NASA/CR—2005-213838

Adapting End Host Congestion Control for Mobility

Wesley M. Eddy
Verizon Federal Network Systems, Cleveland, Ohio

Yogesh P. Swami
Nokia Research Center, Irving, Texas

September 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—2005-213838

Adapting End Host Congestion Control for Mobility

Wesley M. Eddy
Verizon Federal Network Systems, Cleveland, Ohio

Yogesh P. Swami
Nokia Research Center, Irving, Texas

Prepared under Contract NAS3–03100

National Aeronautics and
Space Administration

Glenn Research Center

September 2005

Acknowledgments

Khiem Le, Nokia, helped develop the Internet draft that specifies the LMDR behavior. Several participants in the IETF's TSVWG and TCPM groups provided feedback on this work. Joseph Ishac and other members of NASA's ACAST Architectures and Networks group gave useful analysis of this work.

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Adapting End Host Congestion Control for Mobility

Wesley M. Eddy
Verizon Federal Network Systems
Cleveland, Ohio 44135

Yogesh P. Swami
Nokia Research Center
Irving, Texas 75603

ABSTRACT

Network layer mobility allows transport protocols to maintain connection state, despite changes in a node's physical location and point of network connectivity. However, some congestion-controlled transport protocols are not designed to deal with these rapid and potentially significant path changes. In this paper we demonstrate several distinct problems that mobility-induced path changes can create for TCP performance. Our premise is that mobility events indicate path changes that require re-initialization of congestion control state at both connection end points. We present the application of this idea to TCP in the form of a simple solution (the Lightweight Mobility Detection and Response algorithm, that has been proposed in the IETF), and examine its effectiveness. In general, we find that the deficiencies presented are both relatively easily and painlessly fixed using this solution. We also find that this solution has the counter-intuitive property of being both more friendly to competing traffic, and simultaneously more aggressive in utilizing newly available capacity than unmodified TCP.

1. INTRODUCTION

The Internet's routing architecture is designed to statelessly move packets between hosts at fixed locations. In this regime, IP addresses provide host identifiers for network layer routing, and transport layer port number pairs are further used to identify individual connections between hosts. The IP addresses and port numbers must remain fixed for the lifetime of a connection, as the standard inter-layer interfaces have no mechanisms for dealing with mid-connection changes in a host's address, or an application's port numbers. This makes the natural approach to mobility, where a host's IP address changes to represent its location, undesirable, as it causes existing connections to break whenever either host moves. For this reason, various network layer mobility schemes have been proposed, that extend the routing infrastructure to allow a host to keep a fixed address despite changes in its location.

Several protocols exist for enabling host mobility at the network layer, including Mobile IPv4 (MIPv4) [22], Mobile IPv6 (MIPv6) [13], mobile router techniques [11], all-IP cellular networks [5], and HIP-based mobility [20]. The key feature shared by these protocols is that a mobile node retains some fixed address (or identifier), regardless of the IP addressing structure at the point where it is physically attached to the network. This feature allows transport bindings to static addresses to remain intact, and thus keep connections alive, despite mobility across diverse networks. In this paper, we show that hiding mobility events from the transport layer in this way can be detrimental to performance.

In the Internet's protocol stack, the transport layer is the lowest layer with any view of the end-to-end network path between two hosts. For this reason, the transport layer is a sensible place to implement end-to-end congestion control, and modern transport protocols for bulk-traffic include congestion control mechanisms to perform in a manner friendly to the network and to other traffic [8]. Presently-used congestion control techniques such as TCP congestion control [3] or TCP-Friendly Rate Control (TFRC) [10] use packet loss events (or ECN marks) as indications of a path's congestion level, and determine their sending behavior based on packet losses. These techniques provide a roughly accurate estimation of the path's available capacity at any given time, although TCP congestion control and TFRC differ widely in how they compute this estimate.

TCP congestion control has two distinct phases, *slow start* and *congestion avoidance*. TCP's slow-start algorithm quickly probes the amount of available capacity in a network path by doubling the rate it sends segments every round-trip time (RTT). This allows an upper bound to be quickly reached, when the first packet loss is detected. The steady-state congestion avoidance algorithm is used to keep the sending rate undulating near the rough capacity estimate determined during slow start. During congestion avoidance, the sending rate increases conservatively in a linear fashion.

