Network Working Group
Request for Comments: 2757
Category: Informational

G. Montenegro
Sun Microsystems, Inc.
S. Dawkins
Nortel Networks
M. Kojo
University of Helsinki
V. Magret
Alcatel
N. Vaidya
Texas A&M University

January 2000

Long Thin Networks

Status of this Memo

Copyright Notice

Abstract

In view of the unpredictable and problematic nature of long thin networks (for example, wireless WANs), arriving at an optimized transport is a daunting task. We have reviewed the existing proposals along with future research items. Based on this overview, we also recommend mechanisms for implementation in long thin networks.

Our goal is to identify a TCP that works for all users, including users of long thin networks. We started from the working recommendations of the IETF TCP Over Satellite Links (tcpsat) working group with this end in mind.

We recognize that not every tcpsat recommendation will be required for long thin networks as well, and work toward a set of TCP recommendations that are 'benign' in environments that do not require them.

Table of Contents

1 Introduction

Optimized wireless networking is one of the major hurdles that Mobile Computing must solve if it is to enable ubiquitous access to networking resources. However, current data networking protocols have been optimized primarily for wired networks. Wireless environments have very different characteristics in terms of latency, jitter, and error rate as compared to wired networks. Accordingly, traditional protocols are ill-suited to this medium.

Mobile Wireless networks can be grouped in W-LANs (for example, 802.11 compliant networks) and W-WANs (for example, CDPD [CDPD], Ricochet, CDMA [CDMA], PHS, DoCoMo, GSM [GSM] to name a few). W-WANs present the most serious challenge, given that the length of the wireless link (expressed as the delay*bandwidth product) is typically 4 to 5 times as long as that of its W-LAN counterparts. For example, for an 802.11 network, assuming the delay (round-trip time) is about 3 ms. and the bandwidth is 1.5 Mbps, the delay*bandwidth product is 4500 bits. For a W-WAN such as Ricochet, a typical round-trip time may be around 500 ms. (the best is about 230 ms.), and the sustained bandwidth is about 24 Kbps. This yields a delay*bandwidth product roughly equal to 1.5 KB. In the near future, 3rd Generation wireless services will offer 384Kbps and more. Assuming a 200 ms round-trip, the delay*bandwidth product in this case is 76.8 Kbits (9.6 KB). This value is larger than the default 8KB buffer space used by many TCP implementations. This means that, whereas for W-LANs the default buffer space is enough, future W-WANs will operate inefficiently (that is, they will not be able to fill the pipe) unless they override the default value. A 3rd Generation wireless service offering 2 Mbps with 200-millisecond latency requires a 50 KB buffer.

Most importantly, latency across a link adversely affects throughput. For example, [MSMO97] derives an upper bound on TCP throughput. Indeed, the resultant expression is inversely related to the round-trip time.

The long latencies also push the limits (and commonly transgress them) for what is acceptable to users of interactive applications.

As a quick glance to our list of references will reveal, there is a wealth of proposals that attempt to solve the wireless networking problem. In this document, we survey the different solutions available or under investigation, and issue the corresponding recommendations.

There is a large body of work on the subject of improving TCP performance over satellite links. The documents under development by the tcpsat working group of the IETF [AGS98, ADGGHOSSTT98] are very relevant. In both cases, it is essential to start by improving the characteristics of the medium by using forward error correction (FEC) at the link layer to reduce the BER (bit error rate) from values as high as 10-3 to 10-6 or better. This makes the BER manageable. Once in this realm, retransmission schemes like ARQ (automatic repeat request) may be used to bring it down even further. Notice that sometimes it may be desirable to forego ARQ because of the additional delay it implies. In particular, time sensitive traffic (video, audio) must be delivered within a certain time limit beyond which the data is obsolete. Exhaustive retransmissions in this case merely succeed in wasting time in order to deliver data that will be discarded once it arrives at its destination. This indicates the desirability of augmenting the protocol stack implementation on devices such that the upper protocol layers can inform the link and MAC layer when to avoid such costly retransmission schemes.

Networks that include satellite links are examples of "long fat networks" (LFNs or "elephants"). They are "long" networks because their round-trip time is quite high (for example, 0.5 sec and higher for geosynchronous satellites). Not all satellite links fall within the LFN regime. In particular, round-trip times in a low-earth orbiting (LEO) satellite network may be as little as a few milliseconds (and never extend beyond 160 to 200 ms). W-WANs share the "L" with LFNs. However, satellite networks are also "fat" in the sense that they may have high bandwidth. Satellite networks may often have a delay*bandwidth product above 64 KBytes, in which case they pose additional problems to TCP [TCPHP]. W-WANs do not generally exhibit this behavior. Accordingly, this document only deals with links that are "long thin pipes", and the networks that contain them: "long thin networks". We call these "LTNs".

This document does not give an overview of the API used to access the underlying transport. We believe this is an orthogonal issue, even though some of the proposals below have been put forth assuming a given interface. It is possible, for example, to support the traditional socket semantics without fully relying on TCP/IP transport [MOWGLI].

Our focus is on the on-the-wire protocols. We try to include the most relevant ones and briefly (given that we provide the references needed for further study) mention their most salient points.

1.1 Network Architecture

One significant difference between LFNs and LTNs is that we assume the W-WAN link is the last hop to the end user. This allows us to assume that a single intermediate node sees all packets transferred between the wireless mobile device and the rest of the Internet. This is only one of the topologies considered by the TCP Satellite community.

Given our focus on mobile wireless applications, we only consider a very specific architecture that includes:

- a wireless mobile device, connected via
- a wireless link (which may, in fact comprise several hops at the link layer), to
- an intermediate node (sometimes referred to as a base station) connected via
- a wireline link, which in turn interfaces with
- the landline Internet and millions of legacy servers and web sites.

Specifically, we are not as concerned with paths that include two wireless segments separated by a wired one. This may occur, for example, if one mobile device connects across its immediate wireless segment via an intermediate node to the Internet, and then via a second wireless segment to another mobile device. Quite often, mobile devices connect to a legacy server on the wired Internet.

Typically, the endpoints of the wireless segment are the intermediate node and the mobile device. However, the latter may be a wireless router to a mobile network. This is also important and has applications in, for example, disaster recovery.

Our target architecture has implications which concern the deployability of candidate solutions. In particular, an important requirement is that we cannot alter the networking stack on the legacy servers. It would be preferable to only change the networking stack at the intermediate node, although changing it at the mobile devices is certainly an option and perhaps a necessity.

We envision mobile devices that can use the wireless medium very efficiently, but overcome some of its traditional constraints. That is, full mobility implies that the devices have the flexibility and agility to use whichever happens to be the best network connection

available at any given point in time or space. Accordingly, devices could switch from a wired office LAN and hand over their ongoing connections to continue on, say, a wireless WAN. This type of agility also requires Mobile IP [RFC2002].

1.2 Assumptions about the Radio Link

The system architecture described above assumes at most one wireless link (perhaps comprising more than one wireless hop). However, this is not enough to characterize a wireless link. Additional considerations are:

- What are the error characteristics of the wireless medium? The link may present a higher BER than a wireline network due to burst errors and disconnections. The techniques below usually do not address all the types of errors. Accordingly, a complete solution should combine the best of all the proposals. Nevertheless, in this document we are more concerned with (and give preference to solving) the most typical case: (1) higher BER due to random errors (which implies longer and more variable delays due to link-layer error corrections and retransmissions) rather than (2) an interruption in service due to a handoff or a disconnection. The latter are also important and we do include relevant proposals in this survey.

- Is the wireless service datagram oriented, or is it a virtual circuit? Currently, switched virtual circuits are more common, but packet networks are starting to appear, for example, Metricom's Starmode [CB96], CDPD [CDPD] and General Packet Radio Service (GPRS) [GPRS],[BW97] in GSM.

- What kind of reliability does the link provide? Wireless services typically retransmit a packet (frame) until it has been acknowledged by the target. They may allow the user to turn off this behavior. For example, GSM allows RLP [RLP] (Radio Link Protocol) to be turned off. Metricom has a similar "lightweight" mode. In GSM RLP, a frame is retransmitted until the maximum number of retransmissions (protocol parameter) is reached. What happens when this limit is reached is determined by the telecom operator: the physical link connection is either disconnected or a link reset is enforced where the sequence numbers are resynchronized and the transmit and receive buffers are flushed resulting in lost data. Some wireless services, like CDMA IS95-RLP [CDMA, Karn93], limit the latency on the wireless link by retransmitting a frame only a couple of times. This decreases the residual frame error rate significantly, but does not provide fully reliable link service.

- Does the mobile device transmit and receive at the same time? Doing so increases the cost of the electronics on the mobile device. Typically, this is not the case. We assume in this document that mobile devices do not transmit and receive simultaneously.

- Does the mobile device directly address more than one peer on the wireless link? Packets to each different peer may traverse spatially distinct wireless paths. Accordingly, the path to each peer may exhibit very different characteristics. Quite commonly, the mobile device addresses only one peer (the intermediate node) at any given point in time. When this is not the case, techniques such as Channel-State Dependent Packet Scheduling come into play (see the section "Packet Scheduling" below).

2 Should it be IP or Not?

The first decision is whether to use IP as the underlying network protocol or not. In particular, some data protocols evolved from wireless telephony are not always -- though at times they may be -- layered on top of IP [MOWGLI, WAP]. These proposals are based on the concept of proxies that provide adaptation services between the wireless and wireline segments.

