

Can a Satellite be an Internet Router?

Aaron Falk, Nirav Jasapara

Information Sciences Institute, University of Southern California

falk@isi.edu, jasapara@usc.edu

Abstract

Geosynchronous satellite systems have carried Internet traffic since the earliest days of packet switching. In recent years, both satellite capabilities and users' expectations have increased, motivating new designs in which satellite payloads and systems participate at higher layers of the protocol stack. Making a satellite system a fully functional Internet router has challenges because satellite communication characteristics are mismatched to more common Internet link technologies in several important ways. These include long propagation delays, link intermittency, and mismatches between uplink and downlink terminal reachability. These deviations have implications on system, network, and on-board router designs. In some cases, the protocol challenges imposed by satellite systems are similar to those created by other environments and solutions from those environments may be reused. In other cases these solutions may not be applicable, due to fundamental performance differences or spacecraft hardware constraints. In this paper we examine the protocol effects of various IP over satellite architectural components and discuss the likelihood that solutions from other kinds of networks may be helpful.

1. Introduction

Current satellite systems support Internet services as simple data pipes. Systems in development use on-board IP packet switching to exploit statistical multiplexing and make better use of limited downlink capacity. By adding multiplexing, forwarding, and routing functions to a satellite it begins to take the form of an Internet router. However, transforming a satellite into an Internet router is a tricky task.

Satellite systems and Internet routers are designed with different high-level objectives. Satellite systems are typically designed to squeeze as many bits of capacity out of a link as is possible while contending with limitations of the spectrum and payload size, weight, and power constraints. In contrast, Internet routers have been designed primarily for efficient transfer of IP packets, often ignoring link efficiency.

A satellite system designer wishing to get good performance and link utilization when carrying Internet traffic will need to develop a good understanding of the TCP/IP protocol suite and its expectations of the underlying link technologies. Some areas which deserve attention are correct choice of maximum transmission unit and framing, interactions between link bandwidth-on-demand and TCP, mechanisms for link layer reliability and data compression, and the effects of bandwidth asymmetry and packet re-ordering. These are thoroughly cataloged in the IETF Best Current Practice RFC “Advice to Subnetwork Designers” [10]. In this paper, however, we concentrate on *architectural characteristics* of satellite networks that make it challenging to support Internet protocols such as link intermittency and delay.

Some networks have protocol adaptations that may be applicable to satellite networks. However, this is not always the case.

We start by reviewing the key design constraints of satellite systems in Section 2. Then, in Section 3, we discuss several architectural characteristics, their effects on Internet protocols, and evaluate solutions from similar architectures. Finally, in Section 4, we chose one aspect of performance — TCP throughput — and compare several performance optimizations developed for high-speed networks, showing some protocols perform much better over satellite delays than others.

2. Fundamental Differences Between Satellites and Terrestrial Internet Routers

Satellites are designed with a unique set of design constraints. Minimizing payload complexity is critical. Complex satellite payloads require additional size, weight, and power, are more likely to fail, and may not meet future requirements¹. It is much more expensive to put computational power on a satellite than elsewhere. So, there is a strong incentive for satellite designers to place as much functionality on the ground as possible. This creates a set of tradeoffs between payload complexity and performance

¹Satellite hardware is hard to upgrade once it is in orbit. So, functions placed on-board need to be flexible or unlikely to change.

or flexibility. For example, one design approach for a satellite system with an on-board router would be to consider the on-board portion to do only “fast path” functions and put all other routing functions on the ground.

Functions in hardware need to last the lifetime of a satellite. So, on-board capabilities either use power-intensive general purpose processors, to allow for change, or must be very carefully chosen to avoid obsolescence.

3. Architectural Differences

In this section we discuss how satellites differ from conventional Internet routers. In particular, we compare satellite links to Ethernet, the de-facto Internet link technology. Internet protocols tend to assume link capabilities similar to those of Ethernet. As new link technologies are developed, link and router designers sometimes need to add functions that are expected by the Internet Protocol and higher layers (see [10]). For example, to perform IPv4 to MAC address translation a node needs to use ARP [13] and ensure a broadcast capability exists on the link or to build another address resolution protocol. (The expectation of a broadcast links is not limited to ARP. See Section 3.3.)

