The Performance of HTTP over Satellite Random Access Channels

Dennis P. Connors, Gregory J. Pottie
University of California, Los Angeles
connors@ee.ucla.edu, pottie@icsl.ucla.edu

September 18, 1999

Abstract

This paper attempts to measure the performance of the Hyper Text Transfer Protocol (HTTP) [8] in a multiple access, satellite environment. The HTTP client's access to the Internet is via a shared satellite uplink channel. The multiple access protocols we have chosen to simulate are Slotted ALOHA [13] and Reservation ALOHA [7]. We have constructed a detailed simulation of "web browsing" behavior using ns-2.0 [1] and have used this to simulate HTTP transfers over random access satellite channels. The purpose of this work is to determine if current medium access techniques are sufficient to accommodate many terrestrial users whose traffic patters are dominated by HTTP client-server behavior.

1 Introduction

Recent years has seen a flurry of activity in the area of internetworking via satellite [10]. Several companies are planning to roll-out satellite networks capable of packet switching at the satellite [21]. The aim of these proposals is to provide broadband Internet access to Small Office / Home Office (SOHO) customers, multi-unit dwellings (MUD), and areas of the globe where wire-line infrastructure is not as plentiful, and to provide a LAN interconnect for companies with dispersed campuses. All of these systems have full duplex links between the earth terminal and satellite. This system model leads to multiple access in the earth terminal to satellite uplink channel. Figure 1 shows this pictorially.

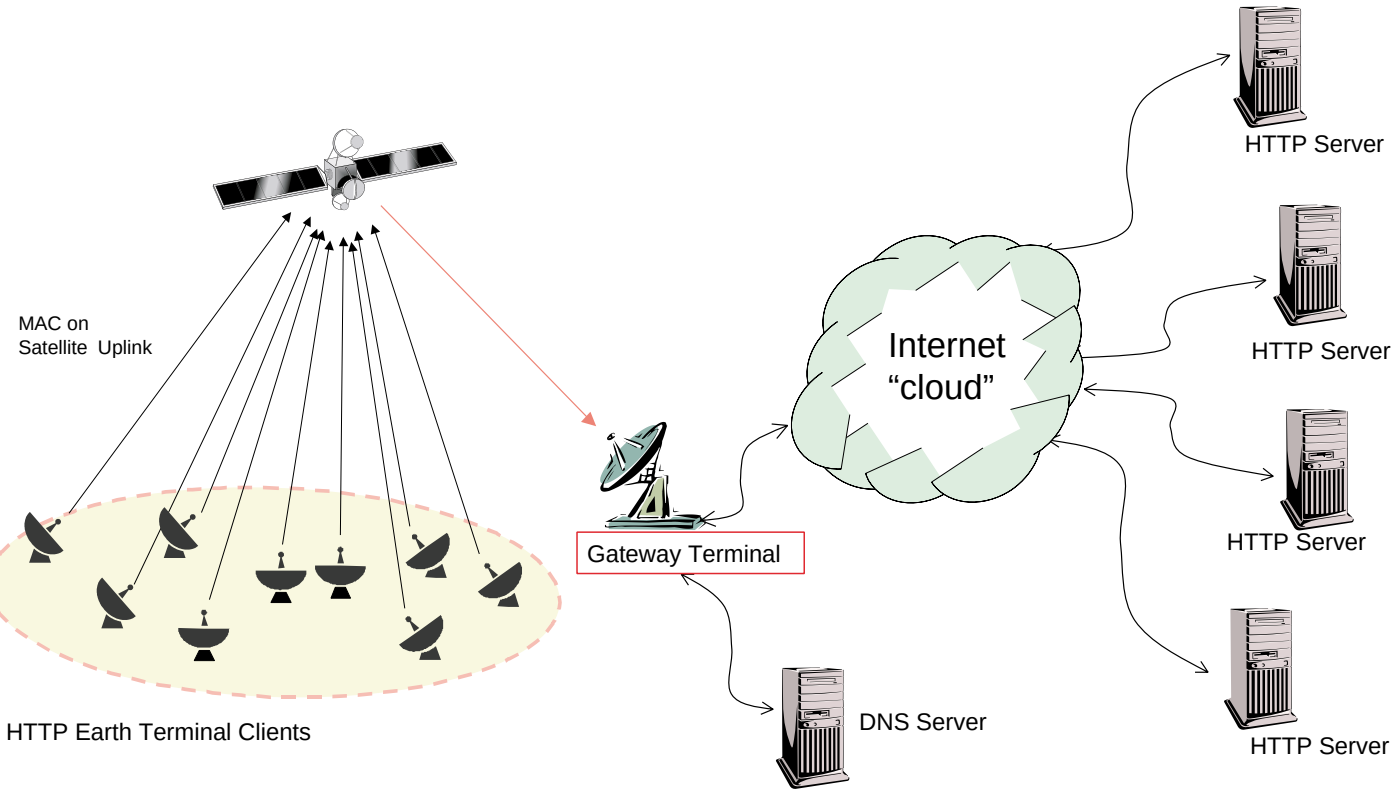

Figure 1: A Satellite Network connecting Clients and Servers through the Internet

Since the traffic patterns many Internet applications are "bursty" in nature, the fact that uplink channel is one-to-many can lead to either underutilization of the channel or to poor end-to-end application performance. Since 1995, the hyper text transfer protocol (HTTP) has been the dominant source of Internet traffic. Although the statistics of Internet traffic are always in a constant state of flux, it is safe to say that the client-server Web browsing model will be a significant source of traffic for years to come. With this assumption in mind we have endevoured to model and simulate the performance of HTTP sessions over multiple access satellite uplink channels, with the goal being to see if current, well known multiple access techniques are sufficient to provide a suitable quality of service (QoS). In order to accomplish this we have performed simulations of Web transfers using the discrete event simulation tool ***ns-2.0*** [1]. Utilizing the built in TCP [18] capabilities of ***ns***, we have implemented both medium access control functionality at the

link layer and a detailed model of Web browsing at the application layer. This enabled us to perform simulations of HTTP World Wide Web transfers over two multiple access techniques, Slotted ALOHA [13] and Reservation ALOHA [7], with a variety of satellite round trip latencies. The outline of the remainder of this paper is as follows, we will first provide a brief outline of the challenges in MAC protocol design for satellite channels, this will be followed by a description of the HTTP "process" that we are modeling, we will then present our simulation methodology followed by results with analysis. We will conclude with some thoughts for future work.

