An Analysis of TCP Startup over an Experimental DVB-RCS Platform

Alberto Gotta
DIST-University of Genoa and CNIT
University of Genoa Research Unit,
Via Opera Pia 13, 16145 Genoa (Italy)
alberto.gotta@isti.cnr.it

Francesco Potortì, Raffaello Secchi
Information Science and Technology Institute "Alessandro Faedo",
Italian National Research Council (C.N.R.),
Via G. Moruzzi, 1, San Cataldo, I-56124 Pisa
{Potorti,raffaello.secchi}@isti.cnr.it

***Abstract*— Satellite systems are evolving towards higher available bandwidths and dynamic allocation based on instantaneous traffic rates offered at the stations, so called BoD (bandwidth on demand) channel sharing. This trend is paired with more and more powerful error correcting schemes, like those adopted in the recent DVB-S2 standard, which promise to make the channel virtually immune from packet errors. These factors, together with the significant round-trip delay of geostationary satellites, combine so that most TCP connections would send all of their data during the Slow Start phase. We investigate the performance of TCP during startup on recent BoD system by observing and explaining the behavior of different TCP flavors on different systems when transmitting data over the Eutelsat's Skyplex Data satellite system. We make recommendations for choosing and improving TCP implementations and for future BoD allocation schemes.**

I. INTRODUCTION

Recent and future satellite systems are characterized by quasi-ideal loss characteristics and broadband links. Consequently, in the first place congestion - as opposed to packet loss due to link errors - dominates the TCP dynamics, and in the second place the delay-bandwidth product is very large. As a consequence of these two effects, TCP sessions are often concluded within the Slow Start phase, without incurring any packet loss.

The performance of the Slow Start phase is then extremely critical as far as network performance is concerned, but this issue has not received much attention from the research community, especially concerning experimental measurements on recent platforms. Moreover, the Slow Start phase is slowed down in BoD (Bandwidth on Demand) systems, which are currently emerging in the market.

In fact, the bandwidth is assigned on the basis of the stations' requests, which in turn depend on the current transmission rate and possibly on the transmission backlog of each station. Requests from the stations need 250 ms propagation delay on geostationary satellite networks, and the allocation needs 250 ms to be broadcasted to the stations, meaning that assignments are always late with respect to incoming traffic by at least 500 ms, to which management overheads should be added. Since the throughput in the Slow Start phase of TCP increases at each RTT, the allocated bandwidth is always less than the offered traffic, which accounts for the very long Slow Start phase we observed in BoD systems.

In [1] we have measured and compared the improvement introduced by TCP variants and options (TCP Westwood and SACK option) in recent Linux kernels, by means of experiments on Skyplex Data, a commercial DVB-RCS satellite platform based on Eutalsat satellite, while [2], [3] do a simulative analysis of TCP behavior on BoD satellite systems. In this work we complement that analysis with measurements focusing on the startup behavior of Linux 2.6 (with and without SACK option) and FreeBSD 5.4 implementations of NewReno TCP on the Skyplex platform.

II. THE SKYPLEX PLATFORM

The measurements reported in this paper are carried out on Skyplex Data, an experimental satellite network based on the Skyplex OBP technology run by Eutelsat on the HotBird 6 satellite [4], which relies on DVB-RCS features, while not being completely DVB-RCS compliant. On Skyplex, IP packets are encapsulated into an Mpeg-2 transport stream and transmitted to the satellite using Time Division Multiple Access (TDMA) uplinks using DVB-RCS format from small traffic terminals. On board the satellite, these signals are demodulated, regenerated and sent downlink in a format very similar to DVB-S.

In order to meet user requirements, the satellite bandwidth (about 36 Mbps) can be divided in Low Rate channels (LR at 2.112 Mb/s) and High Rate channels (HR at 6.226 Mb/s), because each channel has a TDMA structure that allows a configurable number of time slots per frame. Our experimental testbed consists of an LR carrier with a TDMA frame of 48 time slot.