This is a reasonable model for mobile devices that always communicate through the proxy. However, we expect many wireless mobile devices to utilize wireline networks whenever they are available. This model closely follows current laptop usage patterns: devices typically utilize LANs, and only resort to dial-up access when "out of the office."

For these devices, an architecture that assumes IP is the best approach, because it will be required for communications that do not traverse the intermediate node (for example, upon reconnection to a W-LAN or a 10BaseT network at the office).

2.1 Underlying Network Error Characteristics

Using IP as the underlying network protocol requires a certain (low) level of link robustness that is expected of wireless links.

IP, and the protocols that are carried in IP packets, are protected end-to-end by checksums that are relatively weak [Stevens94, Paxson97] (and, in some cases, optional). For much of the Internet, these checksums are sufficient; in wireless environments, the error characteristics of the raw wireless link are much less robust than the rest of the end-to-end path. Hence for paths that include

wireless links, exclusively relying on end-to-end mechanisms to detect and correct transmission errors is undesirable. These should be complemented by local link-level mechanisms. Otherwise, damaged IP packets are propagated through the network only to be discarded at the destination host. For example, intermediate routers are required to check the IP header checksum, but not the UDP or TCP checksums. Accordingly, when the payload of an IP packet is corrupted, this is not detected until the packet arrives at its ultimate destination.

A better approach is to use link-layer mechanisms such as FEC, retransmissions, and so on in order to improve the characteristics of the wireless link and present a much more reliable service to IP. This approach has been taken by CDPD, Ricochet and CDMA.

This approach is roughly analogous to the successful deployment of Point-to-Point Protocol (PPP), with robust framing and 16-bit checksumming, on wireline networks as a replacement for the Serial Line Interface Protocol (SLIP), with only a single framing byte and no checksumming.

[AGS98] recommends the use of FEC in satellite environments.

Notice that the link-layer could adapt its frame size to the prevalent BER. It would perform its own fragmentation and reassembly so that IP could still enjoy a large enough MTU size [LS98].

A common concern for using IP as a transport is the header overhead it implies. Typically, the underlying link-layer appears as PPP [RFC1661] to the IP layer above. This allows for header compression schemes [IPHC, IPHC-RTP, IPHC-PPP] which greatly alleviate the problem.

2.2 Non-IP Alternatives

A number of non-IP alternatives aimed at wireless environments have been proposed. One representative proposal is discussed here.

2.2.1 WAP

The Wireless Application Protocol (WAP) specifies an application framework and network protocols for wireless devices such as mobile telephones, pagers, and PDAs [WAP]. The architecture requires a proxy between the mobile device and the server. The WAP protocol stack is layered over a datagram transport service. Such a service is provided by most wireless networks; for example, IS-136, GSM SMS/USSD, and UDP in IP networks like CDPD and GSM GPRS. The core of

the WAP protocols is a binary HTTP/1.1 protocol with additional features such as header caching between requests and a shared state between client and server.

2.2.2 Deploying Non-IP Alternatives

IP is such a fundamental element of the Internet that non-IP alternatives face substantial obstacles to deployment, because they do not exploit the IP infrastructure. Any non-IP alternative that is used to provide gatewayed access to the Internet must map between IP addresses and non-IP addresses, must terminate IP-level security at a gateway, and cannot use IP-oriented discovery protocols (Dynamic Host Configuration Protocol, Domain Name Services, Lightweight Directory Access Protocol, Service Location Protocol, etc.) without translation at a gateway.

A further complexity occurs when a device supports both wireless and wireline operation. If the device uses IP for wireless operation, uninterrupted operation when the device is connected to a wireline network is possible (using Mobile IP). If a non-IP alternative is used, this switchover is more difficult to accomplish.

Non-IP alternatives face the burden of proof that IP is so ill-suited to a wireless environment that it is not a viable technology.

2.3 IP-based Considerations

Given its worldwide deployment, IP is an obvious choice for the underlying network technology. Optimizations implemented at this level benefit traditional Internet application protocols as well as new ones layered on top of IP or UDP.

2.3.1 Choosing the MTU [Stevens94, RFC1144]

In slow networks, the time required to transmit the largest possible packet may be considerable. Interactive response time should not exceed the well-known human factors limit of 100 to 200 ms. This should be considered the maximum time budget to (1) send a packet and (2) obtain a response. In most networking stack implementations, (1) is highly dependent on the maximum transmission unit (MTU). In the worst case, a small packet from an interactive application may have to wait for a large packet from a bulk transfer application before being sent. Hence, a good rule of thumb is to choose an MTU such that its transmission time is less than (or not much larger than) 200 ms.

Of course, compression and type-of-service queuing (whereby interactive data packets are given a higher priority) may alleviate this problem. In particular, the latter may reduce the average wait time to about half the MTU's transmission time.

2.3.2 Path MTU Discovery [RFC1191]

Path MTU discovery benefits any protocol built on top of IP. It allows a sender to determine what the maximum end-to-end transmission unit is to a given destination. Without Path MTU discovery, the default IPv4 MTU size is 576. The benefits of using a larger MTU are:

- Smaller ratio of header overhead to data
- Allows TCP to grow its congestion window faster, since it increases in units of segments.

Of course, for a given BER, a larger MTU has a correspondingly larger probability of error within any given segment. The BER may be reduced using lower level techniques like FEC and link-layer retransmissions. The issue is that now delays may become a problem due to the additional retransmissions, and the fact that packet transmission time increases with a larger MTU.

Recommendation: Path MTU discovery is recommended. [AGS98] already recommends its use in satellite environments.

2.3.3 Non-TCP Proposals

Other proposals assume an underlying IP datagram service, and implement an optimized transport either directly on top of IP [NETBLT] or on top of UDP [MNCP]. Not relying on TCP is a bold move, given the wealth of experience and research related to it. It could be argued that the Internet has not collapsed because its main protocol, TCP, is very careful in how it uses the network, and generally treats it as a black box assuming all packet losses are due to congestion and prudently backing off. This avoids further congestion.

However, in the wireless medium, packet losses may also be due to corruption due to high BER, fading, and so on. Here, the right approach is to try harder, instead of backing off. Alternative transport protocols are:

- NETBLT [NETBLT, RFC1986, RFC1030]
- MNCP [MNCP]

- ESRO [RFC2188]

- RDP [RFC908, RFC1151]

- VMTP [VMTP]

3 The Case for TCP

This is one of the most hotly debated issues in the wireless arena. Here are some arguments against it:

- It is generally recognized that TCP does not perform well in the presence of significant levels of non-congestion loss. TCP detractors argue that the wireless medium is one such case, and that it is hard enough to fix TCP. They argue that it is easier to start from scratch.

- TCP has too much header overhead.

- By the time the mechanisms are in place to fix it, TCP is very heavy, and ill-suited for use by lightweight, portable devices.

and here are some in support of TCP:

- It is preferable to continue using the same protocol that the rest of the Internet uses for compatibility reasons. Any extensions specific to the wireless link may be negotiated.

- Legacy mechanisms may be reused (for example three-way handshake).

- Link-layer FEC and ARQ can reduce the BER such that any losses TCP does see are, in fact, caused by congestion (or a sustained interruption of link connectivity). Modern W-WAN technologies do this (CDPD, US-TDMA, CDMA, GSM), thus improving TCP throughput.

- Handoffs among different technologies are made possible by Mobile IP [RFC2002], but only if the same protocols, namely TCP/IP, are used throughout.

- Given TCP’s wealth of research and experience, alternative protocols are relatively immature, and the full implications of their widespread deployment not clearly understood.

Overall, we feel that the performance of TCP over long-thin networks can be improved significantly. Mechanisms to do so are discussed in the next sections.

4 Candidate Optimizations

There is a large volume of work on the subject of optimizing TCP for operation over wireless media. Even though satellite networks generally fall in the LFN regime, our current LTN focus has much to benefit from it. For example, the work of the TCP-over-Satellite working group of the IETF has been extremely helpful in preparing this section [AGS98, ADGGHOSSTT98].

4.1 TCP: Current Mechanisms

A TCP sender adapts its use of bandwidth based on feedback from the receiver. The high latency characteristic of LTNs implies that TCP's adaptation is correspondingly slower than on networks with shorter delays. Similarly, delayed ACKs exacerbate the perceived latency on the link. Given that TCP grows its congestion window in units of segments, small MTUs may slow adaptation even further.

4.1.1 Slow Start and Congestion Avoidance

Slow Start and Congestion Avoidance [RFC2581] are essential the Internet's stability. However there are two reasons why the wireless medium adversely affects them:

- Whenever TCP's retransmission timer expires, the sender assumes that the network is congested and invokes slow start. This is why it is important to minimize the losses caused by corruption, leaving only those caused by congestion (as expected by TCP).

- The sender increases its window based on the number of ACKs received. Their rate of arrival, of course, is dependent on the RTT (round-trip-time) between sender and receiver, which implies long ramp-up times in high latency links like LTNs. The dependency lasts until the pipe is filled.

- During slow start, the sender increases its window in units of segments. This is why it is important to use an appropriately large MTU which, in turn, requires requires link layers with low loss.

4.1.2 Fast Retransmit and Fast Recovery

When a TCP sender receives several duplicate ACKs, fast retransmit [RFC2581] allows it to infer that a segment was lost. The sender retransmits what it considers to be this lost segment without waiting for the full timeout, thus saving time.