2 Medium Access Control for Satellite Channels

The goal when designing medium access control (MAC) protocols has always been to maximize the utilization of the shared channel while at the same time providing a suitable quality of service, in terms of packet loss, delay, and jitter, to the users accessing the channel. Medium access techniques can be crudely categorized into three access methods; random access, fixed access, and demand assigned access. In random access, the dispersed users sharing the channel transmit packets in an uncoordinated manner to the arbitrating agent (in this case the satellite). The ALOHA protocol [2] and its variants fall into this category. Since collision-free channel resources cannot be guaranteed with random access methods, QoS guarantees, in terms of packet loss and delay, are very weak. The benefits of random, uncoordinated transmission are reduced control signaling and algorithmic overhead, and in ease of implementation. Random access MAC techniques are traditionally employed when the network traffic is unpredictable and bursty. Fixed access lies at the other end of the medium access spectrum. In fixed access, each channel user is given dedicated bandwidth for the life of the connection. The traditional mobile cellular access techniques TDMA, FDMA, and CDMA fall into this category. The QoS guarantees that fixed access is capable of delivering are much stronger than random access. This QoS improvement comes however at the expense of poor utilization of the uplink channel, since most user traffic is bursty. This poor link utilization leads to low network capacity, i.e. the number of earth terminals that can be supported within a given amount of uplink bandwidth. Demand assigned multiple access (DAMA) seeks to achieve the QoS guarantees of fixed access while achieving the statistical multiplexing gain of random access [6]. Most DAMA variants involve the remote user transmitting a metric to the arbitrating agent (AA) which represents its instantaneous bandwidth needs. The transmission of this DAMA metric is traditionally done in a random access manner. The AA then allocates collision free bandwidth in proportion to the requested amount. This can be thought of as logically splitting the channel into a random access and fixed access region. The main drawback to DAMA techniques is the delay incurred by having to perform a request "handshake" with the AA. For satellite channels, this delay is significant. Since we are simulating HTTP Web transfers, the source traffic emerging from an earth terminal will be quite bursty, so clearly using fixed access will yield very poor network capacity. In considering DAMA, one needs to determine the distribution of packet sizes that will be transmitted in the uplink channel. Demand assigned multiple access works well for bursty sources only if the sizes of the data packets far exceed the size of the DAMA request packets. If this is not the case, then DAMA performs no better than random access. Studies have shown a strongly bi-modal distribution of TCP packet sizes [9][4] due mainly to the fact that most TCP packets are either data carrying packets, which are close to the maximum transmit unit (MTU) size, or acknowledgement packets, which tend to be less that 100 bytes. Since the uplink channel will be used by the HTTP client, and since the bulk of the data will be flowing in the downlink channel (i.e. from server to client), the majority of packets that will be transmitted in the uplink channel will be acknowledgement packets. For this reason and the fact that satellite channels have high latencies we have eliminated traditional DAMA techniques from consideration and focused on two random access methods, Slotted and Reservation ALOHA [13][7]. These two methods of multiple access will be described in more

detail in the Section 4, since implementations of them vary.

3 Web Browsing Statistics and the Hyper Text Transfer Protocol

Clients retrieve web documents from remote servers using the Hyper Text Transfer Protocol [8]. The mechanics of this retrieval are as follows. The client passes a universal resource locator (URL) message to a domain name server (DNS). In a satellite network this DNS will usually reside at a gateway node. The DNS replies with an Internet Protocol (IP) address of the server which contains the Web page the client desires. This transaction is usually done using the user datagram protocol (UDP) [17]. The client then issues an HTTP GET message to the server, with the name of the desired file. The server responds with the desired file. This transaction occurs using TCP. Often there are many inline objects in the requested file (i.e. attached gif images). Once the client detects the presence of these objects, sends HTTP GET messages for each object. The mechanics of these "secondary" retrievals are what separate the various HTTP implementations. In the oldest versions, the secondary retrievals occurred in serial only after the initially requested (primary) file was completely received. This meant that there was only one active TCP connection at any given time during the transaction. Netscape version 3.0 and before performs Web transfers using a different technique called parallel connections. In this case up to (usually) four TCP connections were used to retrieve additional inline objects. In this case, if there were more that four objects to retrieve, the HTTP GET messages would be queued until one of the existing TCP connections was closed. This method was replaced by persistent connections (post Netscape 3.0 and others), in which when on of the parallel TCP connections became idle, instead of closing the connection and restarting a new connection, the existing connection was re-used. This technique reduced the time to transfer files since the TCP connection establishment "handshake" would not need to be repeated and more importantly, the TCP slow start mechanism could be avoided [11]. Since individual Web transfers tend to be small, avoiding slow start would aid in reducing the transfer time. Recently HTTP 1.1 has been proposed [8]. HTTP 1.1 uses only a single TCP connection and pipelines HTTP GET messages. Using this method, when the client discovers an inline object, it simply issues the HTTP GET on the existing ongoing TCP connection. Baring network congestion, a client using HTTP 1.1 will only experience slow start once. For purposes of our simulation, we have only considered HTTP 1.1 since it is anticipated to be the dominant form of HTTP in the coming years. However for completeness sake we will consider the other versions of HTTP (single, parallel, and persistent) in future work.

In order to drive a simulated HTTP session, the statistics of the Web transfer process must be know with a reasonable degree of accuracy. For the purposes of our simulation, we have relied on the measurements taken in [14]. In this work, the HTTP packets arriving to and leaving from a Web server at the University of California at Berkeley were collected. This data was then processed to produce a set of cumulative distribution functions (CDFs). The statistics relevant to our simulation work were: client think time, HTTP GET length, HTTP REPLY length, inline object count, and consecutive file retrieval count. The client "think" time measures the time between successive initiations of a Web transfer. This represents the time the user needs to examine the Web page and decide retrieve another. The HTTP GET length measures the size of the individual GET messages. The HTTP REPLY length measures the size of the individual requested pages or inline objects. The inline object count records the number of inline objects requested per Web transfer and the consecutive file retrieval count records the number of successive "hits" to the same Web server. The average of all of these statistics is shown in Table 1. For a more complete description of these measurements, please refer to [14].

These CDF's described in [14] have been integrated into the network simulator ***ns***. Using these allows us to simulate HTTP sessions over satellite channels using the empirical measurements. We have chosen to

Table 1: Summary of HTTP Statistics

Parameter	ThinkTime	GET	REPLY	Objects	Consecutive Hits
Avg.	1313	398	17932	2.9	4.1
Units	seconds	bytes	bytes	none	none

use all of the above statistics except for one, that being the user think time. The CDF of the user think time in [14] shows a heavy tail and an average value of 1313 seconds, or approximately 22 minutes between successive Web transfers. These numbers intuitively make sense, however since we are interested is system capacity assessments, we would be interested in a "busy" Web client. We therefore have chosen the user think time to be exponentially distributed with a mean of 60 seconds.

4 Simulation Methodology

We now describe our satellite simulation test-bed. Figure 1 shows how the earth terminals are connected with remote Web servers. The HTTP clients are connected to the individual earth terminals, one client per terminal. This emulates the "personal earth terminal" model outlined in [6]. All earth terminals share a common uplink channel. The MAC protocol in place on the uplink channel is either S-ALHOA or R-ALOHA. The DNS server is located at the terrestrial gateway node, one satellite hop away from each earth terminal. Since [14] has shown approximately a 20:1 ratio between the bytes sent by the server to bytes sent by the client, we have set the downlink bandwidth (i.e. the channel from satellite to earth terminals) to be 20Mb/s and the uplink to be 1Mb/s. This will allow our simulation results to predict the number of terrestrial users that can be supported per 1Mb/s of uplink bandwidth. The delay across the satellite channel was chosen to be 7.5ms, 30ms, and 135ms. This allows us to measure the performance in low earth orbit (LEO), medium earth orbit (MEO), and geosynchronous earth orbit (GEO) scenarios. Please note that we are not considering the movement and hand over of satellites that will be present in LEO and MEO systems. We are fairly certain that since packet level medium access operates at much smaller time scales that inter-satellite handover, the performance results we show in this paper will be unaffected by satellite mobility, however further simulation work that incorporates mobility may be needed. Packets that are successfully received at the satellite are then transmitted in the downlink channel to the gateway node. The gateway node then forwards the packets across the Internet "cloud" to the appropriate web server. Packets will experience random delays as they make their way to and from the server. The method for emulating end-to-end Internet delays will be described in detail in Section 4.4. The client measures the time from the start of the HTTP transaction until its completion. This measurement will serve to quantify performance of the two MAC schemes. This entire simulated network was implemented using ***ns-2.0*** a freely available network simulation tool [1]. Since ***ns*** is principly designed to simulate network behavior from the network layer to the application layer, we were required to add MAC layer functionality [5]. We also added the necessary code to accurately model HTTP 1.1 mechanics. With this simulation tool in place we experimented with Web transfers using both Slotted and Reservation ALOHA.

