Response Initiated Multiple Access (RIMA), a Medium Access Control Protocol for Satellite Channels

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

Correspondence should be addressed to the first author at
Magis Networks Inc.
12555 High Bluff Drive, Suite 255
San Diego, CA 92130
Tel. (858) 259-9240x117
Fax. (858) 259-8908

Symposium : Satellite Communications for the New Millennium

General Conference Topics :

Computer Communications

Satellite & Space Communications

Abstract

The means of achieving full duplex broadband access to either the home or office has been the focus of much research and entrepreneurial development in the past 5 years. This is the so-called “last mile” problem. One approach is duplex access to either a geostationary earth orbit (GSO) or non-GSO satellite, through a relatively small satellite dish located at each home or office. This scenario gives rise to a multiple access problem in the uplink (i.e. terrestrial earth terminal to satellite) channel. In this paper we will present *Response Initiated Multiple Access*, RIMA, a medium access protocol which seeks to maximize the uplink system capacity while providing a suitable quality of service (QoS) for those terrestrial users accessing the satellite system. We assume an underlying TDMA physical layer and will present simulation results which detail the performance of RIMA an environment where the dominant source of user traffic comes from the Hyper Text Transfer Protocol.

1 Introduction

Achieving broadband access to the home or office will usher in a new era of accessibility to Internet / multimedia services. The main obstacle to this becoming a reality is that in most cases, the only connection that exists between the home or office and the high speed fiber optic backbone is a low bandwidth copper wire. Several solutions to this "last mile" problem are taking shape. They include cable modems, digital subscriber lines (xDSL), fixed wireless access (sometimes referred to as LMDS), and duplex satellite access [10]. Each of these techniques has their strong and weak points. Ignoring for the moment which, if any, of these techniques will eventually dominate, we will focus on the duplex satellite access (DSA) model and present a medium access control (MAC) protocol which, we believe, solves many of the uplink channel access challenges. Referring to the satellite dish residing at either the home or office as an *earth terminal*, if DSA becomes a reality, there will exist thousands (or perhaps millions for GSO) of earth terminals competing for uplink bandwidth. We refer to this scenario, shown in Figure 1, as the *Personal Earth Terminal* model [5]. For the purposes of this paper, we shall assume that the satellite is capable of *on-board decisions* (i.e. switching, scheduling, etc.) and is not simply a "bent pipe". It is in this context that our MAC ideas take shape.

Medium access control is always needed in a shared communications channel. For our purposes, we will only consider the uplink channel (i.e. dispersed user to an arbitrating agent, where an arbitrating agent can be, among other things, a cellular base station, satellite, or wireless LAN access point). We will now categorize and briefly describe previous medium access techniques, some of which have been developed in the satellite context. In *Random Access* the dispersed users sharing the channel, in our case these are earth terminals, transmit packets in an uncoordinated manner to the arbitrating agent, in our case the satellite. The ALOHA protocol [1] and its variants [9] 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 access are reduced control signaling and algorithmic overhead, and in ease of implementation. *Reservation Random Access* (RRA) protocols are similar to random access techniques, except that once a packet is received at the arbitrating agent, subsequent channel slots can be *implicitly* reserved. This allows for bursts or streams of packets to be delivered in a more efficient manner than random access. Reservation ALOHA [6], and Packet Reserved Multiple Access (PRMA) are the most widely known RRA protocols. *Demand Assigned Multiple Access* (DAMA) [5] migrates further along the spectrum of bandwidth reservation by allowing the dispersed users to transmit a metric to the arbitrating agent which indicates its instantaneous bandwidth needs. This allows the users to *explicitly* reserve precise amounts of future bandwidth. The main drawback to DAMA techniques is the delay incurred by having to perform a request "handshake" with the arbitrating agent. For satellite channels, this delay is significant. We will now introduce another access technique called *Response Initiated Multiple Access*, or RIMA. With Response Initiated Multiple Access, collision free uplink bandwidth is reserved for a particular user based upon a response packet received at the satellite from the end point of the user's connection. This represents a fundamentally different means of bandwidth reservation, which on its own, will only work for certain types of applications (i.e. connection oriented). We have therefore enhanced RIMA with both random access and DAMA features to make it robust. This technique for multiple access will be expounded upon further in Section 3. The organization of this paper is as follows: first, Section 2 describes the mechanics of the Hyper Text Transfer Protocol [7] (HTTP), which RIMA was principally designed to accommodate. Section 3 describes Response Initiated Multiple Access algorithmically and with a state diagram. Section 4 outlines our simulation methodology and details simulation results for HTTP traffic over a GSO satellite channel. Finally, Section 5 provides a conclusion and thoughts for future work.