After a fast retransmit, a sender invokes the fast recovery [RFC2581] algorithm. Fast recovery allows the sender to transmit at half its previous rate (regulating the growth of its window based on congestion avoidance), rather than having to begin a slow start. This also saves time.

In general, TCP can increase its window beyond the delay-bandwidth product. However, in LTN links the congestion window may remain rather small, less than four segments, for long periods of time due to any of the following reasons:

1. Typical "file size" to be transferred over a connection is relatively small (Web requests, Web document objects, email messages, files, etc.) In particular, users of LTNs are not very willing to carry out large transfers as the response time is so long.

2. If the link has high BER, the congestion window tends to stay small

3. When an LTN is combined with a highly congested wireline Internet path, congestion losses on the Internet have the same effect as 2.

4. Commonly, ISPs/operators configure only a small number of buffers (even as few as for 3 packets) per user in their dial-up routers

5. Often small socket buffers are recommended with LTNs in order to prevent the RTO from inflating and to diminish the amount of packets with competing traffic.

A small window effectively prevents the sender from taking advantage of Fast Retransmits. Moreover, efficient recovery from multiple losses within a single window requires adoption of new proposals (NewReno [RFC2582]). In addition, on slow paths with no packet reordering waiting for three duplicate ACKs to arrive postpones retransmission unnecessarily.

Recommendation: Implement Fast Retransmit and Fast Recovery at this time. This is a widely-implemented optimization and is currently at Proposed Standard level. [AGS98] recommends implementation of Fast Retransmit/Fast Recovery in satellite environments. NewReno [RFC2582] apparently does help a sender better handle partial ACKs and multiple losses in a single window, but at this point is not recommended due to its experimental nature. Instead, SACK [RFC2018] is the preferred mechanism.

4.2 Connection Setup with T/TCP [RFC1397, RFC1644]

TCP engages in a "three-way handshake" whenever a new connection is set up. Data transfer is only possible after this phase has completed successfully. T/TCP allows data to be exchanged in parallel with the connection set up, saving valuable time for short transactions on long-latency networks.

Recommendation: T/TCP is not recommended, for these reasons:

- It is an Experimental RFC.
- It is not widely deployed, and it has to be deployed at both ends of a connection.
- Security concerns have been raised that T/TCP is more vulnerable to address-spoofing attacks than TCP itself.
- At least some of the benefits of T/TCP (eliminating three-way handshake on subsequent query-response transactions, for instance) are also available with persistent connections on HTTP/1.1, which is more widely deployed.

[ADGGHOSSTT98] does not have a recommendation on T/TCP in satellite environments.

4.3 Slow Start Proposals

Because slow start dominates the network response seen by interactive users at the beginning of a TCP connection, a number of proposals have been made to modify or eliminate slow start in long latency environments.

Stability of the Internet is paramount, so these proposals must demonstrate that they will not adversely affect Internet congestion levels in significant ways.

4.3.1 Larger Initial Window

Traditional slow start, with an initial window of one segment, is a time-consuming bandwidth adaptation procedure over LTNs. Studies on an initial window larger than one segment [RFC2414, AHO98] resulted in the TCP standard supporting a maximum value of 2 [RFC2581]. Higher values are still experimental in nature.

In simulations with an increased initial window of three packets [RFC2415], this proposal does not contribute significantly to packet drop rates, and it has the added benefit of improving initial response times when the peer device delays acknowledgements during slow start (see next proposal).

[RFC2416] addresses situations where the initial window exceeds the number of buffers available to TCP and indicates that this situation is no different from the case where the congestion window grows beyond the number of buffers available.

[RFC2581] now allows an initial congestion window of two segments. A larger initial window, perhaps as many as four segments, might be allowed in the future in environments where this significantly improves performance (LFNs and LTNs).

Recommendation: Implement this on devices now. The research on this optimization indicates that 3 segments is a safe initial setting, and is centering on choosing between 2, 3, and 4. For now, use 2 (following RFC2581), which at least allows clients running query-response applications to get an initial ACK from unmodified servers without waiting for a typical delayed ACK timeout of 200 milliseconds, and saves two round-trips. An initial window of 3 [RFC2415] looks promising and may be adopted in the future pending further research and experience.

4.3.2 Growing the Window during Slow Start

The sender increases its window based on the flow of ACKs coming back from the receiver. Particularly during slow start, this flow is very important. A couple of the proposals that have been studied are (1) ACK counting and (2) ACK-every-segment.

4.3.2.1 ACK Counting

The main idea behind ACK counting is:

- Make each ACK count to its fullest by growing the window based on the data being acknowledged (byte counting) instead of the number of ACKs (ACK counting). This has been shown to cause bursts which lead to congestion. [Allman98] shows that Limited Byte Counting (LBC), in which the window growth is limited to 2 segments, does not lead to as much burstiness, and offers some performance gains.

Recommendation: Unlimited byte counting is not recommended. Van Jacobson cautions against byte counting [TCPSATMIN] because it leads to burstiness, and recommends ACK spacing [ACKSPACING] instead.

ACK spacing requires ACKs to consistently pass through a single ACK-spacing router. This requirement works well for W-WAN environments if the ACK-spacing router is also the intermediate node.

Limited byte counting warrants further investigation before we can recommend this proposal, but it shows promise.

4.3.2.2 ACK-every-segment

The main idea behind ACK-every-segment is:

- Keep a constant stream of ACKs coming back by turning off delayed ACKs [RFC1122] during slow start. ACK-every-segment must be limited to slow start, in order to avoid penalizing asymmetric-bandwidth configurations. For instance, a low bandwidth link carrying acknowledgements back to the sender, hinders the growth of the congestion window, even if the link toward the client has a greater bandwidth [BPK99].

Even though simulations confirm its promise (it allows receivers to receive the second segment from unmodified senders without waiting for a typical delayed ACK timeout of 200 milliseconds), for this technique to be practical the receiver must acknowledge every segment only when the sender is in slow start. Continuing to do so when the sender is in congestion avoidance may have adverse effects on the mobile device’s battery consumption and on traffic in the network.

This violates a SHOULD in [RFC2581]: delayed acknowledgements SHOULD be used by a TCP receiver.

"Disabling Delayed ACKs During Slow Start" is technically unimplementable, as the receiver has no way of knowing when the sender crosses ssthresh (the "slow start threshold") and begins using the congestion avoidance algorithm. If receivers follow recommendations for increased initial windows, disabling delayed ACKs during an increased initial window would open the TCP window more rapidly without doubling ACK traffic in general. However, this scheme might double ACK traffic if most connections remain in slow-start.

Recommendation: ACK only the first segment on a new connection with no delay.

4.3.3 Terminating Slow Start

New mechanisms [ADGGHOSSTT98] are being proposed to improve TCP's adaptive properties such that the available bandwidth is better utilized while reducing the possibility of congesting the network. This results in the closing of the congestion window to 1 segment (which precludes fast retransmit), and the subsequent slow start phase.

Theoretically, an optimum value for slow-start threshold (ssthresh) allows connection bandwidth utilization to ramp up as aggressively as possible without "overshoot" (using so much bandwidth that packets are lost and congestion avoidance procedures are invoked).

Recommendation: Estimating the slow start threshold is not recommended. Although this would be helpful if we knew how to do it, rough consensus on the tcp-impl and tcp-sat mailing lists is that in non-trivial operational networks there is no reliable method to probe during TCP startup and estimate the bandwidth available.

4.3.4 Generating ACKs during Slow Start

Mitigations that inject additional ACKs (whether "ACK-first-segment" or "ACK-every-segment-during-slow-start") beyond what today's conformant TCPs inject are only applicable during the slow-start phases of a connection. After an initial exchange, the connection usually completes slow-start, so TCPs only inject additional ACKs when (1) the connection is closed, and a new connection is opened, or (2) the TCPs handle idle connection restart correctly by performing slow start.

Item (1) is typical when using HTTP/1.0, in which each request-response transaction requires a new connection. Persistent connections in HTTP/1.1 help in maintaining a connection in congestion avoidance instead of constantly reverting to slow-start. Because of this, these optimizations which are only enabled during slow-start do not get as much of a chance to act. Item (2), of course, is independent of HTTP version.

4.4 ACK Spacing

During slow start, the sender responds to the incoming ACK stream by transmitting N+1 segments for each ACK, where N is the number of new segments acknowledged by the incoming ACK. This results in data being sent at twice the speed at which it can be processed by the network. Accordingly, queues will form, and due to insufficient buffering at the bottleneck router, packets may get dropped before the link's capacity is full.

Spacing out the ACKs effectively controls the rate at which the sender will transmit into the network, and may result in little or no queueing at the bottleneck router [ACKSPACING]. Furthermore, ack spacing reduces the size of the bursts.

Recommendation: No recommendation at this time. Continue monitoring research in this area.

4.5 Delayed Duplicate Acknowlegements

As was mentioned above, link-layer retransmissions may decrease the BER enough that congestion accounts for most of packet losses; still, nothing can be done about interruptions due to handoffs, moving beyond wireless coverage, etc. In this scenario, it is imperative to prevent interaction between link-layer retransmission and TCP retransmission as these layers duplicate each other's efforts. In such an environment it may make sense to delay TCP's efforts so as to give the link-layer a chance to recover. With this in mind, the Delayed Dupacks [MV97, Vaidya99] scheme selectively delays duplicate acknowledgements at the receiver. It is preferable to allow a local mechanism to resolve a local problem, instead of invoking TCP's end-to-end mechanism and incurring the associated costs, both in terms of wasted bandwidth and in terms of its effect on TCP's window behavior.