4.1 Implementation of S-ALOHA

Slotted ALOHA is a time synchronized version of the ALOHA protocol first proposed in [12]. In our simulation, we implemented it in the following manner. When a packet arrives at the MAC layer of an earth terminal, it is transmitted immediately. After transmission, a timer is set that will expire one round trip time (RTT) in the future. A RTT is twice the delay between the earth terminals and the satellite. If the packet arrives successfully at the satellite, the satellite sends an acknowledgement packet in the downlink channel. When the user receives the acknowledgement, it cancels the timer and considers the packet successfully received. If a collision occurs on the channel, no packet is received at the satellite. When the RTT timer expires, the terminal enters the *backoff* state. It chooses a random backoff time according to either an **linear** or **exponential** model. In the linear case, the backoff time is chosen according to $\mathcal{E}(i\lambda)$, where $\mathcal{E}(\xi)$ is an exponential random variable with mean ξ and i is the number of successive collisions. For the exponential case, the backoff time is chosen according to $\mathcal{E}(2^{i-1}\lambda)$. These two techniques mean that for each sequential collision, the variable i is incremented and then a random backoff time is chosen according to either the linear exponential manner. When the packet is successfully received at the satellite and an acknowledgement is received, i is reset to one. The difference between these two backoff strategies is that for the linear method, the mean backoff time grows linearly with each successive collision, and in the exponential case, it grows exponentially. Figure 2 shows a state diagram of how S-ALOHA was implemented for the purposes of our simulation.

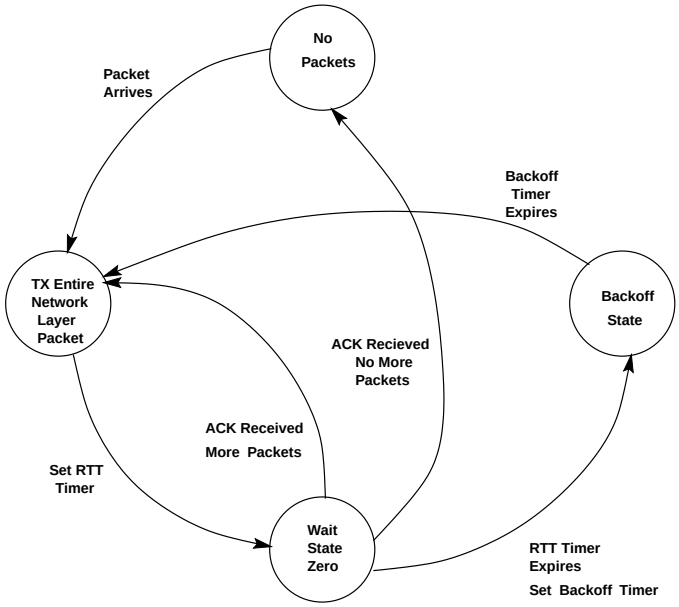

Figure 2: State Transition Diagram for S-ALOHA

4.2 Implementation of R-ALOHA

Reservation ALOHA was first proposed in [7]. It sought to achieve the simplicity of use that ALOHA offers while attempting to implicitly reserve bandwidth to improve on throughput. Our implementation of

R-ALOHA works in the following manner. The satellite is constantly broadcasting who "owns" slots in the uplink channel. If a earth terminal successfully transmits a packet in an uplink slot, that terminal owns that slot in the subsequent time division multiplexed (TDM) frame. A TDM frame therefore is equal to 1 RTT. Packets arriving at the earth terminal are placed in the *outgoing* queue. When the next slot interval arrives, the slot can be classified as one of three states; ***EMPTY***, ***MINE***, or ***OTHER***. ***EMPTY*** means that in the previous frame, either no earth terminal attempted transmission or more than one did resulting in a collision. ***MINE*** means that the slot belongs to this particular earth terminal due to a successful transmission in the previous TDM frame. ***OTHER*** means that the slot is reserved by another terminal. Given that there are packets to transmit, the earth terminal chooses to transmit with the following contention probability P_c.

$$P_c = \begin{cases} 1 & if \quad \text{slot} = \text{MINE} \\ 0 & if \quad \text{slot} = \text{OTHER} \\ 1 - e^{\alpha \cdot \text{QL}} & if \quad \text{slot} = \text{EMPTY} \end{cases} \tag{1}$$

where:

QL = the size (in bytes) of both the *outgoing* and *retransmission* queues
α = the aggressiveness parameter

This method insures that when a terminal has packets to send, it will always send if the slot belongs to him, never if it belongs to another terminal, and if a slot is empty, it will content with a probability that grows with the number of packets queued at the terminal. The slot ownership broadcast from the satellite serves as an implicit acknowledgement. If a terminal transmits a packet on an empty slot, and one RTT later that slot is described by the satellite as ***EMPTY***, the terminal knows that the packet collided and it places the packet in the *retransmission* queue. The state diagram for our implementation of R-ALOHA is shown in Figure 3.

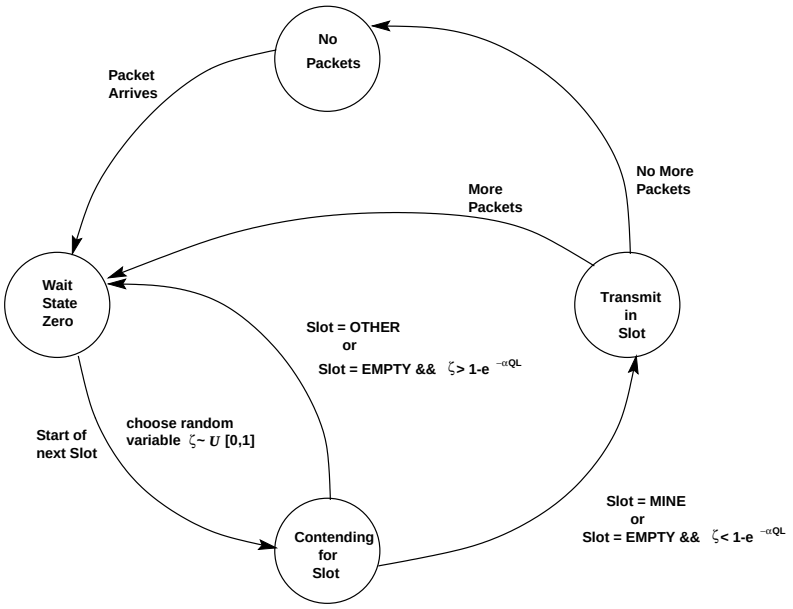

Figure 3: State Transition Diagram for R-ALOHA

4.3 Mechanics of a Simulated Web Transfer

The procedure for a HTTP transaction follows exactly the means outlines in Section 3. When a HTTP client emerges from thinking, it issues a DNS message to the DNS server located at the gateway node. The gateway node responds with the desired IP address. The client then initiates a TCP connection with the server using the TCP connection establishment procedure (i.e. SYN, SYN+ACK, ACK) described in [20]. Once the connection is established the client sends the initial HTTP GET message. The size of this GET message is distributed according to [14]. The server responds with a REPLY, also distributed according to the measurements in [14]. This procedure continues until the initial Web page and any inline images are completely transferred, according to the procedure of HTTP 1.1. This is shown graphically in Figure 4.