The slot assignment is dynamic (BoD). The bandwidth is requested periodically by each terminal to the network control center on the basis of its own instantaneous need, and the time slot are assigned in a best-effort mode. However, a minimum guaranteed bandwidth equivalent to one slot per frame (44 kb/s), used for signalling and user data, is reserved to each traffic terminal.

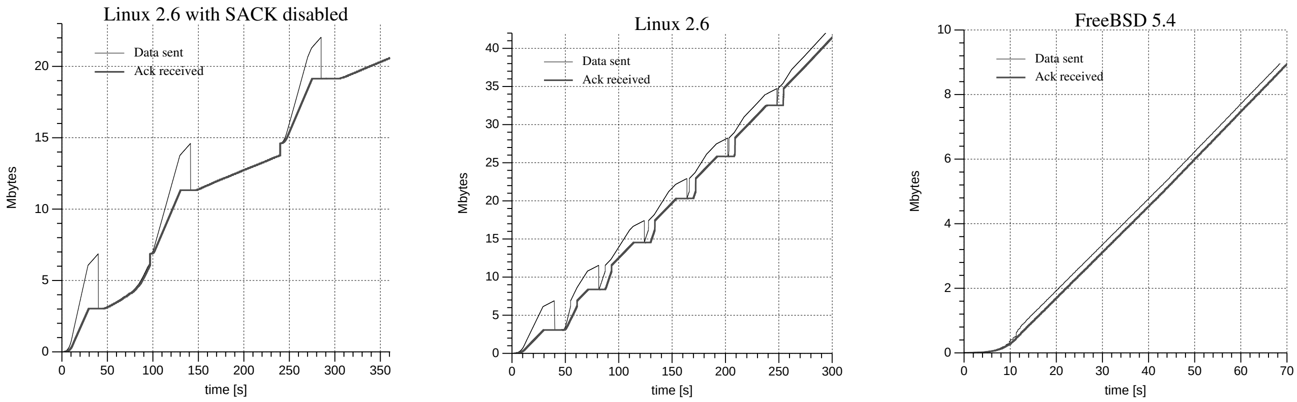


Fig. 1. First experiments with TCP over satellite.

III. TCP MEASUREMENTS OVER SKYPLEX

We started by comparing the performance of some commonly used TCP stacks on Eutelsat's Skyplex Data satellite platform: the results are shown in Fig. 1, which shows how Linux TCP, with and without SACK option, and FreeBSD TCP behave on the satellite link.

First experiments with Linux without SACK (TCP's selective acknowledgement option) reveal that, on the first connection to a given destination, performance is severely hindered because of the first congestion event at the end of the Slow Start phase, which causes the loss of several segments that TCP takes a long time to recover. Use of SACK makes recover much more efficient. Performance on subsequent connections is good if the cached value of the Slow Start threshold has not yet expired.

Instead, experiments with FreeBSD show good performance on all connections, because recent FreeBSD kernels perform an estimation of the bandwidth-delay product and prevent the congestion window from exceeding the estimated value, thus completely preventing packet losses, at least in our setting. This method produces a slight underutilization of the satellite channel, while providing excellent startup performance even on the first connection.

It is apparent from the figures that the inner workings and the overall performance of these three TCP flavors substantially differ:

- Linux without SACK fills up the bottleneck buffer at the end of the Slow Start phase, and experiments many packet losses when the buffer overflows; the subsequent Fast Recovery phase is not fast enough to avoid a timeout and a very slow Slow Start phase, that retransmits a high number of packets already transmitted, whose duplicate ACKs do not contribute to increasing the congestion window. This behavior is well known [5].
- Linux with SACK behaves much better, essentially because the Slow Start phase is fast thanks to the selective acknowledgments; the resulting throughput is, on average, as fast as the channel permits, even if the flow of packets is very irregular.
- FreeBSD uses an algorithm for estimating the bandwidth available to the TCP connection and the latency of the channel, and uses these estimates to cap the rate of packet transmission; the resulting behavior, in this simple case where a single connection occupies the whole channel, is almost as efficient on average as Linux and in addition the packet rate is extremely regular, without even a lost packet and with an RTT only slightly higher than the minimum.