Now we consider the satellite architectural characteristics of delay, intermittent links, and mismatches between uplink and downlink reachability. In each case we discuss the impacts on Internet protocols and highlight the other environments that share similar characteristics and may be exploited.

3.1. Delay

The propagation delay over a geosynchronous satellites is roughly 250ms from terminal to terminal. This delay is increased by the delays for forward error correction encoding and possibly capacity acquisition (in a system where uplinks are shared). The resulting round-trip delays over a GEO system can be well over one second.

The implications are twofold: first, when considering offloading control-plane functions from an on-board packet switch to ground equipment, the delay will degrade the responsiveness of the system. The system designer must find a balance between the responsiveness and payload complexity. In other words, to maximize responsiveness, offload only relatively slowly changing functions. However, retaining functions on-board which require rapid control may be processing intensive (utilizing precious size, weight, and power).

Second, large round-trip times contribute to large bandwidth-delay products and cause TCP to be less able to make use of available bandwidth [2]. There are two aspects of this problem:

Flows in Startup After Unexpected Loss In TCP’s congestion avoidance phase, high bandwidth flows take a *very* long time to recover when a packet is lost, whether from corruption or congestion. For example, a 1Gbps TCP flow over a path with a 500ms RTT would take nearly 6 hours to recover from a packet loss. A large BDP requires a large congestion window and filling it requires the same number of round trips regardless of end-to-end delay. However, as the round-trip time increases, the clock time required to achieve such a large BDP gets very large. Another aspect of this problem is that, when congestion is experienced in a network where large and small RTT flows share a bottleneck, it takes the long RTT flows much longer to learn of the congestion and respond. This means that long RTT flows will continue to increase their send rate (albiet slowly) at a time when the network needs flows to back off. Conversely, the long RTT flows reduce their rate after the congested period has likely passed. *So, long RTT flows do the opposite of the desired behavior.*

Flows in Steady State The probing nature of Van Jacobson congestion control used in TCP means that even under optimal conditions (i.e., low packet loss) links carrying a small number of large BDP TCP flows will have poor link utilization in steady state. Further, TCP’s notion of fairness is not preserved when flows of mixed RTT compete for capacity in the same bottleneck [7]. This means that long RTT flows will get less than their fair share when competing against short RTT flows.

Satellites are not the only large BDP networks. High-performance networks, such as those used by the OptIPuter [5] project, are used to move Petabytes of data over long (terrestrial) distances. They tend to have expensive, dedicated fiber links and wish to make full use of them. Many changes to TCP congestion control have been developed to make better use of the multi-Gigabit links these network use. Protocols like BIC-TCP [15], FAST [9], and XCP [11] [8] can be effective at allowing large BDP flows such as these to make good use of link capacity.

However, we have found that when delay dominates the bandwidth-delay product, some of these alternate congestion control schemes do not perform as well. This will be illustrated in §4.

3.2. Link Intermittency

Internet routers assume links fail infrequently. In contrast, a next hop router in a satellite network may be unreachable for a number of reasons. The terminal may have moved out of the beam area, the line of sight to the satellite

may be blocked by ground clutter, or signal strength may be degraded by weather, low transmitter power, poor terminal aiming, or interference from other transmitters. These mechanisms can cause a link to be intermittent with duty cycles ranging from milliseconds to tens of seconds.

Frequent transitions between link-up and link-down states can degrade routing due to excessive route updates (“flaps”). For example, BGP by default sends a route withdraw message 90 seconds after detection of a link failure[14]. Adding hysteresis to routing updates for intermittent links will make the routing topology more stable at the risk of continuing to advertise truly lost routes for a longer period.

Link intermittency degrades TCP performance, as well. When a satellite link goes down, the TCP sender is unable to receive acknowledgments and will eventually time-out and halt. The connection then moves into a phase known as ‘exponential backoff’ where probes are sent periodically to see if packets can flow again. The period in between probes doubles each time, starting around 3 RTTs and with a maximum of around 2 minutes. Once an intermittent link has been restored, a flow in exponential backoff won’t send a probe (leading to a restart of the flow) until the timer expires, i.e., possibly as long as 2 minutes after the link is restored. So, even brief link outages can cause TCP flows to take a long time to complete.²

Motivated by terrestrial wireless services (that want TCP connections to restore quickly when an automobile emerges from a tunnel), there are currently proposals in the IETF to allow end-systems to negotiate a reduced value for the maximum period between probes (‘MaxRTO’)[4]. Such a mechanism may be useful when end-systems are aware that an intermittent link may be in the path. Another potential mechanism under discussion in the IETF is a mechanism for notifying end-systems of link state changes [1]. An end-system receiving a link-up notification might send a probe immediately rather than waiting for the probe timer to expire.