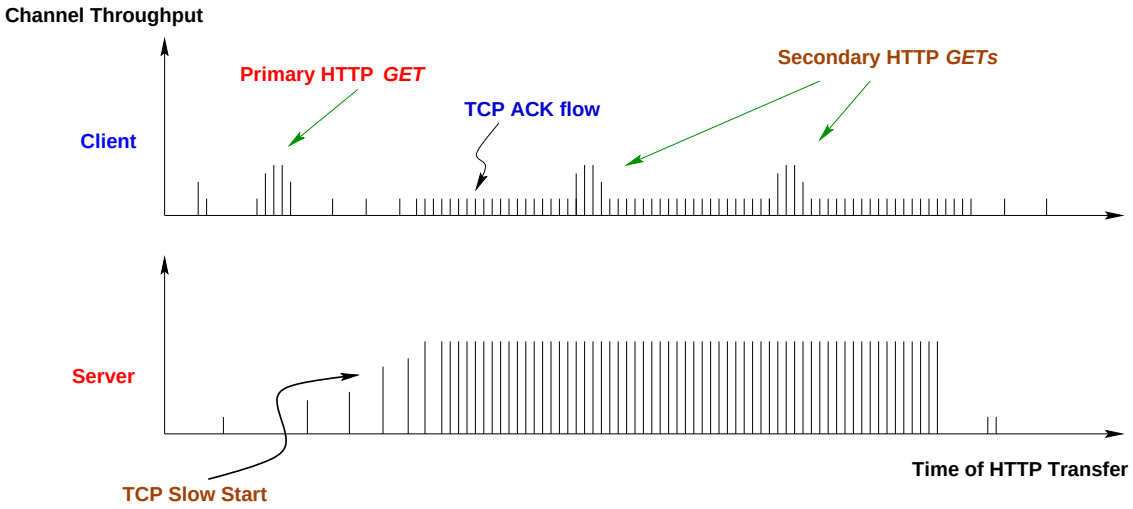

Figure 4: The WWW Page Transfer Process Illustrated

In order to mimic sequential "hits" from a client to a server, when a server is first accessed an integer random variable, h, is chosen according to the CDF of consecutive file retrievals presented in [14]. Every time the client emerges from the thinking state, h is decremented. While h is greater than zero, no DNS lookup is performed prior to TCP connection establishment. Once h reaches zero, a DNS lookup is performed for the next Web transfer and h is randomly re-chosen.

4.4 End-to-End Internet Path Emulation

The primary aim of this simulation is to measure the effect of the satellite medium access control protocol on HTTP performance, but since packets crossing and end-to-end connection will experience delays due to the switches and routers that make up the Internet, we needed to model, in a somewhat abstract manner, this interaction. Following the claim made in [16], we chose to model the round trip times of the packet crossing the Internet "cloud" of Figure 1 as a shifted gamma distribution. A gamma(α,β) random variable has the following distribution.

$$f(x) = \begin{cases} \frac{\beta^{-\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}}{\Gamma(\alpha)} & if \quad x > 0 \\ 0 & otherwise \end{cases} \tag{2}$$

where $\Gamma(\alpha)$ is the gamma function, defined by $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$.

In order to parameterize this distribution, we conducted a series of `ping` [19] measurements on hosts at various distances. Table 2 shows the statistics of each host and the two parameters, α and β, that made the round trip time statistics match a gamma distribution. Using these results we implemented and Internet Path Emulator (IPE) that worked in the following manner. Each packet sent from a client has a flow identification, f_{id}^c. When the consecutive hit variable h described in the previous section reaches zero, the client increments f_{id}^c. This signals the IPE that the client is accessing a different server at a different distance (in terms of hops) from the gateway node. The IPE then chooses a new minimum round trip time m_{id}, and new gamma parameters α_{id} and β_{id}. Thus when packet p_i with flow id f_{id}^c enters the IPE, it sees a random delay d_i, where d_i is chosen as follows.

$$d_i = m_{id} + \nu_{(id,i)} \tag{3}$$

where $\nu_{(id,i)} \sim$ gamma(α_{id}, β_{id})

Consistent with Table 2, for each new flow id we chose m_{id} uniformly between 25ms and 150ms, α_{id} uniformly between 1.1 and 2.0, and β_{id} uniformly between 1.5 and 17.

Table 2: Summary of `ping` Host Statistics

Host Name	RTT Min (mSec)	RTT Max (mSec)	RTT Avg (mSec)	α	β
ees2cy.engr.ccny.cuny.edu	90	415	124	2	16.67
utds05.utmem.edu	143	582	162	2	2.5
www2.ece.uiuc.edu	97	132	102	2	2.5
web.cs.ndsu.NoDak.edu	77	178	95	1.5	14.3
info.tamu.edu	75	236	81	2	2
patch.Colorado.EDU	43	245	45	2	1.4
charlotte.it.wsu.edu	51	167	63	2	3.33
info1.ucdavis.edu	26	281	30	2	2.5
www.mit.edu	93	266	103	2	2.5
www.ee.ucla.edu	29	251	31	2	1.43
vole.eng.fsu.edu	90	207	101	1.5	6.66
rouge.engr.wisc.edu	97	192	102	1.1	3.33
www.umf.maine.edu	95	331	103	1.1	2
ece00ws.ece.ncsu.edu	88	369	90	1.5	3.33

5 Simulation Results

In this section, we will present the extensive simulation work conducted using our modified version of ***ns-2.0***. Before beginning, all of the relevant physical layer statistics are summarized in Table 3. The first thing to determine was for S-ALOHA, which backoff strategy, either **linear** or **exponential** was superior. Figure 5 shows the performance of both techniques in a GEO (i.e. 250ms round trip time) channel, under heavy loading. The x-axis shows the web page size (i.e. the size of the entire HTTP transfer) and the y-axis shows the average time to completely transfer the file. To determine what constitutes a heavy load, we applied the following calculus. According to [14], the average bytes transferred from a server to a client during a typical Web download was $2.9 \cdot 398 = 1,154$ bytes. Assuming that a transaction occurs on average of every 60 seconds, this means a client has an average bit rate of $\frac{1154 \cdot 8bits}{60sec} = 154bps$. This is not taking into account the TCP acknowledgements that will be flowing in the uplink (which is non-negligible). It is well known that the Slotted ALOHA channel has a throughput of 0.36, then taking a 1Mb/s channel at 36% throughput gives us 360 kb/s. Dividing 360 kb/s by 154 b/s, gives us 2338 supported users. Thus to simulate a heavy load, the active user count was set to 3200 HTTP clients per 1 Mb/s of uplink bandwidth. Figure 5 clearly shows that for a GEO channel, an exponential backoff strategy vastly outperforms a linear strategy. For this reason, we have eliminated the linear backoff strategy, and focused only on exponential.

Table 3: Summary of Physical Layer Statistics

Parameter	Value
Uplink Bandwidth	1 Mb/s
Downlink Bandwidth	20 Mb/s
TDMA Frame Length	6.4 mSec
TDMA Slot Length	1.6 mSec
PHY Layer Cell Size	53 bytes
Cells per TDMA Slot	4
Slots per TDMA Frame	4
GEO RTT Delay	250 mSec
MEO RTT Delay	60 mSec
LEO RTT Delay	15 mSec

The next issue for S-ALOHA, is that given an exponential backoff technique is to be employed, what is the optimal initial backoff "seed" λ, mentioned in Section 4.1. It would be reasonable to think that this λ should increase with increasing round trip latencies, since collisions are more costly as the time to acknowledge a collision increases. We therefore conducted a series of experiments, again under the heavy loading condition of 3200 active users, for LEO, MEO, and GEO channels, with initial backoff seeds, λ, of [2,4,6,8,10], [6,8,10,12,14], and [10,12,14,16,18] slots respectively. Figure 6 shows the results. It would appear that in at least an average sense, the initial backoff seed is not as important as backoff strategy, since there appears little difference between the performances for all three channels. Therefore we have chosen the minimum seeds, LEO = 2 slots, MEO = 6 slots, GEO = 10 slots, for all subsequent simulations. The reasoning behind this is that choosing a smaller initial backoff seed will reduce backoff time and increase throughput. If this is aggressive approach is shown to not suffer at high loads, then it should intuitively work better at lighter loads.