The Delayed Dupacks scheme can be used despite IP encryption since the intermediate node does not need to examine the TCP headers.

Currently, it is not well understood how long the receiver should delay the duplicate acknowledgments. In particular, the impact of wireless medium access control (MAC) protocol on the choice of delay parameter needs to be studied. The MAC protocol may affect the ability to choose the appropriate delay (either statically or dynamically). In general, significant variabilities in link-level retransmission times can have an adverse impact on the performance of the Delayed Dupacks scheme. Furthermore, as discussed later in section 4.10.3, Delayed Dupacks and some other schemes (such as Snoop [SNOOP]) are only beneficial in certain types of network links.

Recommendation: Delaying duplicate acknowledgements may be useful in specific network topologies, but a general recommendation requires further research and experience.

4.6 Selective Acknowledgements [RFC2018]

SACK may not be useful in many LTNs, according to Section 1.1 of [TCPHP]. In particular, SACK is more useful in the LFN regime, especially if large windows are being used, because there is a

considerable probability of multiple segment losses per window. In the LTN regime, TCP windows are much smaller, and burst errors must be much longer in duration in order to damage multiple segments.

Accordingly, the complexity of SACK may not be justifiable, unless there is a high probability of burst errors and congestion on the wireless link. A desire for compatibility with TCP recommendations for non-LTN environments may dictate LTN support for SACK anyway.

[AGS98] recommends use of SACK with Large TCP Windows in satellite environments, and notes that this implies support for PAWS (Protection Against Wrapped Sequence space) and RTTM (Round Trip Time Measurement) as well.

Berkeley's SNOOP protocol research [SNOOP] indicates that SACK does improve throughput for SNOOP when multiple segments are lost per window [BPSK96]. SACK allows SNOOP to recover from multi-segment losses in one round-trip. In this case, the mobile device needs to implement some form of selective acknowledgements. If SACK is not used, TCP may enter congestion avoidance as the time needed to retransmit the lost segments may be greater than the retransmission timer.

Recommendation: Implement SACK now for compatibility with other TCPs and improved performance with SNOOP.

4.7 Detecting Corruption Loss

4.7.1 Without Explicit Notification

In the absence of explicit notification from the network, some researchers have suggested statistical methods for congestion avoidance [Jain89, WC91, VEGAS]. A natural extension of these heuristics would enable a sender to distinguish between losses caused by congestion and other causes. The research results on the reliability of sender-based heuristics is unfavorable [BV97, BV98]. [BV98a] reports better results in constrained environments using packet inter-arrival times measured at the receiver, but highly-variable delay - of the type encountered in wireless environments during intercell handoff - confounds these heuristics.

Recommendation: No recommendation at this time - continue to monitor research results.

4.7.2 With Explicit Notifications

With explicit notification from the network it is possible to determine when a loss is due to congestion. Several proposals along these lines include:

- Explicit Loss Notification (ELN) [BPSK96]
- Explicit Bad State Notification (EBSN) [BBKVP96]
- Explicit Loss Notification to the Receiver (ELNR), and Explicit Delayed Dupack Activation Notification (EDDAN) (notifications to mobile receiver) [MV97]
- Explicit Congestion Notification (ECN) [ECN]

Of these proposals, Explicit Congestion Notification (ECN) seems closest to deployment on the Internet, and will provide some benefit for TCP connections on long thin networks (as well as for all other TCP connections).

Recommendation: No recommendation at this time. Schemes like ELNR and EDDAN [MV97], in which the only systems that need to be modified are the intermediate node and the mobile device, are slated for adoption pending further research. However, this solution has some limitations. Since the intermediate node must have access to the TCP headers, the IP payload must not be encrypted.

ECN uses the TOS byte in the IP header to carry congestion information (ECN-capable and Congestion-encountered). This byte is not encrypted in IPSEC, so ECN can be used on TCP connections that are encrypted using IPSEC.

Recommendation: Implement ECN. In spite of this, mechanisms for explicit corruption notification are still relevant and should be tracked.

Note: ECN provides useful information to avoid deteriorating further a bad situation, but has some limitations for wireless applications. Absence of packets marked with ECN should not be interpreted by ECN-capable TCP connections as a green light for aggressive retransmissions. On the contrary, during periods of extreme network congestion routers may drop packets marked with explicit notification because their buffers are exhausted - exactly the wrong time for a host to begin retransmitting aggressively.

4.8 Active Queue Management

As has been pointed out above, TCP responds to congestion by closing down the window and invoking slow start. Long-delay networks take a particularly long time to recover from this condition. Accordingly, it is imperative to avoid congestion in LTNs. To remedy this, active queue management techniques have been proposed as enhancements to routers throughout the Internet [RED]. The primary motivation for deployment of these mechanisms is to prevent "congestion collapse" (a severe degradation in service) by controlling the average queue size at the routers. As the average queue length grows, Random Early Detection [RED] increases the possibility of dropping packets.

The benefits are:

- Reduce packet drops in routers. By dropping a few packets before severe congestion sets in, RED avoids dropping bursts of packets. In other words, the objective is to drop m packets early to prevent n drops later on, where m is less than n.

- Provide lower delays. This follows from the smaller queue sizes, and is particularly important for interactive applications, for which the inherent delays of wireless links already push the user experience to the limits of the non-acceptable.

- Avoid lock-outs. Lack of resources in a router (and the resultant packet drops) may, in effect, obliterate throughput on certain connections. Because of active queue management, it is more probable for an incoming packet to find available buffer space at the router.

Active Queue Management has two components: (1) routers detect congestion before exhausting their resources, and (2) they provide some form of congestion indication. Dropping packets via RED is only one example of the latter. Another way to indicate congestion is to use ECN [ECN] as discussed above under "Detecting Corruption Loss: With Explicit Notifications."

Recommendation: RED is currently being deployed in the Internet, and LTNs should follow suit. ECN deployment should complement RED’s.

4.9 Scheduling Algorithms

Active queue management helps control the length of the queues. Additionally, a general solution requires replacing FIFO with other scheduling algorithms that improve:

1. Fairness (by policing how different packet streams utilize the available bandwidth), and

2. Throughput (by improving the transmitter's radio channel utilization).

For example, fairness is necessary for interactive applications (like telnet or web browsing) to coexist with bulk transfer sessions. Proposals here include:

- Fair Queueing (FQ) [Demers90]

- Class-based Queueing (CBQ) [Floyd95]

Even if they are only implemented over the wireless link portion of the communication path, these proposals are attractive in wireless LTN environments, because new connections for interactive applications can have difficulty starting when a bulk TCP transfer has already stabilized using all available bandwidth.

In our base architecture described above, the mobile device typically communicates directly with only one wireless peer at a given time: the intermediate node. In some W-WANs, it is possible to directly address other mobiles within the same cell. Direct communication with each such wireless peer may traverse a spatially distinct path, each of which may exhibit statistically independent radio link characteristics. Channel State Dependent Packet Scheduling (CSDP) [BBKT96] tracks the state of the various radio links (as defined by the target devices), and gives preferential treatment to packets destined for radio links in a "good" state. This avoids attempting to transmit to (and expect acknowledgements from) a peer on a "bad" radio link, thus improving throughput.

A further refinement of this idea suggests that both fairness and throughput can be improved by combining a wireless-enhanced CBQ with CSDP [FSS98].

Recommendation: No recommendation at this time, pending further study.

4.10 Split TCP and Performance-Enhancing Proxies (PEPs)

Given the dramatic differences between the wired and the wireless links, a very common approach is to provide some impedance matching where the two different technologies meet: at the intermediate node.

The idea is to replace an end-to-end TCP connection with two clearly distinct connections: one across the wireless link, the other across its wireline counterpart. Each of the two resulting TCP sessions operates under very different networking characteristics, and may adopt the policies best suited to its particular medium. For example, in a specific LTN topology it may be desirable to modify TCP Fast Retransmit to resend after the first duplicate ack and Fast Recovery to not shrink the congestion window if the LTN link has an extremely long RTT, is known to not reorder packets, and is not subject to congestion. Moreover, on a long-delay link or on a link with a relatively high bandwidth-delay product it may be desirable to "slow-start" with a relatively large initial window, even larger than four segments. While these kinds of TCP modifications can be negotiated to be employed over the LTN link, they would not be deployed end-to-end over the global Internet. In LTN topologies where the underlying link characteristics are known, a various similar types of performance enhancements can be employed without endangering operations over the global Internet.

In some proposals, in addition to a PEP mechanism at the intermediate node, custom protocols are used on the wireless link (for example, [WAP], [YB94] or [MOWGLI]).

Even if the gains from using non-TCP protocols are moderate or better, the wealth of research on optimizing TCP for wireless, and compatibility with the Internet are compelling reasons to adopt TCP on the wireless link (enhanced as suggested in section 5 below).

4.10.1 Split TCP Approaches

Split-TCP proposals include schemes like I-TCP [ITCP] and MTCP [YB94] which achieve performance improvements by abandoning end-to-end semantics.