3.3. Mismatches in Link Connectivity

Links tend to be either point-to-point or shared. Telephony tends to use point-to-point links. Ethernet links tend to be shared. Satellite networks are a hybrid: uplinks tend to be point-to-point and downlinks tend to be shared. This creates some complexity in supporting Internet protocols.

A satellite *beam* contains logical channels between the satellite and a set of terminals within the beams coverage area (see 1). Packet switching satellites forward packets to a logical channel that contains the appropriate next hop. Downlink beams may cover geographic areas hun-

²This situation is even worse if the links to both terminals are independently intermittent.

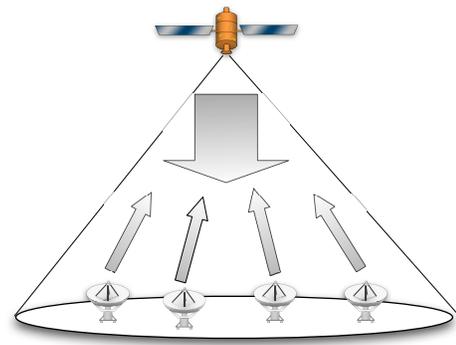


Figure 1. Multiple uplinks can be found in a single downlink.

dreds or thousands of miles in diameter and include very large numbers of terminals. Uplink beams may carry traffic from a single terminal to the satellite or be shared using time-, frequency-, or code-division techniques. Whether an uplink logical channel (sometimes just ‘uplink’) may be shared by more than one terminal is satellite-design dependent. However, downlinks will usually reach more terminals than uplinks.

It should be clear, then, that satellite channels have differences from full-duplex, shared media such as Ethernet. In a downlink channel, the satellite can ‘broadcast’³ to a set of terminals on a logical channel. But the terminals are not capable of broadcasting directly to the other terminals that the downlink can reach. Also, the association of uplink beams to downlink beams may not be fixed. In other words, the terminals may move from one uplink channel to another, independent of their presence in a particular downlink beam. For example, a terminal may switch uplinks to increase or decrease its available transmit rate. Conversely, a terminal may move from one downlink beam to another but retain the same uplink beam. For example, a mobile terminal with a dedicated uplink may move from one downlink coverage area to another, requiring a change in logical channel.

Unicast vs. Multicast. For IPv4, protocols like ARP [13] and DHCP [3] expect a broadcast channel. For IPv6, Neighbor Discovery [12] (providing similar functions) requires multicast. Given the discussion in the previous section, it is tempting to view the satellite-based packet switch as a collection of uplinks (loosely) associated with a downlink as a shared medium. But this is problematic. Satellite-based routers can multicast very efficiently on the downlink to all the terminals on the channel. However, terminals cannot reach other terminals in that downlink logical-channel without going through the satellite. In other words, termi-

³Multicast, actually, since not all the terminals connected to the satellite will be in the downlink logical channel.

nals cannot multicast directly to other terminals.

The architectural challenges of providing multicast for satellite networks are similar to those for cable modem systems for Internet access. Cable systems typically have a head-end broadcasting a multiplexed IP data-stream to a set of residential set-top boxes (STBs). The STBs typically have a unicast upstream channel to a head-end router, seemingly similar to a satellite uplink. Cable systems, however, have avoided implementing broadcast and multicast from the STB. There is an implicit assumption that since STBs tend to be at the edge of the network, they will send all their packets to the upstream router, removing the need to implement broadcast for ARP or DHCP. Nevertheless, there are indications that cable operators are growing more interested in true multicast to support corporate VPN traffic. [6]. The solution appears to be that cable system operators will provide multicast by forcing STB-to-STB traffic to pass through the head-end router, which would presumably replicate multicast packets destined for other STBs in the head-ends 'coverage area'. Simulating multicast by routing all packets through the a hub fits well for satellite systems since even unicast data needs to pass over the satellite to reach other terminals.