2 Web Browsing and the Mechanics of the Hyper Text Transfer Protocol

Clients retrieve World Wide Web (WWW) documents from remote servers using the Hyper Text Transfer Protocol [7]. 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). 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. We have chosen to focus on HTTP 1.1 [7], since it is being standardized and is emerging as the dominant version of HTTP in the Internet. What differentiates HTTP 1.1 from other versions of HTTP is the HTTP 1.1 "pipelines" secondary HTTP GET messages over a single TCP connection. Previous versions of HTTP used parallel TCP connections for inline objects, where these TCP connections could be persistent or not. Using HTTP 1.1, 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* [8] once.

3 Response Initiated Multiple Access

Section 1 gave an overview of medium access control. In that overview, random access (RA), reservation random access (RRA), and demand assigned multiple access (DAMA) were presented. In order to "lay the ground work" for the ideas behind RIMA, some additional intuition about medium access must be stated. One of the main drawbacks to DAMA techniques is the delay incurred by having to perform a request "handshake" with the AA. For most satellite channels, this delay is non-negligible. Since we are simulating HTTP Web transfers, the source traffic emerging from an earth terminal will be quite bursty, so clearly using a fixed bandwidth allocation (such as TDMA) 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 [3] 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 random access techniques when designing RIMA. When any form of random access is used, one must make the *collision space* as large as possible to reduce the collision probability. For instance, the collision space in a TDMA system is the number of time slots available for random access transmission. The collision space is widened when either more slots are available for RA transmissions, or when the number of users contending for these slots is reduced. The collision space can also be logically widened by synchronizing contending users, as was shown by the improvement of Slotted ALOHA [9] over the original ALOHA protocol.

In designing RIMA, three things were taken into consideration : the terrestrial user's needs, the terrestrial user's traffic profile, and the satellite system's needs from the perspective of the network operator.

Terrestrial User's Needs

1. Instant access to the satellite uplink channel without any bandwidth allocation set up time
2. High throughput
3. No collisions on the uplink channel

The first item would indicate that a random access scheme is needed since RA needs no channel set up. The third item seems to preclude random access since collisions are unavoidable with random access. Proceeding now to the terrestrial user's traffic profile:

Terrestrial User's Traffic Profile

- For **HTTP WWW** users the following behaviors / conditions exist:

1. The user will "think" and be inactive for some period of time. This corresponds to the user reading the current Web page. During this think time, no uplink bandwidth is needed.