The Mowgli architecture [MOWGLI] proposes a split approach with support for various enhancements at all the protocol layers, not only at the transport layer. Mowgli provides an option to replace the TCP/IP core protocols on the LTN link with a custom protocol that is tuned for LTN links [KRLKA97]. In addition, the protocol provides various features that are useful with LTNs. For example, it provides priority-based multiplexing of concurrent connections together with shared flow control, thus offering link capacity to interactive applications in a timely manner even if there are bandwidth-intensive background transfers. Also with this option, Mowgli preserves the socket semantics on the mobile device so that legacy applications can be run unmodified.

Employing split TCP approaches have several benefits as well as drawbacks. Benefits related to split TCP approaches include the following:

- Splitting the end-to-end TCP connection into two parts is a straightforward way to shield the problems of the wireless link from the wireline Internet path, and vice versa. Thus, a split TCP approach enables applying local solutions to the local problems on the wireless link. For example, it automatically solves the problem of distinguishing congestion related packet losses on the wireline Internet and packet losses due to transmission error on the wireless link as these occur on separate TCP connections. Even if both segments experience congestion, it may be of a different nature and may be treated as such. Moreover, temporary disconnections of the wireless link can be effectively shielded from the wireline Internet.

- When one of the TCP connections crosses only a single hop wireless link or a very limited number of hops, some or all link characteristics for the wireless TCP path are known. For example, with a particular link we may know that the link provides reliable delivery of packets, packets are not delivered out of order, or the link is not subject to congestion. Having this information for the TCP path one could expect that defining the TCP mitigations to be employed becomes a significantly easier task. In addition, several mitigations that cannot be employed safely over the global Internet, can be successfully employed over the wireless link.

- Splitting one TCP connection into two separate ones allows much earlier deployment of various recent proposals to improve TCP performance over wireless links; only the TCP implementations of the mobile device and intermediate node need to be modified, thus allowing the vast number of Internet hosts to continue running the legacy TCP implementations unmodified. Any mitigations that would require modification of TCP in these wireline hosts may take far too long to become widely deployed.

- Allows exploitation of various application level enhancements which may give significant performance gains (see section 4.10.2).

Drawbacks related to split TCP approaches include the following:

- One of the main criticisms against the split TCP approaches is that it breaks TCP end-to-end semantics. This has various drawbacks some of which are more severe than others. The most detrimental drawback is probably that splitting the TCP connection disables end-to-end usage of IP layer security mechanisms, precluding the application of IPSec to achieve end-to-end

security. Still, IPSec could be employed separately in each of the two parts, thus requiring the intermediate node to become a party to the security association between the mobile device and the remote host. This, however, is an undesirable or unacceptable alternative in most cases. Other security mechanisms above the transport layer, like TLS [RFC2246] or SOCKS [RFC1928], should be employed for end-to-end security.

- Another drawback of breaking end-to-end semantics is that crashes of the intermediate node become unrecoverable resulting in termination of the TCP connections. Whether this should be considered a severe problem depends on the expected frequency of such crashes.

- In many occasions claims have been stated that if TCP end-to-end semantics is broken, applications relying on TCP to provide reliable data delivery become more vulnerable. This, however, is an overstatement as a well-designed application should never fully rely on TCP in achieving end-to-end reliability at the application level. First, current APIs to TCP, such as the Berkeley socket interface, do not allow applications to know when an TCP acknowledgement for previously sent user data arrives at TCP sender. Second, even if the application is informed of the TCP acknowledgements, the sending application cannot know whether the receiving application has received the data: it only knows that the data reached the TCP receive buffer at the receiving end. Finally, in order to achieve end-to-end reliability at the application level an application level acknowledgement is required to confirm that the receiver has taken the appropriate actions on the data it received.

- When a mobile device moves, it is subject to handovers by the serving base station. If the base station acts as the intermediate node for the split TCP connection, the state of both TCP endpoints on the previous intermediate node must be transferred to the new intermediate node to ensure continued operation over the split TCP connection. This requires extra work and causes overhead. However, in most of the W-WAN wireless networks, unlike in W-LANs, the W-WAN base station does not provide the mobile device with the connection point to the wireline Internet (such base stations may not even have an IP stack). Instead, the W-WAN network takes care of the mobility and retains the connection point to the wireline Internet unchanged while the mobile device moves. Thus, TCP state handover is not required in most W-WANs.

- The packets traversing through all the protocol layers up to transport layer and again down to the link layer result in extra overhead at the intermediate node. In case of LTNs with low

bandwidth, this extra overhead does not cause serious additional performance problems unlike with W-LANs that typically have much higher bandwidth.

- Split TCP proposals are not applicable to networks with asymmetric routing. Deploying a split TCP approach requires that traffic to and from the mobile device be routed through the intermediate node. With some networks, this cannot be accomplished, or it requires that the intermediate node is located several hops away from the wireless network edge which in turn is unpractical in many cases and may result in non-optimal routing.

- Split TCP, as the name implies, does not address problems related to UDP.

It should noted that using split TCP does not necessarily exclude simultaneous usage of IP for end-to-end connectivity. Correct usage of split TCP should be managed per application or per connection and should be under the end-user control so that the user can decide whether a particular TCP connection or application makes use of split TCP or whether it operates end-to-end directly over IP.

Recommendation: Split TCP proposals that alter TCP semantics are not recommended. Deploying custom protocols on the wireless link, such as MOWGLI proposes is not recommended, because this note gives preference to (1) improving TCP instead of designing a custom protocol and (2) allowing end-to-end sessions at all times.

4.10.2 Application Level Proxies

Nowadays, application level proxies are widely used in the Internet. Such proxies include Web proxy caches, relay MTAs (Mail Transfer Agents), and secure transport proxies (e.g., SOCKS). In effect, employing an application level proxy results in a "split TCP connection" with the proxy as the intermediary. Hence, some of the problems present with wireless links, such as combining of a congested wide-area Internet path with a wireless LTN link, are automatically alleviated to some extent.

The application protocols often employ plenty of (unnecessary) round trips, lots of headers and inefficient encoding. Even unnecessary data may get delivered over the wireless link in regular application protocol operation. In many cases a significant amount of this overhead can be reduced by simply running an application level proxy on the intermediate node. With LTN links, significant additional improvement can be achieved by introducing application level proxies with application-specific enhancements. Such a proxy may employ an enhanced version of the application protocol over the wireless link.

In an LTN environment enhancements at the application layer may provide much more notable performance improvements than any transport level enhancements.

The Mowgli system provides full support for adding application level agent-proxy pairs between the client and the server, the agent on the mobile device and the proxy on the intermediate node. Such a pair may be either explicit or fully transparent to the applications, but it is, at all times, under the end-user control. Good examples of enhancements achieved with application-specific proxies include Mowgli WWW [LAKLR95], [LHKR96] and WebExpress [HL96], [CTCSM97].

Recommendation: Usage of application level proxies is conditionally recommended: an application must be proxy enabled and the decision of employing a proxy for an application must be under the user control at all times.

4.10.3 Snoop and its Derivatives

Berkeley’s SNOOP protocol [SNOOP] is a hybrid scheme mixing link-layer reliability mechanisms with the split connection approach. It is an improvement over split TCP approaches in that end-to-end semantics are retained. SNOOP does two things:

1. Locally (on the wireless link) retransmit lost packets, instead of allowing TCP to do so end-to-end.

2. Suppress the duplicate acks on their way from the receiver back to the sender, thus avoiding fast retransmit and congestion avoidance at the latter.

Thus, the Snoop protocol is designed to avoid unnecessary fast retransmits by the TCP sender, when the wireless link layer retransmits a packet locally. Consider a system that does not use the Snoop agent. Consider a TCP sender S that sends packets to receiver R via an intermediate node IN. Assume that the sender sends packet A, B, C, D, E (in that order) which are forwarded by IN to the wireless receiver R. Assume that the intermediate node then retransmits B subsequently, because the first transmission of packet B is lost due to errors on the wireless link. In this case, receiver R receives packets A, C, D, E and B (in that order). Receipt of packets C, D and E triggers duplicate acknowledgements. When the TCP sender receives three duplicate acknowledgements, it triggers fast retransmit (which results in a retransmission, as well as reduction of congestion window). The fast retransmit occurs despite the link level retransmit on the wireless link, degrading throughput.

SNOOP [SNOOP] deals with this problem by dropping TCP dupacks appropriately (at the intermediate node). The Delayed Dupacks (see section 4.5) attempts to approximate Snoop without requiring modifications at the intermediate node. Such schemes are needed only if the possibility of a fast retransmit due to wireless errors is non-negligible. In particular, if the wireless link uses a stop-and-go protocol (or otherwise delivers packets in-order), then these schemes are not very beneficial. Also, if the bandwidth-delay product of the wireless link is smaller than four segments, the probability that the intermediate node will have an opportunity to send three new packets before a lost packet is retransmitted is small. Since at least three dupacks are needed to trigger a fast retransmit, with a wireless bandwidth-delay product less than four packets, schemes such as Snoop and Delayed Dupacks would not be necessary (unless the link layer is not designed properly). Conversely, when the wireless bandwidth-delay product is large enough, Snoop can provide significant performance improvement (compared with standard TCP). For further discussion on these topics, please refer to [Vaidya99].