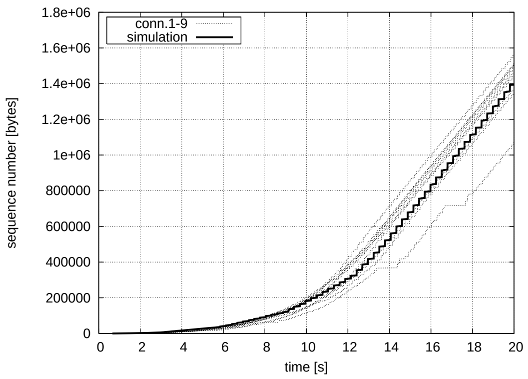
IV. FOCUSING ON TCP STARTUP PHASE

In this section we focus on the behavior of TCP during the Slow Start phase. Fig. 2 compares the first 20 seconds of acknowledged sequence numbers for two different TCP implementation transmitting data over the satellite link. We show the performance of the default FreeBSD and Linux implementations, reporting for each case 9 connection traces.

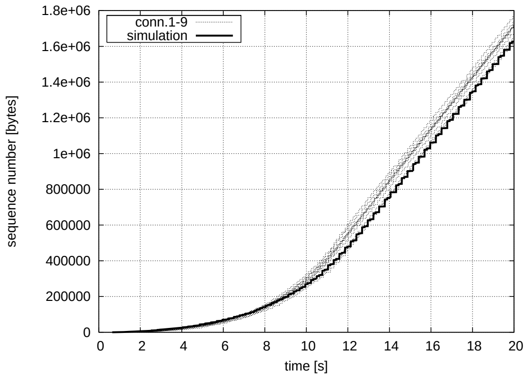
We note that at the beginning of the connections the behaviors of both TCP implementations are similar. We can distinguish two phases: an approximatively exponential increase of sequence number and a linear increase phase. Indeed, since during the Slow Start phase the congestion window is increased by twice the amount of acknowledged bytes, the ACK reception rate increases exponentially until the TCP throughput reaches the available channel capacity; after that point the ACK rate remains constant. However, we notice that the time to double the congestion window is considerably longer than the connection round trip time usually observed over terrestrial networks. As will be clarified in the following, this phenomenon is to attribute to the BoD policy adopted to share the satellite bandwidth.

In fig. 2 real traces are plotted together with simulated traces obtained with a satellite TDMA BoD allocator for ns-2 [6], which implements the dynamics of bandwidth assignment protocol. The simulator, validated with comparison with satellite measurements made on Eutelsat's Skyplex Data platform, is useful to assess the TCP behavior under different working parameters.

Though FreeBSD's TCP has the same qualitative behavior of Linux's TCP during startup, Linux is quicker because



(a) FreeBSD TCP



(b) Linux TCP

Fig. 2. Startup phase behavior for TCP Newreno in FreeBSD and Linux.

Linux, according to RFC 3390 [7], sets the initial window to three packets, while FreeBSD more conservatively uses an initial window of only one packet.

In the following we develop a simple approximate analytical model of TCP startup over BoD controlled links. Though this model is not meant to be accurate at the packet level, it can be very useful to evaluate the impact of the relevant parameters on the overall connection completion time. In particular, it accounts for the delay occurring between bandwidth request and assignment introduced by the BoD mechanism.