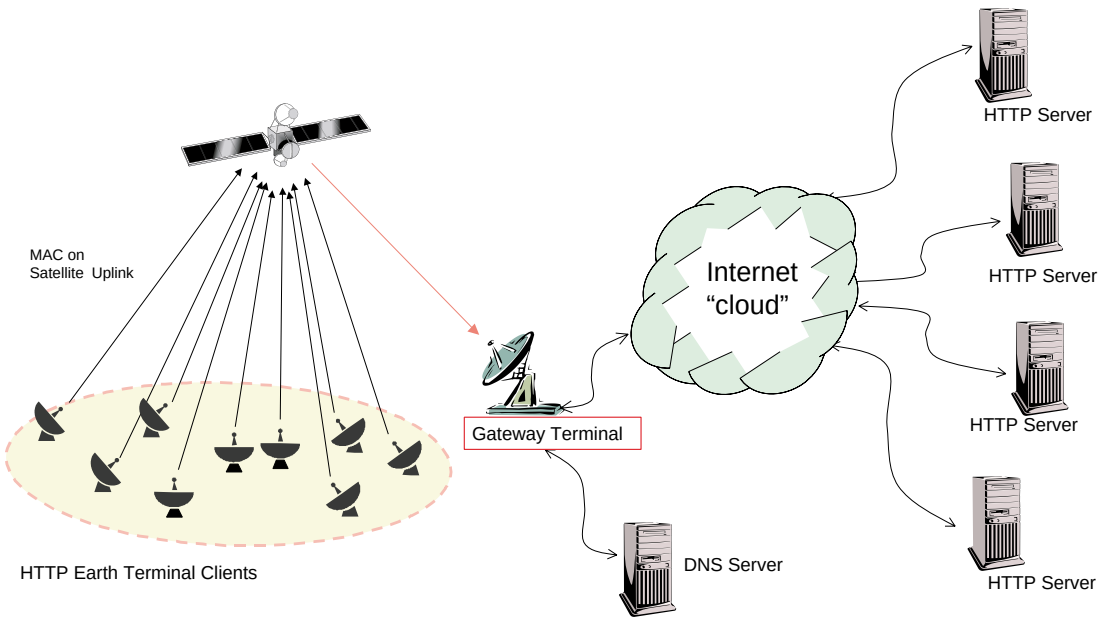

Figure 1: A Satellite Network connecting Clients and Servers through the Internet

2. When the user emerges from thinking, it will need to transmit a relatively small packet (either a DNS lookup packet, which is on the order of 50 to 100 bytes , or a TCP *SYN* packet which is exactly 40 bytes).
3. The user will then commence a heavily asymmetric transaction with a remote HTTP server. The asymmetry will be in the reverse link (i.e. server to client), however there will be points in the transaction when the client will have a sizeable amount of data to send on the forward link. This will occur for the initial HTTP *GET* message and any subsequent GET messages needed to request inline objects. Figure 2 shows an example of the bandwidth utilization of a HTTP transfer on both the forward and reverse links.

• For other protocols, the following conditions may exist:

1. A terminal may produce and transmit packets in a sporadic bursty manner (i.e. `Telnet`).
2. A terminal may require a bulk data transfer in the forward link (i.e. `ftp` or `smtp`).
3. A terminal may produce and transmit packets in a constant bit rate or *sustained stream* manner.

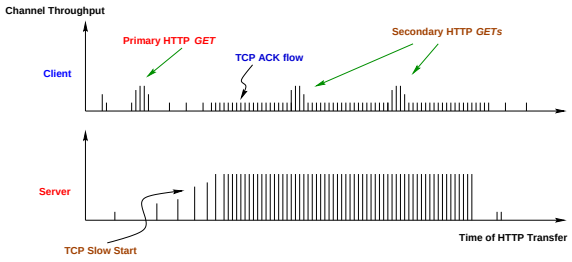

Figure 2: Link Utilization For a Client-Server HTTP Transfer

Satellite System's Needs

1. To support as many terrestrial users as possible. This maximizes the network operator's revenues.
2. To provide a suitable QoS for those users accessing the network

The key idea behind the development of RIMA was that TCP (upon which HTTP runs), has predictable behavior and that TCP packet headers contain a port identification field where what are called "well known port numbers" reside. These well known port numbers indicate the type of application that the particular TCP connection is currently carrying [14]. For HTTP, this well know port number is 80. Let us now define a *TCPAgent* as on end of a duplex TCP connection. When a large packet arrives at a TCP Agent, it will (most likely) transmit an acknowledgement (ACK), since the large packet it just received (most likely) contained data needing to be acknowledged. If a TCP Agent receives a small packet (i.e. an ACK) , then it may transmit a large packet, if it has more data needing to be sent. This behavior is due to the window based flow control mechanism on which TCP operates. Refer to [14] for more detail on TCP behavior. For a star topology, with high latency, the ramifications of this behavior are as follows. When the allocating agent (AA) at the satellite receives a packet from the gateway node, a packet whose destination is an earth terminal (ET) within its spot beam, it can know that once the ET receives the packet, if it was a large packet, then with high probability the ET will need uplink bandwidth in which to transmit the ACK. If the packet was small, with reasonable probability it will need uplink bandwidth in which to transmit more data. Regardless of how bursty and unpredictable the arrival of traffic from the gateway to the satellite is, this behavior will still exist and be predictable. Thus if collision free (i.e. reserved) uplink bandwidth is allocated in the uplink channel for a particular user upon receiving a packet destined for that user from the gateway, this will reduce the need for random access transmissions, since the uplink bandwidth needs of users can be accurately predicted. This has the effect of widening the *collision space*. Since random access collisions are very costly in high latency systems, if random access is used sparingly, then a potentially huge throughput increase can be realized. This gives rise to the name *Response Initiated Multiple Access*, in that a response from the server, initiates collision free multiple access in the uplink. The only time random access must be employed is when a HTTP user emerges from the "thinking" period, since this cannot be predicted. When a user emerges from the thinking period, it will either issue a DNS lookup packet or a TCP SYN packet, both of which are small. Small packets have lower collision probability since they will occupy a smaller region of the collision space. Since RIMA will guarantee collision free ACKs to flow in the uplink channel, the TCP window at the server will open very quickly [8, 14], resulting in excellent throughput. With this methodology understood, a framework for implementing RIMA can be established.