Now we turn our attention to R-ALOHA. The unknown simulation parameter, is the aggressiveness param-

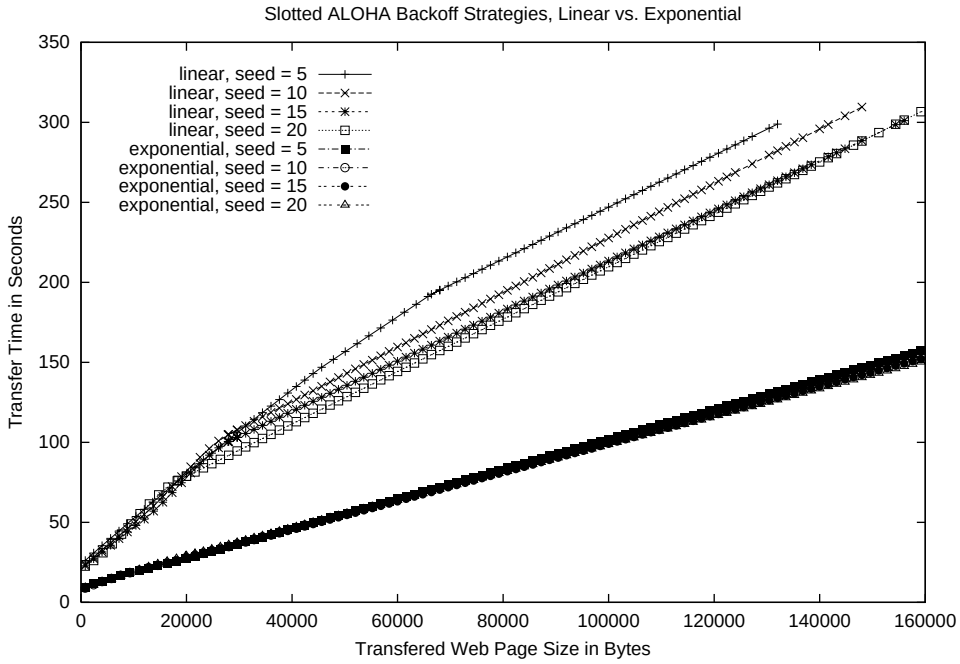

Figure 5: Comparing Linear and Exponential Backoff Strategies in S-ALOHA

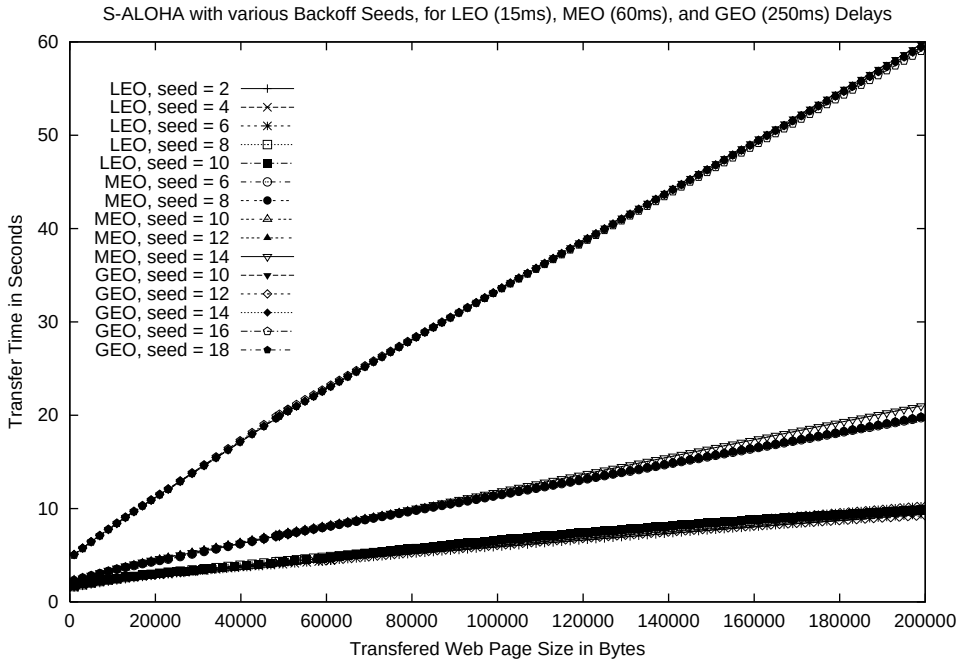

Figure 6: Evaluating the effect of the Initial Backoff Seed in S-ALOHA

eter α. This parameter determines, along with the output queue length, how likely a terminal is to contend for an empty slot. After some "back of the envelope" calculations, we determined to simulate R-ALOHA, over all three channels, with an α that varied over an order of magnitude, 0.01 - 0.5. Again, all of these simulations were conducted under heavy network loading. Figures 7, 8, and 9 show the result. For the LEO channel, the optimal α is 0.05, with 0.01 being too timid and 0.1 being too aggressive. An α of 0.5 was not shown since it was too aggressive and produced erratic results. Figure 8 shows very little difference between an α of 0.1 and 0.05 for the MEO channel. We have therefore sided with the more aggressive value of 0.1 for our future simulations. Again, α of 0.5 was not shown for the same reasons as the LEO case. In the GEO channel, a conservative α of 0.01 and a more aggressive value of 0.1 appear to achieve the same performance. This shows that both a greedy and a conservative approach achieve the same performance, in the average sense, under heavy loading. We will again side with the aggressive approach and choose α to be 0.1 under the intuition that with lighter network loads, an aggressive strategy will produce better throughput.

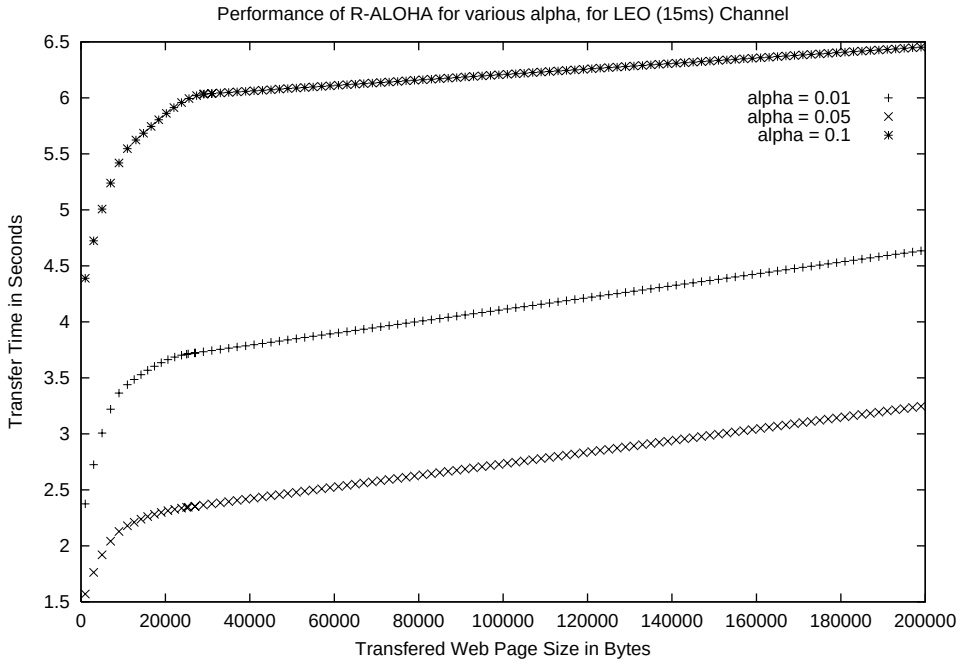

Figure 7: LEO Channel performance with respect to the Aggressiveness Parameter, α

Once the parameters of simulation were established, we conducted a series of simulations, for both MAC protocols, for all three channels, at various loading conditions. Since the initial simulations were conducted at heavy loading, we decided to backoff the offered channel load and see what performance each protocol was capable of delivering. We therefore chose to simulate three additional loading conditions, 400, 800, and 1600 active users, all sharing the 1 Mb/s uplink channel. The results of these simulations are shown in Figures 10 through 15. What is immediately apparent is that R-ALOHA vastly outperforms S-ALOHA, but for completeness sake we will briefly analyze Figures 10, 11, and 12. These figures show literally no difference in download performance as the load on the channel changes. This is most likely due to the fact that S-ALOHA only works well for bursty "exponential" like traffic, and becomes unstable when the traffic pattern becomes more deterministic. Referring back to Figure 4, after a certain amount of time and for sufficiently large web pages, the uplink traffic becomes a stream of TCP **ACKs**, which are deterministic and