A. *Congestion window model*

We consider a single greedy TCP connection, with no packet losses due to corruption and no throughput limitations due transmit or receive buffers, receive window or bottleneck buffer. Also, we assume that the initial Slow Start threshold is big enough that the Slow Start phase does not end before the channel is saturated. TCP starts to inject packet in a previously idle satellite terminal that accesses a shared channel using a rate-based allocation policy. This means that the terminal asks for an allocation proportional to its incoming traffic rate with a *proportionality factor* k that is normally set to 1, and the BoD controller assigns a share equal to the request, as long as there is space on the channel. New requests are done once every *request period*, denoted by T_r, and new assignments are done once every *allocation period*, denoted by T_a. This means that, if the mean round trip time is τ, the average *allocation delay* D_a, that is, the delay between making a request and receiving the corresponding allocation is

$$D_a = \tau + T_r/2 + T_a/2. \tag{1}$$

Note that we speak about *mean RTT* because this is TDMA system, that is, data at the terminal are not sent continuously, but in TDMA bursts. Let $\mathrm{w}(t)$ be the TCP congestion window expressed in segments, and $\mathrm{a}(t)$ the number of ACKs arrived at the sender in interval $[0:t]$. During the Slow Start phase the congestion window is increased by one for each ACK arrival, therefore we have

$$d\mathrm{w}(t) = d\mathrm{a}(t). \tag{2}$$

Assuming no delayed ACKs, the ACK arrival rate at the terminal $d\mathrm{a}(t)/dt$ is equal to the rate of packets entering the satellite channel one RTT before, that is, the bandwidth assigned to the satellite terminal one RTT before. Consequently, if enough space is available on the channel, the bandwidth available to the TCP connection is $k\mathrm{w}(t)/\tau$, and we have

$$\frac{d\mathrm{w}(t)}{dt} = \frac{d\mathrm{a}(t)}{dt} = k\frac{\mathrm{w}(t-T)}{\tau}, \qquad T \equiv \tau + D_a. \tag{3}$$

In order to take delayed ACKs into account, we should divide the right hand side of (3) by 2, or alternatively divide k by 2. However, the TCP implementations we observed disable the DelACK mechanism during Slow Start in order to accelerate the growth of the congestion window. Also, from simulation we have observed that, even when DelACK is enabled during startup, the interarrival time between ACKs is larger than the DelACK retransmit timeout for most of the Slow Start phase, resulting in the transmission of one ACK for each received packet. For these reasons, in the following we assume no DelACK.

By transforming equation (3) into the Laplace domain and denoting by W_0 the TCP initial window, we have

$$s\mathrm{W}(s) - W_0 = k\frac{\mathrm{W}(s)}{\tau}\exp(-sT) \tag{4}$$

that can be rewritten as

$$\mathrm{W}(s) = \frac{W_0}{s - k\frac{\exp(-sT)}{\tau}}. \tag{5}$$

Though an exact expression for $\mathrm{w}(t)$ could be obtained as an infinite sum, we prefer to use a second order approximation, which can be expressed in closed form. Thus, we use the approximate form $\exp(-sT) \simeq \frac{1-s\frac{T}{2}}{1+s\frac{T}{2}}$ to obtain

$$\mathrm{W}(s) \simeq \frac{W_0}{s - \frac{k}{\tau}\frac{1-s\frac{T}{2}}{1+s\frac{T}{2}}} = \frac{\frac{\tau}{k}W_0(2+sT)}{s^2\frac{\tau}{k}T + s(\frac{2\tau}{k}+T) - 2}, \tag{6}$$

which has roots s_1, s_2

$$s_1, s_2 = \frac{1}{T}\left(-1 - \frac{\rho k}{2} \mp \sqrt{1 + 3\rho k + \left(\frac{\rho k}{2}\right)^2}\right), \quad \rho \equiv T/\tau \tag{7}$$

We can now write an expression for the time evolution of the congestion window from the beginning of the Slow Start phase until the moment the channel is saturated:

$$\mathrm{w}(t) = A\exp(s_1 t) + B\exp(s_2 t), \tag{8}$$

where A and B are appropriate constants. The expression (10) holds until $t = t_s$, the moment when the TCP rate $w(t)/\tau$ becomes equal to the available capacity μ (pkt/sec). For $t > t_s$ the ACK rate is capped by μ, and the growth of the congestion window depends on the exact implementation, as mentioned above.