Figure 3 shows how RIMA is implemented in a satellite based AA. As with the ALOHA protocols, RIMA is TDM based and assumes time slots making up a TDMA frame. At the beginning of each downlink TDM frame, the RIMA AA issues a *frame descriptor packet* (FDP). This packet indicates which collision free slots are allocated to which ETs. The remaining slots are available for RA transmissions. After the FDP is sent, the remaining bandwidth in the downlink is used for data transmissions. The FDP is generated by a combination of two inputs : the RIMA Allocation Algorithm operating on server packets arriving via the gateway (see the following pseudo code), and additional bandwidth requests received from the ETs within the satellite's spot beam.

The following pseudo code illustrates the RIMA Allocation Algorithm (RAA).

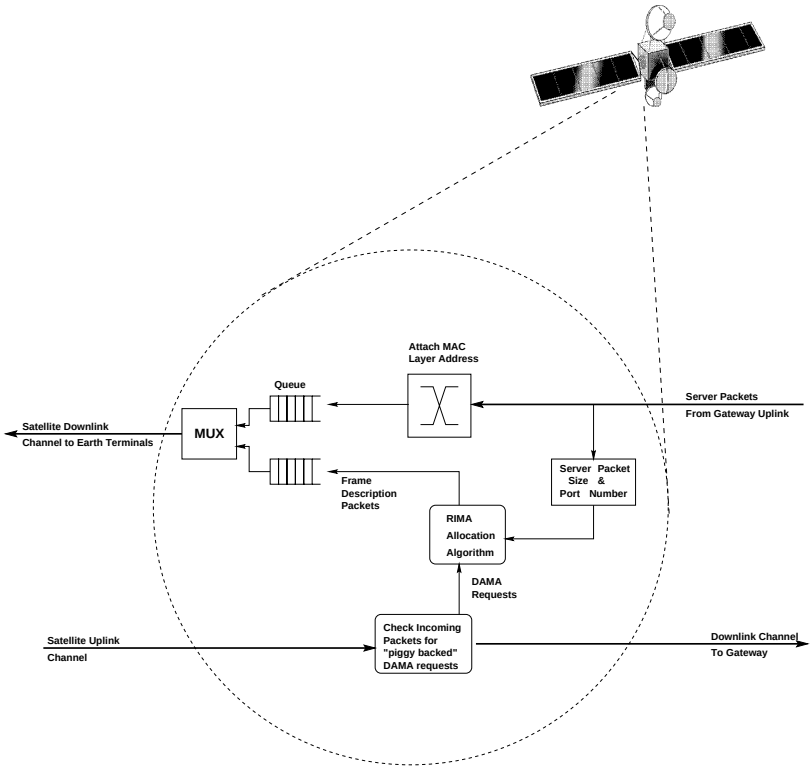

Figure 3: A Functional Block Diagram of how RIMA is implemented in a Satellite

```
if( PORT_NUM == 21 ) { /* ftp is port number 21 */
     if( PACKET_SIZE <= ACK_SIZE )
          BYTES_ALLOC = MTU_SIZE
     else BYTES_ALLOC = ACK_SIZE
} else if( PORT_NUM == 23 ) { /* telnet is 23 */
     if( PACKET_SIZE <= ACK_SIZE )
          BYTES_ALLOC = 0
     else BYTES_ALLOC = ACK_SIZE
} else if( PORT_NUM == 25 ) { /* smtp is 25 */
     if( PACKET_SIZE <= ACK_SIZE )
          BYTES_ALLOC = MTU_SIZE
     else BYTES_ALLOC = ACK_SIZE
} else if( PORT_NUM == 80 ) { /* http is 80 */
     if(PACKET_SIZE <= ACK_SIZE)
          BYTES_ALLOC = 500
     else BYTES_ALLOC = ACK_SIZE
} else BYTES_ALLOC = 0
```

Where:

PORT_NUM is is the TCP well known port number
PACKET_SIZE is the size in bytes of the received server packet
ACK_SIZE is the size of a TCP ACK packet (in IPv4 this is 40 bytes)
BYTES_ALLOC is the amount of bandwidth in bytes that is apportioned by the RAA
MTU_SIZE is the size of the maximum transmit unit

What this algorithm accomplishes is that if a packet arrives at the satellite from the server, the TCP and or UDP [14] port number can be read and the proper amount of uplink bandwidth can be pre-allocated. We are most interested in HTTP, therefore when a small HTTP packet arrives (i.e. a TCP ACK), 500 bytes are allocated in the