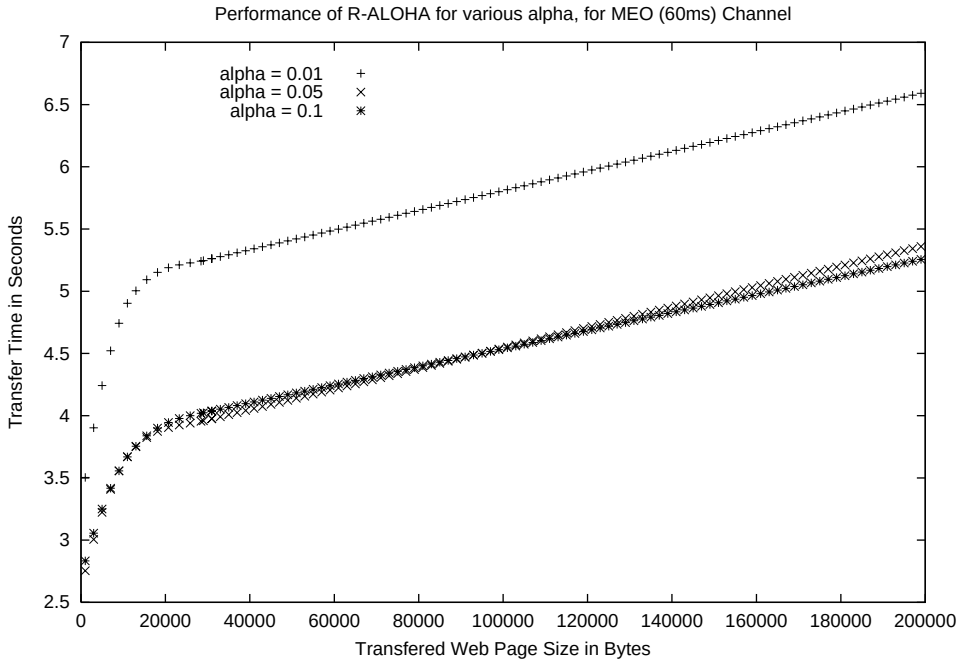

Figure 8: MEO Channel performance with respect to the Aggressiveness Parameter, α

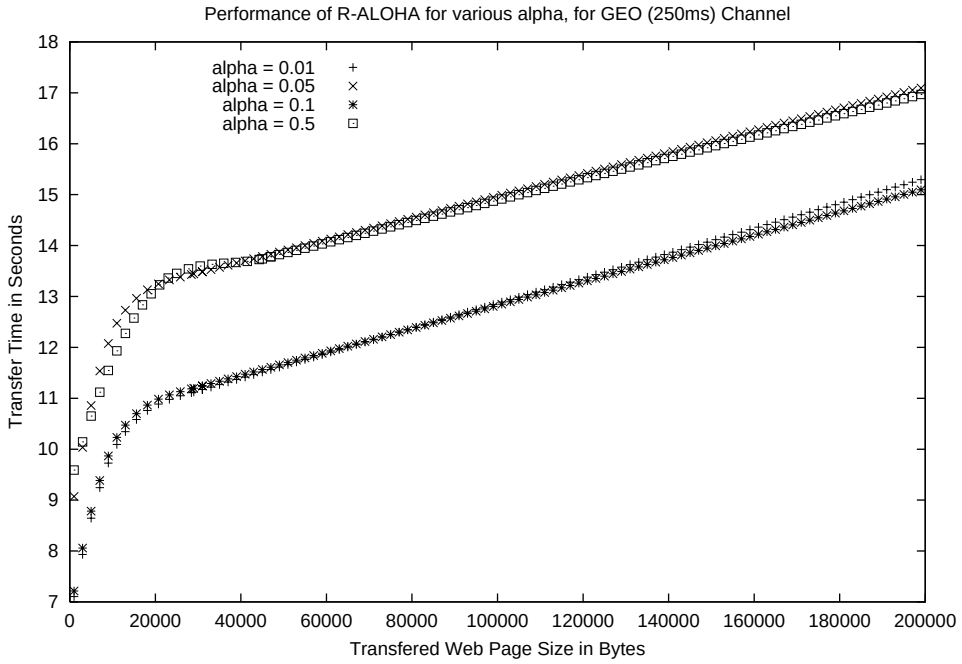

Figure 9: GEO Channel performance with respect to the Aggressiveness Parameter, α

to some degree constant bit rate in nature. When this is the case, an S-ALOHA channel achieves very poor throughput, due mainly to the high collision rate.

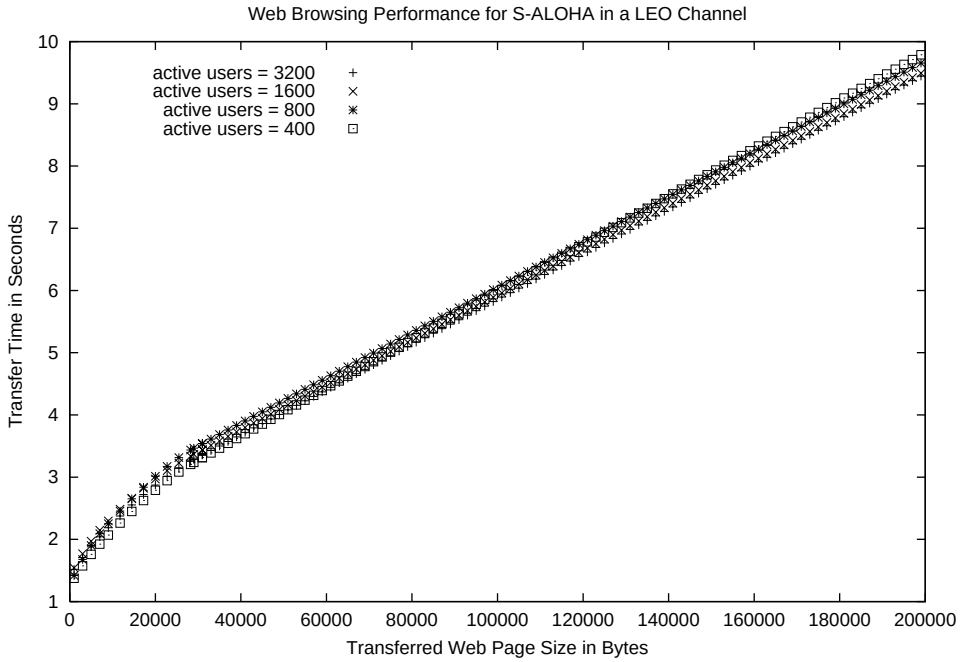

Figure 10: The performance of S-ALOHA for various active HTTP user counts, LEO Channel

Turning our attention now to Figures 13, 14, and 15, we see that much better web page download times can be achieved with R-ALOHA. Figure 13 shows a gradual increase in transfer time as the channel gets more crowded with users. Still however, for LEO like delays, web page download times of well under 4 seconds are being achieved. Figures 14 and 15 show the same general behavior, with download times increasing as channel delay increases. One interesting artifact of both Figure 14 and Figure 15 is that the download time of the 1600 user case exceeds that of the 3200 user case. We believe that this is primarily due to the choice of the aggressiveness parameter α. We decided on the more aggressive value, under the assumption that if the performance at heavy loads was optimal, this would carry over to lighter loads. Apparently this is not the case. Most likely an α of 0.05, like the LEO case, would have been more appropriate.