Reachability. If a satellite-based packet switch is route, it needs to generate and receive routing advertisements to and from adjacent routers. A terminal-router would send the satellite-router an advertisement on an uplink. The satellite-router then needs to apply that route advertisement to the downlink associated with the advertising terminal. As discussed in above, there is a complex and possibly dynamic mapping between uplink and downlink logical channels. Also, a geosynchronous satellite can view a large geographical area and so may be connected to a *very* large number of terminal routers. There could be very many updates and a very large routing table. The routing function, i.e., the computation of next-hop forwarding mappings, may be a good candidate function for offloading to ground systems. As discussed earlier, this makes the system less responsive to rapid changes – a rapidly moving terminal, for instance.

The cellular phone industry may offer a lesson for managing mobility. Most current systems, GPRS, for example, confine mobility management to the link layer. One way to apply this solution would be to have terminals continue to stay in the same logical channel as long as possible.

4. High-speed Transport Protocol Performance over Satellite

In this section we look in more detail at the impacts of delay on performance over satellite networks. The topic of improving TCP performance over large bandwidth-delay

products has been studied quite a bit and it is reasonable to expect that there may be existing solutions which would benefit satellite networks. We look at bottleneck link utilization and fairness in a simulated satellite network with competing short RTT flows. It is a typical and interesting scenario to have flows traversing satellite networks compete with flows that do not.

We evaluate three candidate high-speed congestion control algorithms: BIC-TCP [15], FAST TCP[9], and XCP [11]. We chose these algorithms because they represent congestion control variant gaining a great deal of interest in the high-speed networking community.

We summarize the algorithms below:

TCP The NewReno variant of TCP is current standard for TCP congestion control.

BIC-TCP The protocol combines two schemes called additive increase and binary search increase. When the congestion window is large, additive increase with a large increment ensures linear RTT fairness as well as good scalability. Under small congestion windows, binary search increase is designed to provide TCP friendliness.

FAST TCP This protocol is based on Vegas TCP. It uses queuing delay, rather than loss probability, as the main measure of congestion.

XCP eXplicit Control Protocol (XCP) packets carry a congestion header through which the sender requests a desired throughput. Routers participate in the algorithm and mark each packet with a fair per-flow capacity allocation but do not maintain per-flow state. Thus, the sender learns of the bottleneck router's allocation in a single round trip.

These three algorithms represent three different approaches to congestion control. BIC-TCP is a small change to additive-increase, multiplicative-decrease control law in TCP and is a small change to TCP congestion control. FAST uses changes in RTT to detect queues (and hence congestion) developing in the network. FAST is a significant change in the congestion control algorithm but still uses implicit feedback from the packet stream to identify network congestion. XCP, on the other hand, uses explicit feedback from routers to set sender's rate. XCP represents a significant departure from prior TCP congestion control approaches and provides more rapid, accurate information on the state of the network. Versions of the algorithms for the NS2 simulator were available to us. ⁴

⁴All three algorithms have continued to evolve since the ns2 versions were released. However, we used the most current versions which were available.

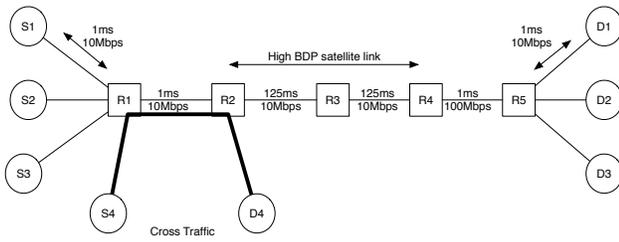


Figure 2. The simulation topology.

4.1. Simulation Environment

Figure 2 shows the the simulation topology used in our analysis. Three 508ms RTT flows start and compete for capacity with a single 6ms RTT flow. The short delay link nearest the sources is set to 10Mbps, creating a bottleneck at R1.

The maximum queue size at the bottleneck router was set to twice the BDP of the network. Flows are infinite length, have 1040 byte packets and start times are staggered by 5 sec.