The Delayed Dupacks scheme tends to provide performance benefit in environments where Snoop performs well. In general, performance improvement achieved by the Delayed Dupacks scheme is a function of packet loss rates due to congestion and transmission errors. When congestion-related losses occur, the Delayed Dupacks scheme unnecessarily delays retransmission. Thus, in the presence of congestion losses, the Delayed Dupacks scheme cannot achieve the same performance improvement as Snoop. However, simulation results [Vaidya99] indicate that the Delayed Dupacks can achieve a significant improvement in performance despite moderate congestion losses.

WTCP [WTCP] is similar to SNOOP in that it preserves end-to-end semantics. In WTCP, the intermediate node uses a complex scheme to hide the time it spends recovering from local errors across the wireless link (this typically includes retransmissions due to error recovery, but may also include time spent dealing with congestion). The idea is for the sender to derive a smooth estimate of round-trip time. In order to work effectively, it assumes that the TCP endpoints implement the Timestamps option in RFC 1323 [TCPHP]. Unfortunately, support for RFC 1323 in TCP implementations is not yet widespread. Beyond this, WTCP requires changes only at the intermediate node.

SNOOP and WTCP require the intermediate node to examine and operate on the traffic between the portable wireless device and the TCP server on the wired Internet. SNOOP and WTCP do not work if the IP traffic is encrypted, unless, of course, the intermediate node shares

the security association between the mobile device and its end-to-end peer. They also require that both the data and the corresponding ACKs traverse the same intermediate node. Furthermore, if the intermediate node retransmits packets at the transport layer across the wireless link, this may duplicate efforts by the link-layer. SNOOP has been described by its designers as a TCP-aware link-layer. This is the right approach: the link and network layers can be much more aware of each other than traditional OSI layering suggests.

Encryption of IP packets via IPSEC's ESP header (in either transport or tunnel mode) renders the TCP header and payload unintelligible to the intermediate node. This precludes SNOOP (and WTCP) from working, because it needs to examine the TCP headers in both directions. Possible solutions involve:

- making the SNOOP (or WTCP) intermediate node a party to the security association between the client and the server

- IPSEC tunneling mode, terminated at the SNOOPing intermediate node

However, these techniques require that users trust intermediate nodes. Users valuing both privacy and performance should use SSL or SOCKS for end-to-end security. These, however, are implemented above the transport layer, and are not as resistant to some security attacks (for example, those based on guessing TCP sequence numbers) as IPSEC.

Recommendation: Implement SNOOP on intermediate nodes now. Research results are encouraging, and it is an "invisible" optimization in that neither the client nor the server needs to change, only the intermediate node (for basic SNOOP without SACK). However, as discussed above there is little or no benefit from implementing SNOOP if:

1. The wireless link provides reliable, in-order packet delivery, or,

2. The bandwidth-delay product of the wireless link is smaller than four segments.

4.10.4 PEPs to handle Periods of Disconnection

Periods of disconnection are very common in wireless networks, either during handoff, due to lack of resources (dropped connections) or natural obstacles. During these periods, a TCP sender does not receive the expected acknowledgements. Upon expiration of the retransmit timer, this causes TCP to close its congestion window with all the related drawbacks. Re-transmitting packets is useless

since the connection is broken. [M-TCP] aims at enabling TCP to better handle handoffs and periods of disconnection, while preserving end-to-end semantics. M-TCP adds an element: supervisor host (SH-TCP) at the edge of the wireless network.

This intermediate node monitors the traffic coming from the sender to the mobile device. It does not break end-to-end semantics because the ACKs sent from the intermediate node to the sender are effectively the ones sent by the mobile node. The principle is to generally leave the last byte unacknowledged. Hence, SH-TCP could shut down the sender's window by sending the ACK for the last byte with a window set to zero. Thus the sender will go to persist mode.

The second optimization is done on both the intermediate node and the mobile host. On the latter, TCP is aware of the current state of the connection. In the event of a disconnection, it is capable of freezing all timers. Upon reconnection, the mobile sends a specially marked ACK with the number of the highest byte received. The intermediate node assumes that the mobile is disconnected because it monitors the flow on the wireless link, so in the absence of acknowledgments from the mobile, it will inform SH-TCP, which will send the ACK closing the sender window as described in the previous paragraph. The intermediate node learns that the mobile is again connected when it receives a duplicate acknowledgment marked as reconnected. At this point it sends a duplicate ACK to the sender and grows the window. The sender exits persist mode and resumes transmitting at the same rate as before. It begins by retransmitting any data previously unacknowledged by the mobile node. Non overlapping or non soft handoffs are lightweight because the previous intermediate system can shrink the window, and the new one modifies it as soon as it has received an indication from the mobile.

Recommendation: M-TCP is not slated for adoption at this moment, because of the highly experimental nature of the proposal, and the uncertainty that TCP/IP implementations handle zero window updates correctly. Continue tracking developments in this space.

4.11 Header Compression Alternatives

Because Long Thin Networks are bandwidth-constrained, compressing every byte out of over-the-air segments is worth while.

Mechanisms for TCP and IP header compression defined in [RFC1144, IPHC, IPHC-RTP, IPHC-PPP] provide the following benefits:

- Improve interactive response time

- Allow using small packets for bulk data with good line efficiency

- Allow using small packets for delay sensitive low data-rate traffic

- Decrease header overhead (for a common TCP segment size of 512 the header overhead of IPv4/TCP within a Mobile IP tunnel can decrease from 11.7 to less than 1 per cent.

- Reduce packet loss rate over lossy links (because of the smaller cross-section of compressed packets).

Van Jacobson (VJ) header compression [RFC1144] describes a Proposed Standard for TCP Header compression that is widely deployed. It uses TCP timeouts to detect a loss of synchronization between the compressor and decompressor. [IPHC] includes an explicit request for transmission of uncompressed headers to allow resynchronization without waiting for a TCP timeout (and executing congestion avoidance procedures).

Recommendation: Implement [IPHC], in particular as it relates to IP-in-IP [RFC2003] and Minimal Encapsulation [RFC2004] for Mobile IP, as well as TCP header compression for lossy links and links that reorder packets. PPP capable devices should implement [IPHC-PPP]. VJ header compression may optionally be implemented as it is a widely deployed Proposed Standard. However, it should only be enabled when operating over reliable LTNs, because even a single bit error most probably would result in a full TCP window being dropped, followed by a costly recovery via slow-start.

4.12 Payload Compression

Compression of IP payloads is also desirable. "IP Payload Compression Protocol (IPComp)" [IPPCP] defines a framework where common compression algorithms can be applied to arbitrary IP segment payloads. IP payload compression is something of a niche optimization. It is necessary because IP-level security converts IP payloads to random bitstreams, defeating commonly-deployed link-layer compression mechanisms which are faced with payloads that have no redundant "information" that can be more compactly represented.

However, many IP payloads are already compressed (images, audio, video, "zipped" files being FTPed), or are already encrypted above the IP layer (SSL/TLS, etc.). These payloads will not "compress" further, limiting the benefit of this optimization.

HTTP/1.1 already supports compression of the message body. For example, to use zlib compression the relevant directives are: "Content-Encoding: deflate" and "Accept-Encoding: deflate" [HTTP-PERF].

HTTP-NG is considering supporting compression of resources at the HTTP level, which would provide equivalent benefits for common compressible MIME types like text/html. This will reduce the need for IPComp. If IPComp is deployed more rapidly than HTTP-NG, IPComp compression of HTML and MIME headers would be beneficial.

In general, application-level compression can often outperform IPComp, because of the opportunity to use compression dictionaries based on knowledge of the specific data being compressed.

Recommendation: IPComp may optionally be implemented. Track HTTP-NG standardization and deployment for now. Implementing HTTP/1.1 compression using zlib SHOULD is recommended.

4.13 TCP Control Block Interdependence [Touch97]

TCP maintains per-connection information such as connection state, current round-trip time, congestion control or maximum segment size. Sharing information between two consecutive connections or when creating a new connection while the first is still active to the same host may improve performance of the latter connection. The principle could easily be extended to sharing information amongst systems in a LAN not just within a given system. [Touch97] describes cache update for both cases.

Users of W-WAN devices frequently request connections to the same servers or set of servers. For example, in order to read their email or to initiate connections to other servers, the devices may be configured to always use the same email server or WWW proxy. The main advantage of this proposal is that it relieves the application of the burden of optimizing the transport layer. In order to improve the performance of TCP connections, this mechanism only requires changes at the wireless device.

In general, this scheme should improve the dynamism of connection setup without increasing the cost of the implementation.

Recommendation: This mechanism is recommended, although HTTP/1.1 with its persistent connections may partially achieve the same effect without it. Other applications (even HTTP/1.0) may find it useful. Continue monitoring research on this. In particular, work on a "Congestion Manager" [CM] may generalize this concept of sharing information among protocols and applications with a view to making them more adaptable to network conditions.

5 Summary of Recommended Optimizations

The table below summarizes our recommendations with regards to the main proposals mentioned above.

The first column, "Stability of the Proposal," refers to the maturity of the mechanism in question. Some proposals are being pursued within the IETF in a somewhat open fashion. An IETF proposal is either an Internet Drafts (I-D) or a Request for Comments (RFC). The former is a preliminary version. There are several types of RFCs. A Draft Standards (DS) is standards track, and carries more weight than a Proposed Standard (PS), which may still undergo revisions. Informational or Experimental RFCs do not specify a standard. Other proposals are isolated efforts with little or no public review, and unknown chances of garnering industry backing.