We contend that the main reason that R-ALOHA outperforms S-ALOHA is that World Wide Web page downloads can be thought of as small to medium size file transfers, from the server to the client. As the TCP congestion window opens, what is most important for achieving good throughput is to have a collision free channel to deliver the TCP ACKs. As ACKs arrive at the server, more packets are sent to the client, increasing throughput and decreasing web page download time. R-ALOHA, due to its implicit reservations, provides a much more reliable channel for these ACKs to reach the satellite. This results in an overall lower transfer time, especially for larger web pages.

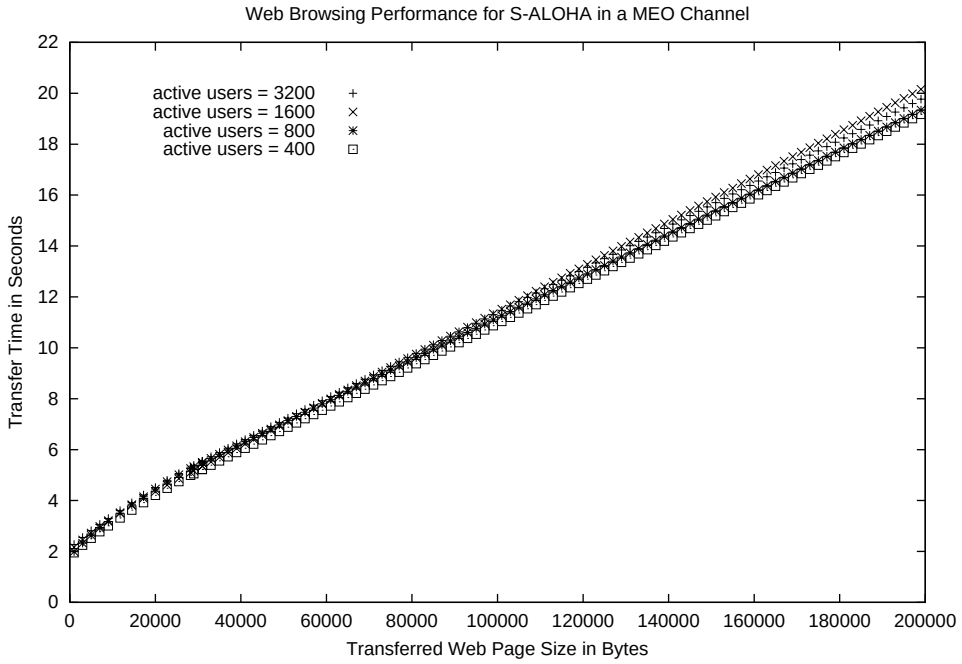

Figure 11: The performance of S-ALOHA for various active HTTP user counts, MEO Channel

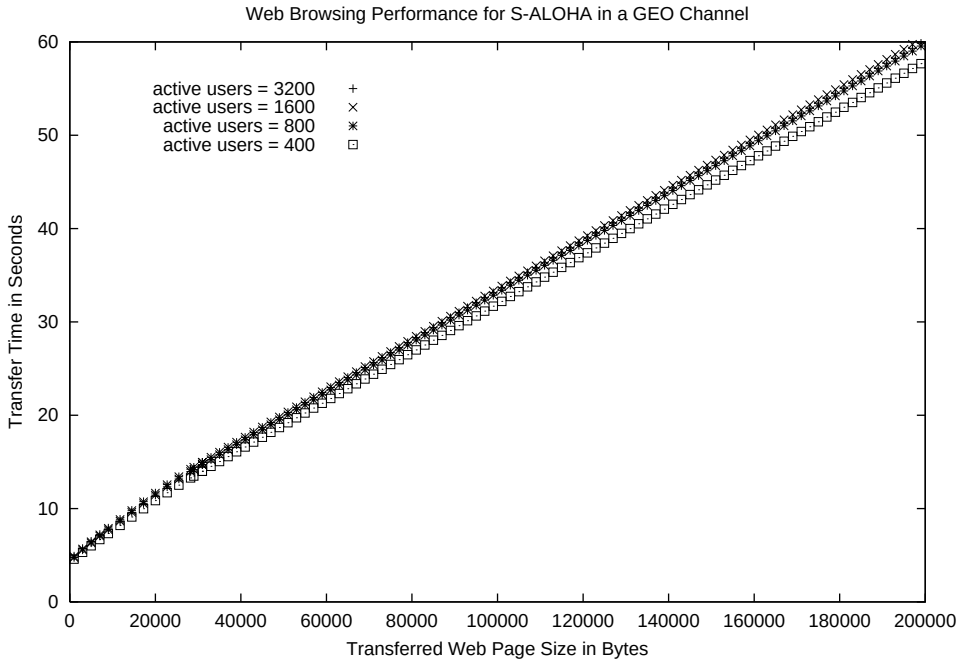

Figure 12: The performance of S-ALOHA for various active HTTP user counts, GEO Channel

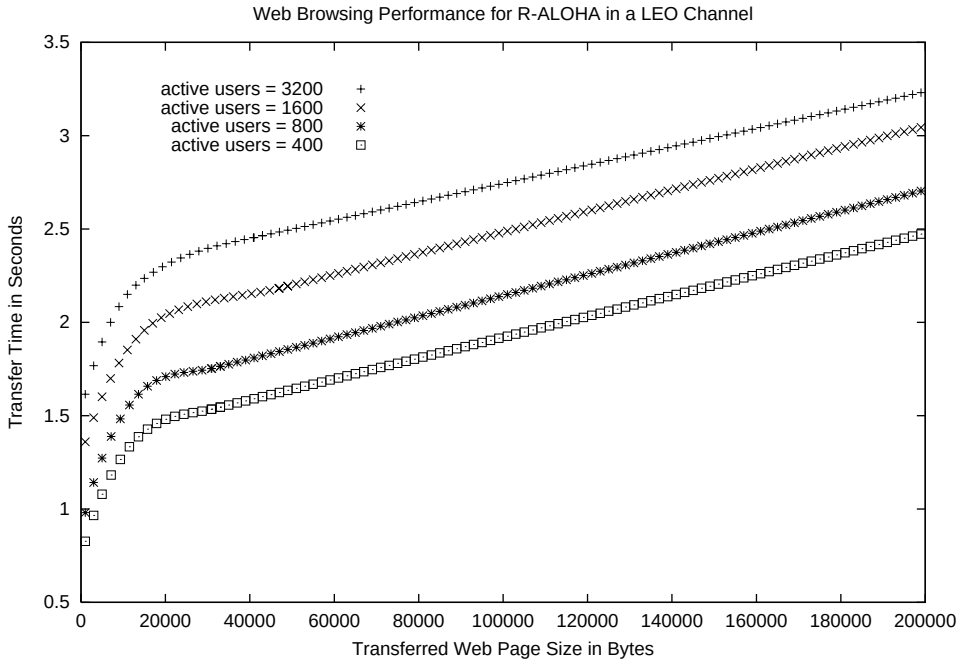

Figure 13: The performance of R-ALOHA for various active HTTP user counts, LEO Channel