We use the recommended parameters for each algorithm. In most cases these were the defaults. However, FAST uses two parameters, alpha and beta, for congestion window estimation. Alpha and beta indicate the minimum and the maximum number of packets that a source tries to maintain in the bottleneck router. Setting alpha and beta to an optimal value is important, otherwise the performance is very poor. The FAST authors recommend setting alpha to 2C where C is the capacity of the path.⁵ When we ran tests using FAST, we chose an alpha value that resulted in near 100% link utilization at steady state and minimal packet losses.

We ran the simulations for 100 seconds and collected congestion window, link utilization, queue length, and packet loss data. (Paper length constraints prohibit analysis of queue length and packet loss.) We present sender throughput derived by averaging the congestion window over an RTT. Link utilization is obtained by counting packet sizes received over a link during a one second interval and dividing by the link rate.

4.2. Analysis

When long RTT TCP flows compete with a short RTT flow (see Fig. 3), the short RTT flow is able to quickly acquire unused bottleneck capacity released by the long RTT flows as they back off following packet loss. The

⁵In practice C is not known. The FAST authors suggest automatic alpha tuning wherein the source detects the maximum throughput achieved and then sets alpha using a table which maps alpha-values to network capacities. How this will work in a dynamic network with varying number of sources and shifting bottlenecks is not clear. Automatic alpha tuning support was not available in NS.

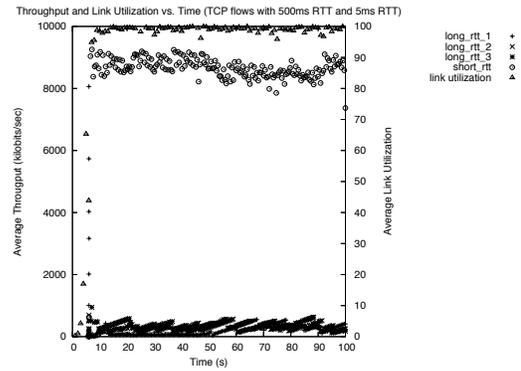


Figure 3. TCP Throughput

short RTT flow appears more aggressive, preventing the long RTT flows from acquiring a fair share of the capacity. The result is high bottleneck link utilization but low fairness. [7] explores RTT unfairness further.⁶

TCP-BIC seems to yield worse throughput (& fairness) to the long RTT flows than TCP (see Fig. 4). The short RTT flow quickly grabs and sustains over 90% of the link capacity. At around 87 seconds, it appears that some packet losses in the short RTT flow allow the long RTT flow to acquire some capacity.

FAST, in contrast, shares better with flows of different RTT but, after 100 seconds of simulation time, the allocations were still far from fair, with the low RTT flow acquiring about half the link capacity and the remaining three flows sharing the rest. (See Fig. 5.) Compared to TCP and TCP-BIC, convergence to good link utilization takes somewhat longer (about 10 sec. for BIC and 40 sec. for FAST). This appears to be the low RTT FAST sender slowly releasing capacity to the long RTT flows.

XCP performs the best in the presence of flows with diverse RTT. We see in Fig. 6 that XCP achieves 100% utilization at about 5 sec. and converges by roughly 20 sec. to an equal share of bottleneck link capacity between flows.

5. Conclusion

Can a satellite system be an Internet router? The answer is fundamentally yes, but there are mismatches between satellites and Internet routers and designers need to understand the protocols to find the best ways to address them. Other link technologies have faced similar issues and, in some cases, solutions exist which can benefit satellite systems. However, this is not always true.

In the case of TCP throughput and link utilization, when the delay dominates the bandwidth-delay product, we have shown that some performance enhancements for large

⁶Note that the high link utilization is for the terrestrial link. The satellite flows remain at very low throughput and, since they are getting less than their fair share, the satellite link has degraded utilization.