In order to compute the value of t_s, we observe that the root $s_2 > 0$ gives rise to an increasing exponential while the root $s_1 < 0$ to a decreasing one. Hence, for $t_s > 1/s_1$ we can neglect the first term of expression (8) and assume $\mathrm{w}(t_s) \simeq B\exp(s_2 t_s) = \mu\tau$, thus obtaining

$$t_s \simeq \frac{1}{s_2}\ln\left(\frac{\mu\tau(s_2T - s_1T)}{W_0(2 + s_2T)}\right). \tag{9}$$

By transforming (8), equating it to (5) and solving for A and B in $s = s_1$ and $s = s_2$, we can write (8) as

$$w(t) = W_0\frac{(2 + s_1T)\mathrm{e}^{s_1 t} - (2 + s_2T)\mathrm{e}^{s_2 t}}{s_1T - s_2T} \qquad \text{for } 0 \le t \le t_s, \tag{10}$$

which makes it clear that, during the Slow Start phase on a BoD system using a rate-based policy, the congestion window evolution has an approximately exponential evolution with a time constant $1/s_2$. This is equivalent to having a fixed-allocation policy together with an increased RTT depending on both the allocation delay through ρ (see (7)) and the proportionality factor k:

$$\tau_{\mathrm{eq}}(\rho, k) = \tau\frac{1 + \frac{\rho k}{2} + \sqrt{1 + 3\rho k + \left(\frac{\rho k}{2}\right)^2}}{1 + \frac{k}{2} + \sqrt{1 + 3k + \left(\frac{k}{2}\right)^2}} \tag{11}$$

The expression for $\tau_{\mathrm{eq}}(\rho, k)$ is an increasing function of ρ, and consequently of the allocation delay. This means that, from the point of view of a TCP connection that has not yet reached channel saturation, increasing the allocation delay D_a on a rate-based BoD allocation system has the same effect as increasing the round trip delay τ on a fixed allocation system. So, generally speaking, any rate-based BoD system slows down the throughput growth of TCP connections. This slowdown can be partially compensated by choosing a larger k value for the BoD system, as discussed below.

B. Transfer time during startup

Using (8), and neglecting the exponential term $\exp(s_1 t)$, we can estimate the amount of data $\mathrm{N}(t)$ transferred in the time interval $[0 : t]$. To this purpose we integrate the TCP throughput $\mathrm{w}(t)/\tau$:

$$\mathrm{N}(t) = \frac{B(\exp(s_2 t) - 1)}{s_2\tau}. \tag{12}$$

From t_s to the time t_l when the first congestion loss is detected, the TCP throughput is limited by the channel available capacity μ. The two phases are put together in the following expression

$$\mathrm{N}(t) = \begin{cases} \frac{B}{s_2\tau}\exp(s_2 t) - \frac{W_0}{k} & \text{for } 0 < t \le t_s \\ N(t_s) + \mu(t - t_s) & \text{for } t_s < t < t_l \end{cases}, \tag{13}$$

from which one computes the amount of data transmitted before saturation, that is, for $0 < t \le t_s$

$$\mathrm{N}(t_s) \simeq \frac{\mu}{s_2} - \frac{W_0}{k} \tag{14}$$

and the time D required to transmit N packets (for $D > 1/s_1$) as a function of the initial window W_0, the round trip time τ and the value of T

$$D = \begin{cases} \frac{1}{s_2}\ln\left(\frac{kN + W_0}{W_0}\frac{s_2T(s_2T - s_1T)}{\rho(2 + s_2T)}\right) & N \le N(t_s) \\ \frac{1}{\mu}(N - N(t_s)) + t_s & N > N(t_s) \end{cases}. \tag{15}$$