uplink channel. Apportioning 500 bytes if the packet is small is a subjective value chosen based upon the measurement of HTTP GET sizes presented in [11]. Likewise, when a large HTTP packet is received, just enough bandwidth is allocated to accommodate a ACK packet (40 bytes). Since we are running RIMA using a TMDA multiple access physical layer, these byte quantities will be rounded up to the appropriate amount of TDMA slots. The inclusion of other TCP/UPD port numbers is to make RIMA robust across the entire suite of dominant TCP/IP applications. Since there is always the possibility that an individual ET needs more bandwidth than RIMA Allocation Algorithm gives, then additional bandwidth requests can be "piggy backed" on arriving data packets. These additional bandwidth requests will be accommodated, subject to B_{max}^{frame}, the maximum amount of bandwidth an individual user may have of a per frame basis. Having a upper bound on bandwidth will preclude the real possibility of one user "grabbing" a disproportionate amount of uplink bandwidth when performing a bulk transfer. Another notable benefit of RIMA is that very little *state* is maintained at the satellite. Many previously proposed MAC techniques had the AA at the satellite maintaining state information about every active user in the spot beam. In a GEO satellite environment where there are potentially tens of thousands of users, a scheme that requires the AA to maintain user state information is not scalable.

Since RIMA has piggy backed DAMA functionality, it can accommodate bulk transfers (`ftp`, `smtp`) with the same efficiency as DAMA. If a constant bit rate (CBR) source is attached to RIMA, it would also work since DAMA techniques work on CBR sources as well. Sporadic bursty users (such as `Telnet`, `rlogin`) can also be accommodated through random access (since their byte allocation in zero). All of this means that although RIMA is optimized for a particular, currently dominant Internet traffic pattern (i.e. HTTP WWW transfers), it will also work as well as other MAC techniques for the rest of the applications common in the Internet. Figure 4 shows the state machine that makes up the RIMA protocol.