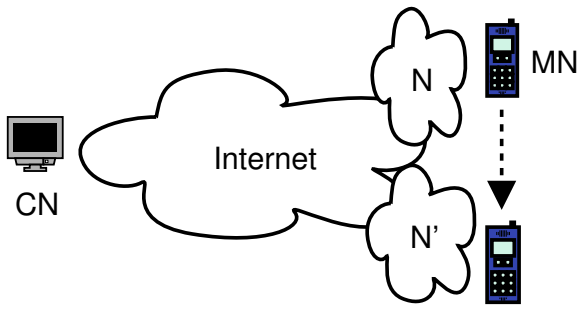


Figure 1: Host MN moves from network N to network N' while maintaining a TCP connection with host CN

Figure 1 illustrates a mobile node, MN, moving from a connectivity point on subnet N to one on subnet N'. The mobile node has an active TCP connection with some correspondent node CN, which remains intact across the transition using a network layer mobility protocol, such as MIPv4. The change in MN's attachment point, from subnet N to subnet N', implies a change in the end-to-end path that the connection's segments follow. Since the congestion control state (congestion window, slow start threshold, retransmission timeout, etc) of a connection is based upon estimates of the

end-to-end path, a path change may immediately invalidate some portion of the congestion control state. In figure 1, the networks are represented as clouds to signify that their topology is obscured from the end nodes, and the degree of path change is unknown. We assume that such mobility events occur relatively infrequently, no more than once per several dozen RTTs.

Since the significance of the path change between two attachment points is a mystery, there is no way for a connection to know whether or not its old path property estimates from subnet N are reasonable in subnet N'. For example, subnet N and subnet N' may be similarly configured and loaded networks who both attach to the Internet via a common point. In this case the difference is insignificant, and TCP's congestion state remains accurate even after the path change. However, there is no way of ensuring that this is the case, and a mobile node may just as easily move from a 54 Mbps 802.11g link, to a 384 kbps cellular link. Such changes are entirely possible, and are currently completely hidden from the transport layer, so that TCP does not even get an indication from a lower layer that a mobility event has occurred.

Regardless of whether or not subnet N and subnet N' use vastly different media, the end-to-end paths from them to CN may have substantially dissimilar properties that negatively influence TCP congestion control. For example, a mobile node might move from one wireless LAN access point to another, and yet experience a wide variation in path properties depending upon the network load and number of users. In some cases, the routing between subnet N and CN may be via an entirely different path than from subnet N' to CN. There are no guarantees about the significance or insignificance of the path change corresponding to a mobile node's changing points of attachment. They may lie anywhere on the spectrum from trivial to severe, and TCP is left to infer and adapt on its own.

Apart from path-dissimilarity, a mobile node's TCP performance is also influenced by the underlying mobility management scheme[1]. Broadly, the range of mobility management schemes can be classified into two categories, soft-handoff and hard-handoff, depending upon whether or not *packets in flight* to MN's old location on subnet N are lost after movement. In the case of a hard handoff, when the MN moves to subnet N', the access router in subnet N does not keep any state about MN's new location. Therefore, immediately after subnet change, packets in flight destined to MN's old location are lost. Since a full flight of loss (whether loss of data segments, or loss of acknowledgements) often results in an idle TCP retransmission timeout (RTO) wait period, a hard handoff results in lost throughput.

With soft-handoffs, the access router in subnet N keeps a soft-state mapping between the mobile node's old care-of address and its new address [14]. When a packet destined to the mobile node's old address arrives, the access router in subnet N tunnels those packets to subnet N', preventing losses. Because less packets are not lost during soft handoffs, there is less danger of a TCP retransmission timeout, but it is more likely that the mobile connection will temporarily behave unfairly if the new path is already congested.

The hard/soft handoff terminology for describing network layer mobility support should not be confused with similar link layer terminology. In the case of the network layer, soft handoff refers to the old access router's ability to forward packets to the new access router. In the link-layer, soft handoff refers to an interface's ability to re-associate with a new link without breaking the association on the old link. It is possible to have both soft and hard handoffs at the network layer with either kind of link layer technology.

[1]Appendix A provides brief descriptions of several mobility management schemes.

Although TCP behavior is influenced by the hardness or softness of network layer handovers, it can have problems with both types of underlying protocol, simply because TCP has no mechanisms for dealing with the quick change of path properties presented to it. Some protocols route all packets through an indirection point, like a MIPv4 Home Agent when bi-directional tunneling is used [18], while others provide a means of route optimization whereby a more efficient path can be used. Although these protocol features have some effect on the potential path change, they do not constrain it. Additionally, some protocols provide means for "fast" or "smooth" transitions. This may mitigate losses and latency during the change of network attachment points, making TCP have less recovery work to do, but it also does not limit the degree of potential difference in end-to-end network path properties. The exact network-layer mobility strategy used is mostly irrelevant to TCP, which always has the task of quickly adapting to a new and potentially completely unknown path.

Section 2 outlines a number of problems that arise when congestion control is oblivious to mobility events. In Section 3, we describe a means to make network layer mobility events less transparent to TCP. We also outline a response algorithm for TCP-like congestion control that should address the problems described in this paper. The effectiveness of our approach is then evaluated in Section 4, and some broad discussion of the mechanism is provided in Section 5.