"Implemented at" indicates which participant in a TCP session must be modified to implement the proposal. Legacy servers typically cannot be modified, so this column indicates whether implementation happens at either or both of the two nodes under some control: mobile device and intermediate node. The symbols used are: WS (wireless sender, that is, the mobile device's TCP send operation must be modified), WR (wireless receiver, that is, the mobile device's TCP receive operation must be modified), WD (wireless device, that is, modifications at the mobile device are not specific to either TCP send or receive), IN (intermediate node) and NI (network infrastructure). These entities are to be understood within the context of Section 1.1 ("Network Architecture"). NA simply means "not applicable."

The "Recommendation" column captures our suggestions. Some mechanisms are endorsed for immediate adoption, others need more evidence and research, and others are not recommended.

Name	Stability of the Proposal	Implemented at	Recommendation
Increased Initial Window	RFC 2581 (PS)	WS	Yes (initial_window=2)
Disable delayed ACKs during slow start	NA	WR	When stable
Byte counting instead of ACK counting	NA	WS	No

TCP Header compression for PPP	RFC 1144 (PS)	WD IN	Yes (see 4.11)
IP Payload Compression (IPComp)	RFC 2393 (PS)	WD (simultaneously needed on Server)	Yes
Header Compression	RFC 2507 (PS), RFC 2509 (PS)	WD IN	Yes (For IPv4, TCP and Mobile IP, PPP)
SNOOP plus SACK	In limited use	IN WD (for SACK)	Yes
Fast retransmit/fast recovery	RFC 2581 (PS)	WD	Yes (should be there already)
Transaction/TCP	RFC 1644 (Experimental)	WD (simultaneously needed on Server)	No
Estimating Slow Start Threshold (ssthresh)	NA	WS	No
Delayed Duplicate Acknowledgements	Not stable	WR IN (for notifications)	When stable
Class-based Queuing on End Systems	NA	WD	When stable
Explicit Congestion Notification	RFC 2481 (EXP)	WD NI	Yes
TCP Control Block Interdependence	RFC 2140 (Informational)	WD	Yes (Track research)

Of all the optimizations in the table above, only SNOOP plus SACK and Delayed duplicate acknowledgements are currently being proposed only for wireless networks. The others are being considered even for non-wireless applications. Their more general applicability attracts more attention and analysis from the research community.

Of the above mechanisms, only Header Compression (for IP and TCP) and "SNOOP plus SACK" cease to work in the presence of IPSec.

6 Conclusion

In view of the unpredictable and problematic nature of long thin networks, arriving at an optimized transport is a daunting task. We have reviewed the existing proposals along with future research items. Based on this overview, we also recommend mechanisms for implementation in long thin networks (LTNs).

7 Acknowledgements

The authors are deeply indebted to the IETF tcpsat and tcpimpl working groups. The following individuals have also provided valuable feedback: Mark Allman (NASA), Vern Paxson (ACIRI), Raphi Rom (Technion/Sun), Charlie Perkins (Nokia), Peter Stark (Phone.com).

8 Security Considerations

The mechanisms discussed and recommended in this document have been proposed in previous publications. The security considerations outlined in the original discussions apply here as well. Several security issues are also discussed throughout this document. Additionally, we present below a non-exhaustive list of the most salient issues concerning our recommended mechanisms:

- Larger Initial TCP Window Size

  No known security issues [RFC2414, RFC2581].

- Header Compression

  May be open to some denial of service attacks. But any attacker in a position to launch these attacks would have much stronger attacks at his disposal [IPHC, IPHC-RTP].

- Congestion Control, Fast Retransmit/Fast Recovery

  An attacker may force TCP connections to grind to a halt, or, more dangerously, behave more aggressively. The latter possibility may lead to congestion collapse, at least in some regions of the network [RFC2581].

- Explicit Congestion Notification

  It does not appear to increase the vulnerabilities in the network. On the contrary, it may reduce them by aiding in the identification of flows unresponsive to or non-compliant with TCP congestion control [ECN].

- Sharing of Network Performance Information (TCP Control Block Sharing and Congestion Manager module)

  Some information should not be shared. For example, TCP sequence numbers are used to protect against spoofing attacks. Even limiting the sharing to performance values leaves open the possibility of denial-of-service attacks [Touch97].

- Performance Enhancing Proxies

  These systems are men-in-the-middle from the point of view of their security vulnerabilities. Accordingly, they must be used with extreme care so as to prevent their being hijacked and misused.

This last point is not to be underestimated: there is a general security concern whenever an intermediate node performs operations different from those carried out in an end-to-end basis. This is not specific to performance-enhancing proxies. In particular, there may be a tendency to forego IPSEC-based privacy in order to allow, for example, a SNOOP module, header compression (TCP, UDP, RTP, etc), or HTTP proxies to work.

Adding end-to-end security at higher layers (for example via RTP encryption, or via TLS encryption of the TCP payload) alleviates the problem. However, this still leaves protocol headers in the clear, and these may be exploited for traffic analysis and denial-of-service attacks.

9 References

[ACKSPACING] Partridge, C., "ACK Spacing for High Delay-Bandwidth Paths with Insufficient Buffering", Work in Progress.

[ADGGHOSSTT98] Allman, M., Dawkins, S., Glover, D., Griner, J., Henderson, T., Heidemann, J., Kruse, H., Osterman, S., Scott, K., Semke, J., Touch, J. and D. Tran, "Ongoing TCP Research Related to Satellites", Work in Progress.

[AGS98] Allman, M., Glover, D. and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, January 1999.

[Allman98] Mark Allman. On the Generation and Use of TCP Acknowledgments. ACM Computer Communication Review, 28(5), October 1998.

[AHO98] Allman, M., Hayes, C., Ostermann, S., "An Evaluation of TCP with Larger Initial Windows," Computer Communication Review, 28(3), July 1998.

[BBKT96] Bhagwat, P., Bhattacharya, P., Krishna, A., Tripathi, S., "Enhancing Throughput over Wireless LANs Using Channel State Dependent Packet Scheduling," in Proc. IEEE INFOCOM'96, pp. 1133-40, March 1996.

[BBKVP96] Bakshi, B., P., Krishna, N., Vaidya, N., Pradhan, D.K., "Improving Performance of TCP over Wireless Networks," Technical Report 96-014, Texas A&M University, 1996.

[BPSK96] Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R., "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," in ACM SIGCOMM, Stanford, California, August 1996.

[BPK99] Balakrishnan, H., Padmanabhan, V., Katz, R., "The effects of asymmetry on TCP performance," ACM Mobile Networks and Applications (MONET), Vol. 4, No. 3, 1999, pp. 219-241.

[BV97] S. Biaz and N. H. Vaidya, "Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result," Seventh International Conference on Computer Communications and Networks (IC3N), New Orleans, October 1998.

[BV98] Biaz, S., Vaidya, N., "Sender-Based heuristics for Distinguishing Congestion Losses from Wireless Transmission Losses," Texas A&M University, Technical Report 98-013, June 1998.

[BV98a] Biaz, S., Vaidya, N., "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," Texas A&M University, Technical Report 98-014, June 1998.

[BW97] Brasche, G., Walke, B., "Concepts, Services, and Protocols of the New GSM Phase 2+ general Packet Radio Service," IEEE Communications Magazine, Vol. 35, No. 8, August 1997.

[CB96] Cheshire, S., Baker, M., "Experiences with a Wireless Network in MosquitoNet," IEEE Micro, February 1996. Available online as: http://rescomp.stanford.edu/~cheshire/papers /wireless.ps.

[CDMA] Electronic Industry Alliance(EIA)/Telecommunications Industry Association (TIA), IS-95: Mobile Station-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System, 1993.

[CDPD] Wireless Data Forum, CDPD System Specification, Release 1.1, 1995.

[CM] Hari Balakrishnan and Srinivasan Seshan, "The Congestion Manager," Work in Progress.

[CTCSM97] Chang, H., Tait, C., Cohen, N., Shapiro, M., Mastrianni, S., Floyd, R., Housel, B., Lindquist, D., "Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express," in Proc. MobiCom'97, Budapest, Hungary, September 1997.

[Demers90] Demers, A., Keshav, S., and Shenker, S., Analysis and Simulation of a Fair Queueing Algorithm, Internetworking: Research and Experience, Vol. 1, 1990, pp. 3-26.

[ECN] Ramakrishnan, K. and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999.

[Floyd95] Floyd, S., and Jacobson, V., Link-sharing and Resource Management Models for Packet Networks. IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995.

[FSS98] Fragouli, C., Sivaraman, V., Srivastava, M., "Controlled Multimedia Wireless Link Sharing via Enhanced Class-Based Queueing with Channel-State-Dependent Packet Scheduling," Proc. IEEE INFOCOM'98, April 1998.

[GPRS] ETSI, "General Packet Radio Service (GPRS): Service Description, Stage 2," GSM03.60, v.6.1.1 August 1998.

[GSM] Rahnema, M., "Overview of the GSM system and protocol architecture," IEEE Communications Magazine, vol. 31, pp 92-100, April 1993.

[HL96] Hausel, B., Lindquist, D., "WebExpress: A System for Optimizing Web Browsing in a Wireless Environment," in Proc. MobiCom'96, Rye, New York, USA, November 1996.

