakumar, "Client-Side Web Acceleration for Low-Bandwidth Hosts", to appear in *Proceedings of IEEE International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, Raleigh, NC, USA, September 2007.
- [6] A. Kamerman and L. Monteban, "WaveLAN-II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, Aug. 1997.
- [7] The Network Simulator: *ns2 simulator*, <http://www.isi.edu/nsnam/ns>.
- [8] W. Turin and R. V. Nobelen, "Hidden Markov models for fading channels," *IEEE J. Select. Areas Commun.*, vol. 16, no. 12, pp. 1809–1817, Dec. 1998.
- [9] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt, "Performance Optimizations for Wireless Wide-Area Networks: Comparative Study and Experimental Evaluation," in *Proceedings of ACM Mobicom 2004*, Philadelphia, PA, Sep. 2004.
- [10] W3C Document Object Model, [www.w3.org/DOM/](http://www.w3.org/DOM/).