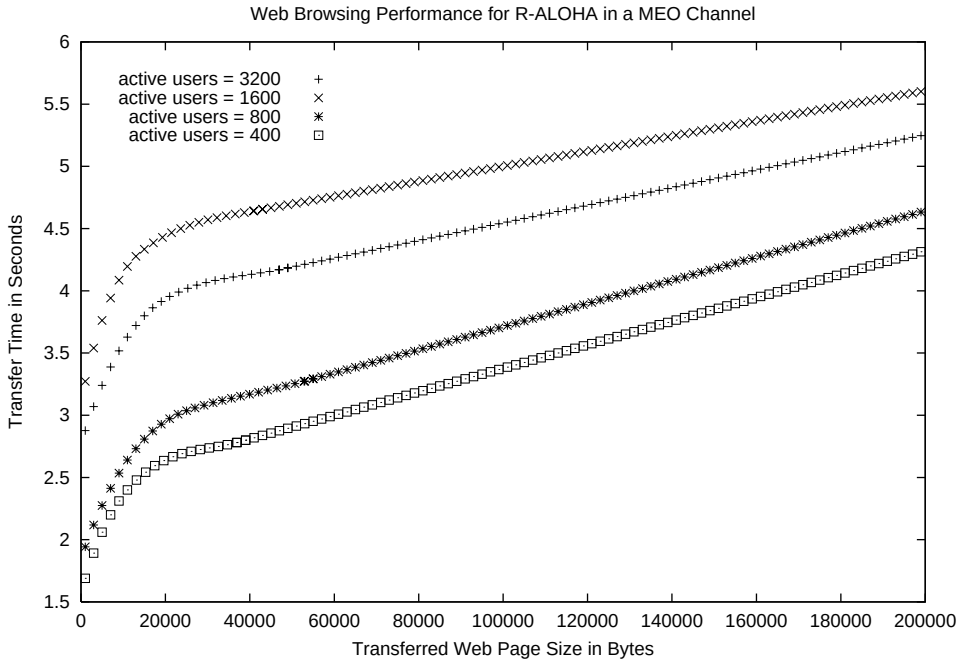

Figure 14: The performance of R-ALOHA for various active HTTP user counts, MEO Channel

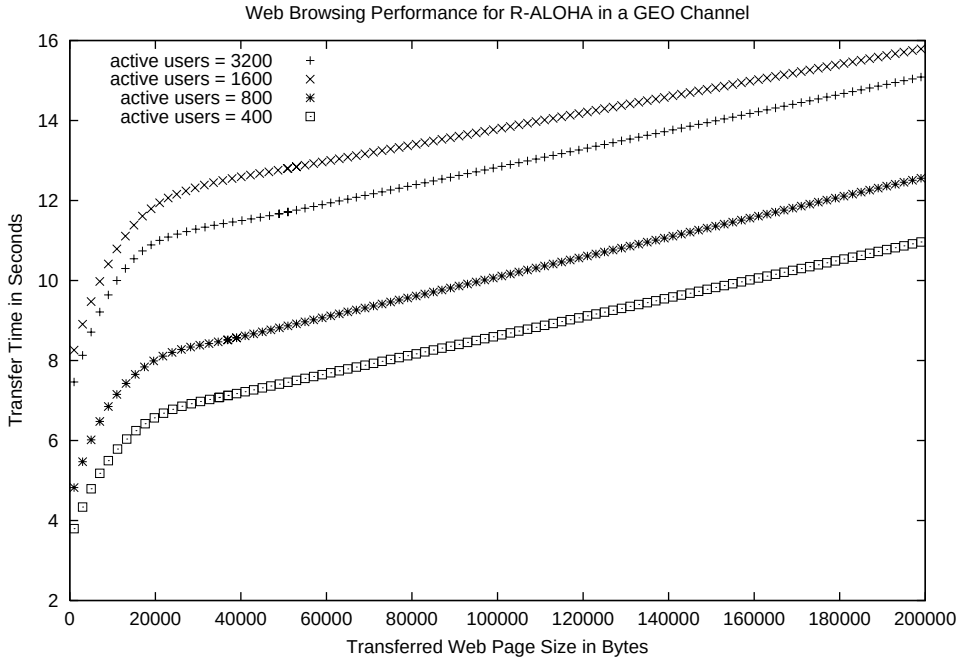

Figure 15: The performance of R-ALOHA for various active HTTP user counts, GEO Channel

6 Conclusions and Future Work

We have implemented a very realistic HTTP 1.1 mechanism in the network simulator ***ns-2.0***, and used this along with our implementation of Slotted ALOHA and Reservation ALOHA to predict the performance of "Web browsing over satellite". We have found that Reservation ALOHA is far superior to Slotted ALOHA for HTTP traffic, due to its ability to achieve implicit reservations. A necessary future work, is to accurately quantify the MAC induced delay (MID) for HTTP downloads. It is well known that TCP performs poorly on high latency satellite channels due to the high time-bandwidth product (TBP) [3]. Showing the increase in download time due to MID, will give a better indication of the penalty paid for using a wireless shared uplink medium. What also must be investigated is a uplink MAC scheme that works for the entire TCP/IP suite of applications (i.e. `telnet, ftp, smtp`) and for real-time applications, such as ***vic*** and ***vat*** [15]. Developing such a MAC technique is necessary if satellite networks are to become a viable option for Internet access.

7 Acknowledgments

The authors would like to acknowledge Dr. Yongguang Zhang of HRL Laboratories for his advice and consultation regarding the modeling of Web transfers and more specifically HTTP 1.1.

References

[1] VINT Network Simulator(*ns*), version 2.0, 1998. http://mash.cs.berkeley.edu/ns/ns.html.

[2] N. Abramson. The ALOHA System - Another Alternative for Computer Communications. In *Proc. of the Fall Joint Computer Conference*, pages 281–285, 1970.

[3] M. Allman and Dan Glover. Enhancing TCP Over Satellite Channels using Standard Mechanisms. Technical Report draft-ietf-tcpsat-stand-mech-04.txt, Internet Engineering Task Force, May 1998. **URL**: http://www.ietf.org/internet-drafts/draft-ietf-tcpsat-stand-mech-04.txt.

[4] R. Caceres, P. B. Danzig, S. Jamin, and D. J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proc. ACM SIGCOMM*, Zurich, Switzerland, 1991.

[5] D. P. Connors, B. Ryu, and S. Dao. Protocols and Tools for Medium Access Control in Satellite Networks. Technical report, HRL Laboratories, 1998. Informational Sciences Laboratory Technical Report TR-605, December 1998.

[6] D. P. Connors, B. Ryu, and S. K. Dao. Modeling and Simulation of Broadband Satellite Networks, Part I: Medium Access Control for QOS Provisioning. *IEEE Communications Magazine*, Mar. 1999.

[7] W. Crowther et al. A System for Broadcast Communication: Reservation ALOHA. In *Proc. of the 6th Hawaii International Conference on Systems Sciences*, Honolulu, Hawaii, 1973.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068, Jan. 1997.

[9] R. Gusella. A measurement study of diskless workstation traffic on an Ethernet. *IEEE Trans. Comm.*, 38:1557–1568, 1990.

[10] J. Hutcheson and M. Laurin. Network flexibility of the IRIDIUM global mobile satellite system. In *Proc. 4th IMSC*, pages 503–507, Ottawa, Cananda, June 1995.

[11] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, Stanford, CA, 1988.

[12] L. Kleinrock and S. S. Lam. Packet Switching in a Slotted Satellite Channel. In *1973 National Computer Conference*, pages 703–710, New York, NY, 1973.

[13] L. Kleinrock and S. S. Lam. Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation. *IEEE Transactions on Communications*, 23(2):410–422, April 1975.

[14] B. Mah. An Empirical Model of HTTP Network Traffic. In *Proc. IEEE INFOCOM*, 1997.

[15] S. McCanne and V. Jacobson. *vic*: A flexible framework for packet video. In *Proc. ACM Multimedia*, San Francisco, CA, 1995.

[16] A. Mukherjee. On the Dynamics and Significance of Low Frequency Components of Internet Load. *Internetworking: Research and Experience*, 5(4):163–205, dec 1994.

[17] J. Postel. User Datagram Protocol. RFC 768, Aug. 1980.

[18] J. Postel. Transmission Control Protocol. RFC 793, Sept. 1981.

[19] W. R. Stevens. *UNIX Network Programming*. Prentice-Hall, 1990.

[20] W. R. Stevens. *TCP/IP Illustrated*, volume I: The Protocols. Addison Wesley, 1994.

[21] M. A. Sturza. Architecture of the TELEDESIC satellite system. In *Proc. 4th IMSC*, pages 212–218, Ottawa, Cananda, June 1995.