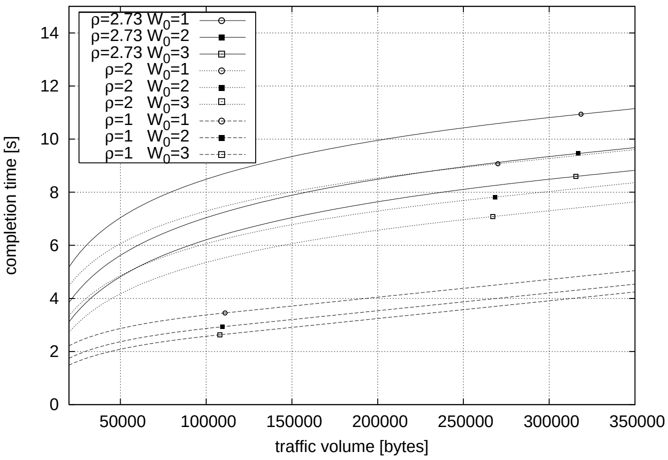
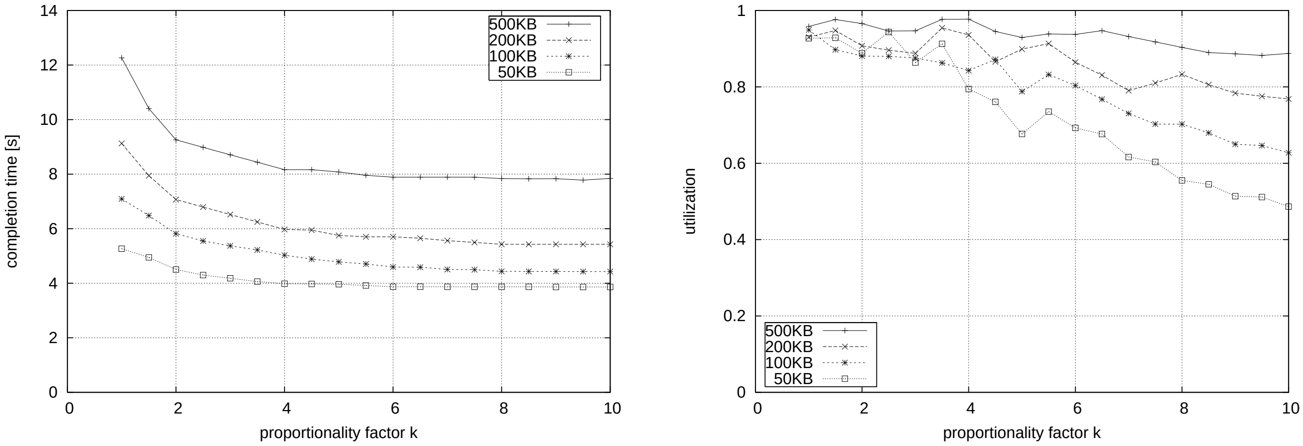


Fig. 3. Time needed to send a given volume of data for different values of ρ and of the initial window W_0.

Fig. 3, using (15), shows the time needed to complete transmission as a function of the data volume, for initial windows of 1, 2 and 3 packets. Three sets of curves are plotted: one for a value of ρ corresponding to our actual experimental setup, one with $D_a = \tau \Rightarrow \rho = 2$, that is an ideal condition where the allocation delay is as small as theoretically possible for BoD, and one with $D_a = 0 \Rightarrow \rho = 1$, which means no allocation delay, that is the same as a statically preassigned bandwidth share. The value of ρ corresponding to the experimental Skyplex data platform is computed from parameters measured in real tests: $\tau = 750$ ms, $T_r = 273$ ms, $T_a = 819$ ms and $k = 1$. The symbols on the curves mark the moment t_s when saturation is reached.

The vertical separation between curves with different ρ values and same initial window is the delay introduced by the BoD mechanism. The graph highlights that BoD increases significantly the completion time of short connections and delays the moment when channel saturation is reached, resulting in several seconds of poor bandwidth utilization.