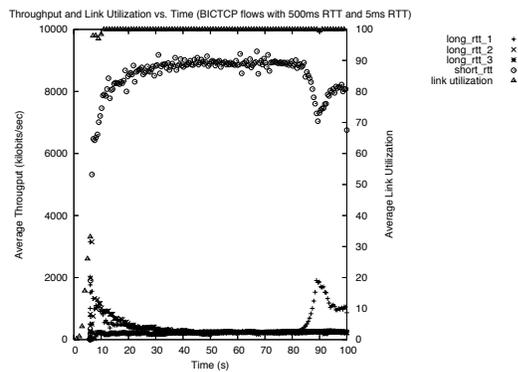


Figure 4. BIC Throughput

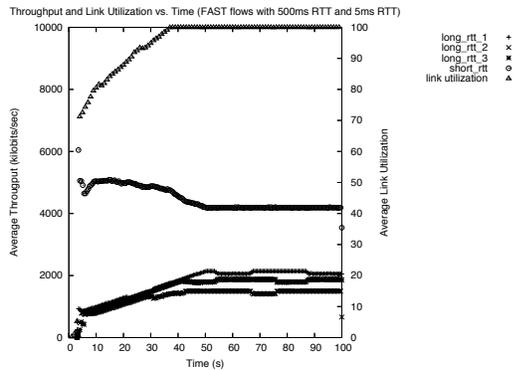


Figure 5. FAST Throughput

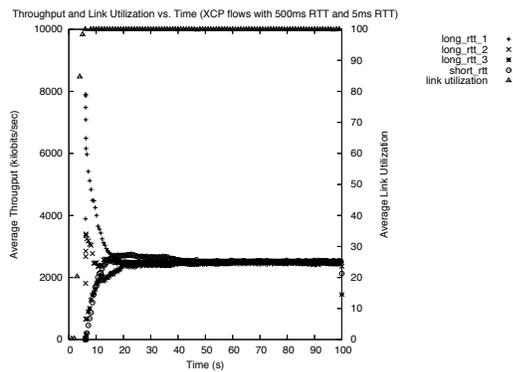


Figure 6. XCP Throughput

BDP networks perform poorly. TCP-BIC performed only slightly better than TCP. FAST was somewhat better but required hand configuration. XCP, however, demonstrated good link utilization and fair capacity allocation among flows of similar and different RTT.

Naturally, satellite designers need to consider the size, weight, power and complexity implications of putting XCP functionality on a satellite-based packet switch. Early indications are that XCP can be implemented on a router with small number of instructions. [11] states that XCP can be

implemented with “a few additions and 3 multiplications per packet”.⁷ So, while XCP would require participation by a satellite-based router, it may show promise for use in satellite networks.

References

- [1] B. Aboba and Internet Architecture Board. Architectural implications of link indications. Work in progress, July 2005.
- [2] M. Allman, D. Glover, and L. Sanchez. Enhancing TCP Over Satellite Channels using Standard Mechanisms. RFC 2488 (Best Current Practice), January 1999.
- [3] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), March 1997. Updated by RFC 3396.
- [4] L. Eggert and F. Gont. Tcp user timeout option. Work in progress, 2005.
- [5] Aaron Falk, Ted Faber, Joseph Bannister, Andrew Chien, Robert Grossman, and Jason Leigh. Transport protocols for high performance. *Commun. ACM*, 46(11):42–49, 2003.
- [6] Joe Godas, Brian Field, Alon Bernstein, Sanjeev Desai, Torless Eckert, and Harsh Parandekar. Ip multicast in cable data networks. White paper, Cisco Systems, San Jose, CA, 2005.
- [7] Tom Henderson, Emile Sahouria, Steven McCanne, and Randy Katz. On improving the fairness of tcp congestion avoidance. In *Proc. IEEE Globecom '98*, Sydney, Australia, November 1998. <http://daedalus.cs.berkeley.edu/publications/globe98.ps.gz>.
- [8] USC Information Sciences Institute. Xcp @ isi: The explicit control protocol. project web page, 2005. <http://www.isi.edu/isi-xcp>.
- [9] C. Jin, D. Wei, and S. Low. Fast tcp: Motivation, architecture, algorithms, performance, 2004.
- [10] P. Karn, C. Bormann, G. Fairhurst, D. Grossman, R. Ludwig, J. Mahdavi, G. Montenegro, J. Touch, and L. Wood. Advice for Internet Subnetwork Designers. RFC 3819 (Best Current Practice), July 2004.
- [11] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments, 2002.
- [12] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461 (Draft Standard), December 1998.
- [13] D.C. Plummer. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. RFC 826 (Standard), November 1982.
- [14] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), March 1995.
- [15] “Lisong Xu, Khaled Harfoush, and Injong Rhee”. “binary increase congestion control for fast long-distance networks”. In *“IEEE Infocom”*. IEEE, 2004.

⁷As part of our XCP development work at ISI[8], we are validating this claim by implementing XCP on a network processor.