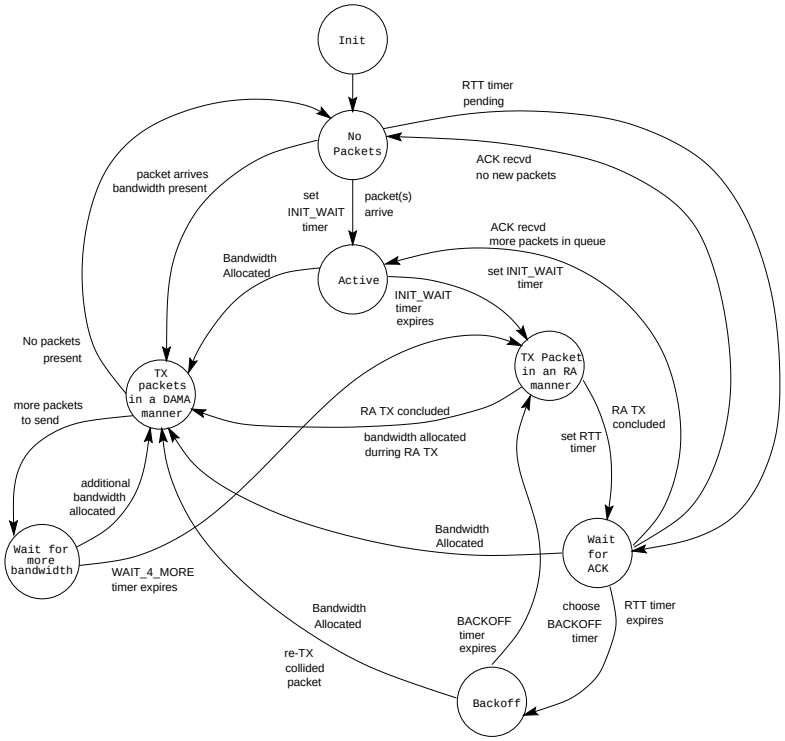

Figure 4: A State Transition Diagram for Response Initiated Multiple Access

4 Simulation Methodology and Results

We have chosen to compare RIMA to two other well known random access (RA) techniques, Slotted and Reservation ALOHA. Although these are well known techniques, there implementations vary. For a complete description as to how S-ALOHA and R-ALOHA were implemented for the purposes of this simulation, please refer to [4]. 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. All earth terminals share a common uplink channel. The MAC protocol in place on the uplink channel is either S-ALHOA, R-ALOHA, or RIMA. The DNS server is located at the terrestrial gateway node,

one satellite hop away from each earth terminal. Since we are interested in the performance of the uplink portion of the satellite link, we have chosen a heavily asymmetric (in terms of bandwidth) duplex link. We have set the downlink bandwidth to be 10 Mb/s while only allowing for 100 Kb/s in the uplink. This will allow our simulation results to predict the number of terrestrial users that can be supported per 100 Kb/s of uplink bandwidth. We would liked to have chosen greater bandwidths for both the up and down links, but the limitations of computer simulation prohibited this. The delay across the satellite channel was chosen to be 135ms. This allows us to measure the performance in geosynchronous earth orbit (GEO) scenarios. 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 4.3. The client measures the time from the start of the HTTP transaction until its completion. This measurement will serve to quantify performance of the three MAC schemes. This entire simulated network was implemented using the discrete event network simulator ***ns*** [12]. We had to add the MAC layer functionality and also added the necessary code to accurately model HTTP 1.1 mechanics (see Section 4.2). With this simulation tool in place we experimented with Web transfers using both Slotted and Reservation ALOHA, and RIMA.

4.1 Empirical HTTP Measurements as simulation Input

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 [11]. 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 [11].

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

These CDF's 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 use all of the above statistics except for one, that being the user think time. The CDF of the user think time in [11] 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.

4.2 Mechanics of a Simulated Web Transfer

The procedure for a HTTP transaction follows exactly the means outlined in Section 2. 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 [14]. Once the connection is established the client sends the initial HTTP GET message. The size of this GET message is distributed according to [11]. The server responds with a REPLY, also distributed according to the measurements in [11]. This procedure continues until the initial Web page and any inline images are completely transferred, according to the procedure of HTTP 1.1. 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 [11]. 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.

4.3 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 [13], 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.

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

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` 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 (RTT) 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, c 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.

$$d_i = m_{id} + \nu_{(id,i)} \tag{2}$$

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.