The initial window is another parameter influencing the startup performance. In fact, choosing a larger initial window



(a) Connection duration

(b) Assigned slot utilization

Fig. 4. Simulation for varying k factor

may save up to two transmission RTT with minimal impact on network congestion [7]. Since the RTT over satellite links is much larger than in terrestrial networks, a larger W_0 can lead to a considerable time saving. We stress here that this effect is even greater in the presence of rate-based BoD policies, which have the same effect as increasing the RTT.

V. Considerations on parameters choice

As previously noted, the harmful effect of BoD on TCP startup could be compensated by choosing a larger value of the ratio k between allocated and requested bandwidth. Ideally, the BoD scheduler should provide to a starting TCP connection enough bandwidth to double the congestion window every RTT so to increase the throughput at the fastest possible rate.

The condition allowing the TCP connection to exploit the assigned share without buffering and without wasted channel resources is that the TCP throughput be equal to the assigned share

$$\frac{w(t)}{\tau} = k\frac{w(t-T)}{\tau}, \tag{16}$$

which translates into $k = \exp(Ts_2)$ which, for $\rho = 2.73$ as in the case of Skyplex Data, implies $k = 4.8$.

These considerations are backed by the traces in Fig. 4 that illustrates a parametric simulative experiment where we evaluate the effect of changing the k factor on connection duration for several volumes of data to transmit and on utilization of assigned channel share. Lines plotted in Fig. 4 are averaged over all the possible offsets of TCP starting time with respect to DVB frame. As expected, completion time improves with increasing k, and gain diminishes up to a value of k near 4.8, as previously computed. Improvements are most significant for bigger data volumes.

On the other hand, channel utilization worsens with increasing k, and worsening is most significant for smaller volumes of data.

VI. Conclusions

Experimental and simulative studies on Eutelsat's Skyplex Data system highlighted how rate-based Bandwidth on demand systems can significantly slow down the transmission of small volumes of data using TCP. This observation is significant because most transmissions on the Internet are made of short TCP connections, and because BoD satellite systems are going to be more and more diffused.

We presented an analytical modeling of this phenomenon, which is similar to what would happen on a traditional fixed-allocation satellite system if the round-trip time was increased. We show that a small change to the allocation policy can bring significant benefits to short TCP connections, but that there are also drawbacks in terms of channel utilization, an important matter because of the costs associated to satellite bandwidth.

Further studies are required in order to modify the basic rate-based allocation algorithm to make it more responsive to starting TCP connections and similar types of traffic, such as those based on TCP Friendly Rate Based (TFRC) algorithms.

References

[1] N. Celandroni, F. Davoli, E. Ferro, and A. Gotta. Tcp performace measured over wireless integrated networks with high delay-bandwidth products. In *proc. of ASMS 2006*, May 2006.

[2] M. Sooriyabandara and G. Fairhurst. Dynamics of tcp over bod satellite networks. *Int. Journal of Satellite Comunication and Networking*, 21(4–5):427–449, July 2003.

[3] M. Karaliopoulos, R. Tafazolli, and B. Evans. On the interaction of tcp with bod in geo broadband satellite networks. In *proc. IEEE Globecom 2002*, November 2002.

[4] E. Feltrin, E. Weller, E. Martin, and K. Zamani. Design, implementation and performance analysis of an on board processor-based satellite network. In *ICC04*, volume 6, pages 3321–3325, June 2004.

[5] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi, and M. Gerla. Tcp startup performance in large bandwidth delay networks. In *INFOCOMM*, pages 796–805, 2004.

[6] Alberto Gotta, Raffaello Secchi, and Francesco Potort. Simulating dynamic bandwidth allocation on satellite links. In *proceedings of the International Conference on Performance Evaluation Methodologies and Tools (Valuetools)*, page to appear, Pisa (IT), October 2006. ACM. WNS2 workshop.

[7] M. Postel, S. Floyd, and C. Partridge. Increasing tcp's initial window, October 2002. RFC 3390.