[HTTP-PERF] Henrik Frystyk Nielsen (W3C, MIT), Jim Gettys (W3C, Digital), Anselm Baird-Smith (W3C, INRIA), Eric Prud'hommeaux (W3C, MIT), Hon Lie (W3C, INRIA), Chris Lilley (W3C, INRIA), "Network Performance Effects of HTTP/1.1, CSS1, and PNG," ACM SIGCOMM '97, Cannes, France, September 1997. Available at: http://www.w3.org/Protocols/HTTP/Performance /Pipeline.html

[IPPCP] Shacham, A., Monsour, R., Pereira, R. and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 2393, December 1998.

[IPHC] Degermark, M., Nordgren, B. and S. Pink, "IP Header Compression", RFC 2507, February 1999.

[IPHC-RTP] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[IPHC-PPP] Engan, M., Casner, S. and C. Bormann, "IP Header Compression over PPP", RFC 2509, February 1999.

[ITCP] Bakre, A., Badrinath, B.R., "Handoff and Systems Support for Indirect TCP/IP. In Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing, Ann Arbor, Michigan, April 10-11, 1995.

[Jain89] Jain, R., "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," Digital Equipment Corporation, Technical Report DEC-TR-566, April 1989.

[Karn93] Karn, P., "The Qualcomm CDMA Digital Cellular System" Proc. USENIX Mobile and Location-Independent Computing Symposium, USENIX Association, August 1993.

[KRLKA97] Kojo, M., Raatikainen, K., Liljeberg, M., Kiiskinen, J., Alanko, T., "An Efficient Transport Service for Slow Wireless Telephone Links," in IEEE Journal on Selected Areas of Communication, volume 15, number 7, September 1997.

[LAKLR95] Liljeberg, M., Alanko, T., Kojo, M., Laamanen, H., Raatikainen, K., "Optimizing World-Wide Web for Weakly-Connected Mobile Workstations: An Indirect Approach," in Proc. 2nd Int. Workshop on Services in Distributed and Networked Environments, Whistler, Canada, pp. 132-139, June 1995.

[LHKR96] Liljeberg, M., Helin, H., Kojo, M., Raatikainen, K., "Mowgli WWW Software: Improved Usability of WWW in Mobile WAN Environments," in Proc. IEEE Global Internet 1996 Conference, London, UK, November 1996.

[LS98] Lettieri, P., Srivastava, M., "Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency," Proc. IEEE INFOCOM'98, April 1998.

[MNCP] Piscitello, D., Phifer, L., Wang, Y., Hovey, R., "Mobile Network Computing Protocol (MNCP)", Work in Progress.

[MOWGLI] Kojo, M., Raatikainen, K., Alanko, T., "Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network," in Proc. Workshop on Mobile and Wireless Information Systems (MOBIDATA), Rutgers University, NJ, November 1994. Available at: http://www.cs.Helsinki.FI/research/mowgli/. Revised version published in Mobile Computing, pp. 253-270, Kluwer, 1996.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J., Ott, T., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," in Computer Communications Review, a publication of ACM SIGCOMM, volume 27, number 3, July 1997.

[MTCP] Brown, K. Singh, S., "A Network Architecture for Mobile Computing," Proc. IEEE INFOCOM'96, pp. 1388-1396, March 1996. Available at ftp://ftp.ece.orst.edu/pub/singh/papers /transport.ps.gz

[M-TCP] Brown, K. Singh, S., "M-TCP: TCP for Mobile Cellular Networks," ACM Computer Communications Review Vol. 27(5), 1997. Available at ftp://ftp.ece.orst.edu/pub/singh/papers/mtcp.ps.gz

[MV97] Mehta, M., Vaidya, N., "Delayed Duplicate-Acknowledgements: A Proposal to Improve Performance of TCP on Wireless Links," Texas A&M University, December 24, 1997. Available at http://www.cs.tamu.edu/faculty/vaidya/mobile.html

[NETBLT] White, J., "NETBLT (Network Block Transfer Protocol)", Work in Progress.

[Paxson97] V. Paxson, "End-to-End Internet Packet Dynamics," Proc. SIGCOMM '97. Available at ftp://ftp.ee.lbl.gov/papers/vp-pkt-dyn-sigcomm97.ps.Z

[RED] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J. and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[RLP] ETSI, "Radio Link Protocol for Data and Telematic Services on the Mobile Station - Base Station System (MS-BSS) interface and the Base Station System - Mobile Switching Center (BSS-MSC) interface," GSM Specification 04.22, Version 3.7.0, February 1992.

[RFC908] Velten, D., Hinden, R. and J. Sax, "Reliable Data Protocol", RFC 908, July 1984.

[RFC1030] Lambert, M., "On Testing the NETBLT Protocol over Divers Networks", RFC 1030, November 1987.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1144] Jacobson, V., "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[RFC1151] Partridge, C., Hinden, R., "Version 2 of the Reliable Data Protocol (RDP)", RFC 1151, April 1990.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1397] Braden, R., "Extending TCP for Transactions -- Concepts", RFC 1397, November 1992.

[RFC1644] Braden, R., "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994.

[RFC1661] Simpson, W., "The Point-To-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D. and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996.

[RFC1986] Polites, W., Wollman, W., Woo, D. and R. Langan, "Experiments with a Simple File Transfer Protocol for Radio Links using Enhanced Trivial File Transfer Protocol (ETFTP)", RFC 1986, August 1996.

[RFC2002] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2004] Perkins, C., "Minimal Encapsulation within IP", RFC 2004, October 1996.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2188] Banan, M., Taylor, M. and J. Cheng, "AT&T/Neda's Efficient Short Remote Operations (ESRO) Protocol Specification Version 1.2", RFC 2188, September 1997.

[RFC2246] Dierk, T. and E. Allen, "TLS Protocol Version 1", RFC 2246, January 1999.

[RFC2414] Allman, M., Floyd, S. and C. Partridge. "Increasing TCP's Initial Window", RFC 2414, September 1998.

[RFC2415] Poduri, K.and K. Nichols, "Simulation Studies of Increased Initial TCP Window Size", RFC 2415, September 1998.

[RFC2416] Shepard, T. and C. Partridge, "When TCP Starts Up With Four Packets Into Only Three Buffers", RFC 2416, September 1998.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2582] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[SNOOP] Balakrishnan, H., Seshan, S., Amir, E., Katz, R., "Improving TCP/IP Performance over Wireless Networks," Proc. 1st ACM Conf. on Mobile Computing and Networking (Mobicom), Berkeley, CA, November 1995.

[Stevens94] R. Stevens, "TCP/IP Illustrated, Volume 1," Addison-Wesley, 1994 (section 2.10 for MTU size considerations and section 11.3 for weak checksums).

[TCPHP] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[TCPSATMIN] TCPSAT Minutes, August, 1997. Available at: http://tcpsat.lerc.nasa.gov/tcpsat/meetings/munich-minutes.txt.

[Touch97] Touch, T., "TCP Control Block Interdependence", RFC 2140, April 1997.

[Vaidya99] N. H. Vaidya, M. Mehta, C. Perkins, G. Montenegro, "Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless," Technical Report 99-003, Computer Science Dept., Texas A&M University, February 1999.

[VEGAS] Brakmo, L., O'Malley, S., "TCP Vegas, New Techniques for Congestion Detection and Avoidance," SIGCOMM'94, London, pp 24-35, October 1994.

[VMTP] Cheriton, D., "VMTP: Versatile Message Transaction Protocol", RFC 1045, February 1988.

[WAP] Wireless Application Protocol Forum. http://www.wapforum.org/

[WC91] Wang, Z., Crowcroft, J., "A New Congestion Control Scheme: Slow Start and Search," ACM Computer Communication Review, vol 21, pp 32-43, January 1991.

[WTCP] Ratnam, K., Matta, I., "WTCP: An Efficient Transmission Control Protocol for Networks with Wireless Links," Technical Report NU-CCS-97-11, Northeastern University, July 1997. Available at: http://www.ece.neu.edu/personal/karu/papers/WTCP-NU.ps.gz

[YB94] Yavatkar, R., Bhagawat, N., "Improving End-to-End Performance of TCP over Mobile Internetworks," Proc. Workshop on Mobile Computing Systems and Applications, IEEE Computer Society Press, Los Alamitos, California, 1994.

Authors' Addresses

Questions about this document may be directed at:

Gabriel E. Montenegro
Sun Labs Networking and Security Group
Sun Microsystems, Inc.
901 San Antonio Road
Mailstop UMPK 15-214
Mountain View, California 94303

Phone: +1-650-786-6288
Fax: +1-650-786-6445
EMail: gab@sun.com

Spencer Dawkins
Nortel Networks
P.O. Box 833805
Richardson, Texas 75083-3805

Phone: +1-972-684-4827
Fax: +1-972-685-3292
EMail: sdawkins@nortel.com

Markku Kojo
Department of Computer Science
University of Helsinki
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 HELSINKI
Finland

Phone: +358-9-1914-4179
Fax: +358-9-1914-4441
EMail: kojo@cs.helsinki.fi

Vincent Magret
Corporate Research Center
Alcatel Network Systems, Inc
1201 Campbell
Mail stop 446-310
Richardson Texas 75081 USA
M/S 446-310

Phone: +1-972-996-2625
Fax: +1-972-996-5902
EMail: vincent.magret@aud.alcatel.com

Nitin Vaidya
Dept. of Computer Science
Texas A&M University
College Station, TX 77843-3112

Phone: 979-845-0512
Fax: 979-847-8578
EMail: vaidya@cs.tamu.edu

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.