4.4 Comparison via Simulation Results

Figures 5- 7 show the performance of S-ALOHA, R-ALOHA, and RIMA under a GEO (250ms) channel under what we presume to be a *heavy loading condition*. The x-axis shows the file size (i.e. the size of the entire HTTP transfer) and the y-axis shows the average time to completely transfer the file for the various MAC protocols. To determine what constituted a heavy load, we applied the following calculus. According to [11], 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 $\frac{1154 \cdot 8 bits}{60 sec} = 154 bps$. 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 100Kb/s channel at 36% throughput gives us 36 kb/s. Dividing 36 kb/s by 154 b/s, gives us 234 supported users. Thus to simulate a heavy load, the active user count was set to 300 HTTP clients per 100 Kb/s of uplink bandwidth. We have also varied the physical layer *granularity* to see what if any effect that has on the performance of each protocol. By granularity, we mean the size of each TDMA slot. We set the basic unit of granularity to be 50 bytes (close to a 53 byte ATM cell) and ran simulations with 1, 2, 4, and 8 cells per slot. Figure 5 shows very poor performance for Slotted

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

ALOHA, especially when the Web page gets sizeable. Instead of focusing on granularity for S-ALOHA, we showed the performance as a function of load, varying the number of active users from 20 to 150. It appears that the channel becomes saturated very quickly, even for only 20 users. The reason that the channel becomes saturated, instead of "blowing up" and becoming unstable is that we set an *Abnormal Timeout* timer to just over a minute (75 seconds). This means that a user "gives up" if the Web transfer does not complete within this 75 second time interval. Examining Figures 6 and 7 we see that using some form of reserved access drastically reduces Web page transfer times. However RIMA outperforms R-ALOHA by over 30% for all levels of granularity. For small Web pages (under 20 KBytes), which tend to dominate according to [11], for a 250ms GEO channel, RIMA has a transfer time of 8-10 seconds as opposed to 12-15 seconds for R-ALOHA. This 30% improvement appears throughout the graph. At the bottom of each graph there lies the "time-bandwidth product" [2] lower bound. This line represents the minimum transfer time due to TCP's *slow start* algorithm coupled with the high latency of the GEO channel. It appears that RIMA comes within 1-2 seconds of this lower bound. A final thing to consider is that RIMA is immune to high variations it client-to-server round trip times (RTTs). We developed our Internet Path Emulator by conducting a series of `ping` measurements in the continental United States. `ping` packets tend to experience better round trip times due to the fact that they are not "dropped" as frequently by routers because of their small size. Therefore we would assert that in a more realistic scenario, RIMA would perform the same, while R-ALOHA would deteriorate as RTTs increased in variance.

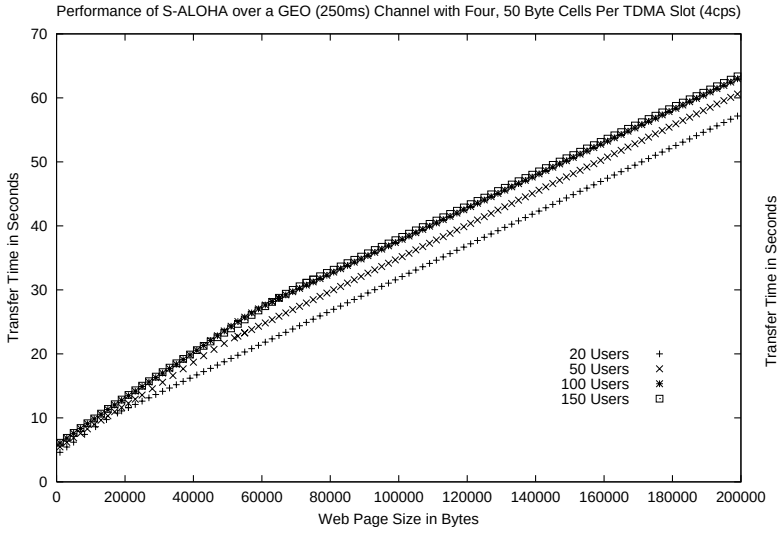

Figure 5: The Performance of S-ALOHA with multiple number of Active Users

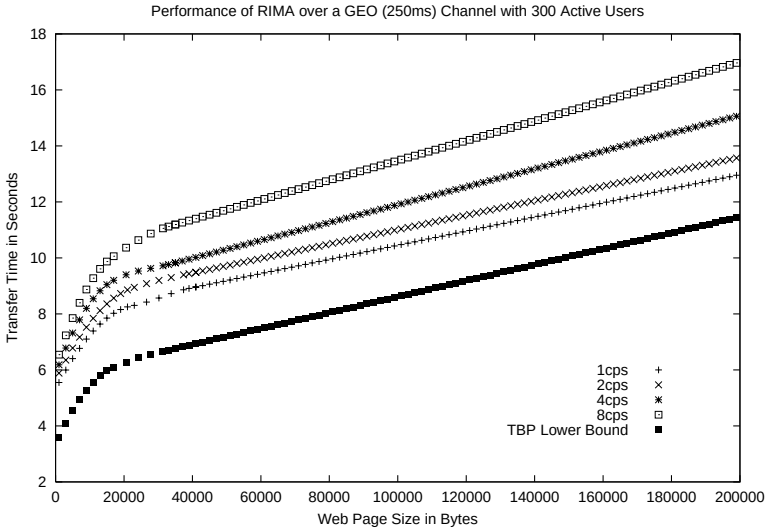

Figure 6: The Performance of RIMA with 300 Active Users with the number of 50 byte cells per slot Varying

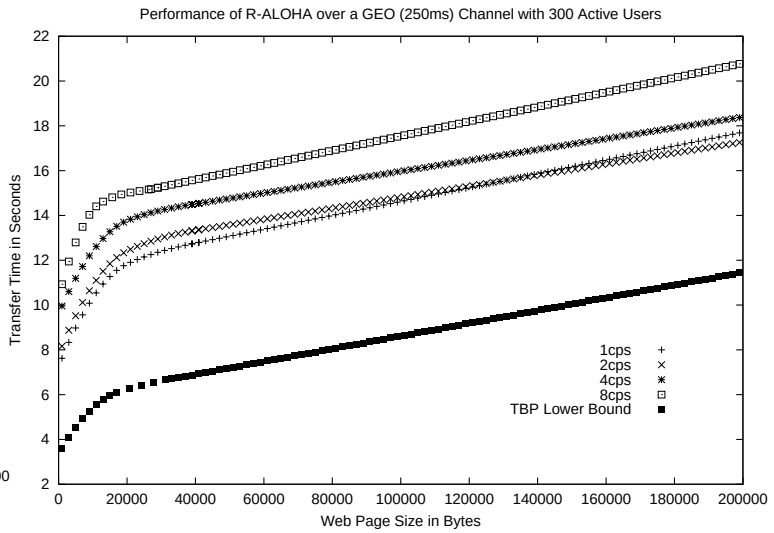

Figure 7: The Performance of R-ALOHA with 300 Active Users with the number of 50 byte cells per slot Varying

5 Conclusions and Future Work

Some conclusions are now stated in summary form:

• Some inherent reservation mechanism is needed for HTTP transfers. This eliminates S-ALOHA from consideration.
• RIMA consistently outperforms R-ALOHA by at least 30% under mild RTT variation
• Even though not simulated, RIMA can support sporadic, bursty sources (i.e. `Telnet`) naturally, while R-ALOHA achieves half the throughput of S-ALOHA under these traffic patterns [6] and therefore cannot support the full **TCP/IP** suite of applications.
• The DAMA aspects of RIMA allows it to support bulk transfers well (since the TCP ACK flow in the uplink during the HTTP transfer is identical to an `ftp` transfer, only with smaller packet sizes).
• The only thing not clear from these simulations is RIMA's ability, with respect to R-ALOHA, to support CBR or *sustained stream* traffic.

6 Acknowledgments

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

References

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

[2] M. Allman, Dan Glover, and L. Sanchez. Enhancing TCP Over Satellite Channels using Standard Mechanisms. Technical Report RFC 2488, Internet Engineering Task Force, January 1999. **URL**: ftp://ftp.isi.edu/in-notes/rfc2488.txt.

[3] 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.

[4] D. P. Connors and G.J. Pottie. The performance of http over satellite random access channels. In *33rd Annual Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, Jan. 1999. Society for Computer Simulation (SCS) 2000 Western Multi-Conference.

[5] 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.

[6] 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.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

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

[9] 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.

[10] Kdd to launce two-way satellie internet service. Distributed by Wireless-NetNow, April 1999.

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

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

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

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