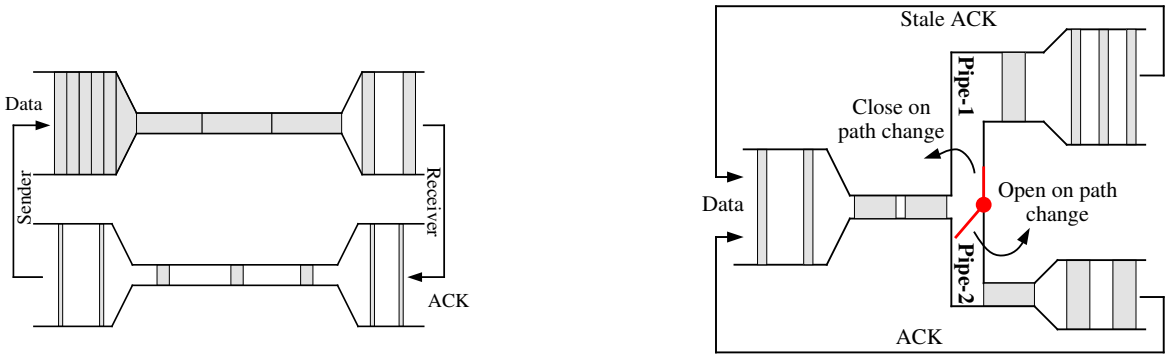
2. MOBILITY'S EFFECT ON CONGESTION CONTROL

TCP has a feedback response of increasing its congestion window for successful transmissions, and decreasing the window for losses (or ECN marks). This strategy is based upon the assumption that future segments will traverse the same basic path as past segments. Yet, the IP architecture provides a datagram service where each packet *may* be routed independently of all others, even when their sources and destinations are the same. Despite the potential independence in packet treatment, congestion control algorithms *assume* that once a connection is established, all segments follow the same path. More precisely, the assumption is that links, routing tables, propagation delays, maximum buffer capacities, link MTUs, and other path properties are mostly static, with the only variable being the amount of other traffic filling links and buffers at any time. This leads to the "network-pipe" model, described by Jacobson [12]. As demonstrated in figure 2(a), in this model, y-axis distances (pipe widths) represent link bandwidth, and x-axis distances (pipe lengths) represent time (queueing and propagation delays). The data stream flows through one set of pipes to the receiver, and acknowledgements flow through another set of pipes back to the sender.

Based on the network-pipe model, Jacobson's principles for congestion control can be understood as attempting to keep the network's pipes full, without overfilling them.

- To reach equilibrium, the sender should quickly probe the network for a capacity estimate. TCP's slow start algorithm satisfies this need, and once this estimate is made, the congestion avoidance algorithm takes over, and future probing is much less aggressive.

- A sender in equilibrium should follow the conservation of packets principle to avoid congestion. Conservation of packets is the idea that to maintain equilibrium, a new packet is only put onto the network after an old packet has left the network, keeping a constant number of packets in flight. TCP



(a) Visualization of TCP segments and acknowledgments moving through a network path, using the network-pipe model

(b) The split-pipe model adaptation, in which a valve switches packet flow from one pipe to another at the time of a mobility event.

Figure 2: Adapting the network-pipe model to reflect mobility

achieves this packet conservation through its ACK-clocking mechanism, sending new data upon receipt of acknowledgements for old data.

When a mobile node moves during the course of a TCP connection, the network path between the hosts changes in a way that is difficult to visualize using the standard network-pipe model. We propose an extension, illustrated in figure 2(b) called the split-pipe model, which captures this behavior. In the split-pipe model, mobility between network attachment points corresponds to switching the valve between the top and bottom pipes. Using this model, we can re-examine Jacobson's congestion control principles. Figure 2(b) has been simplified by abstracting out the ACK-carrying pipes, to focus on the data-carrying pipes. In theory, changes to the ACK path could be serious as well.

2.1 Problems with Soft Handoffs

If the underlying mobility management scheme allows soft handoffs, then for some period of time, acknowledgements for data segments sent through the top pipe will be received. We call these *stale* acknowledgments. Stock TCP uses these stale acknowledgements both to clock out data segments that travel through the bottom pipe, and to increase its sending rate. This behavior is not in line with the packet conservation principle, as stale acknowledgements do not indicate that segments have left the bottom pipe, where new segments are sent. Nor should stale acknowledgements be used to increment the sending rate into the bottom pipe, as they represent feedback about the top pipe's state, and convey no information about the bottom pipe. Stale acknowledgments should be used to indicate successful transmissions and remove data from the retransmission queue, but otherwise be ignored for the purposes of congestion control and clocking out new data.

In addition to ignoring stale acknowledgments for congestion control purposes, the behavior of attempting to maintain the equilibrium achieved in the top pipe, after a change to the bottom pipe, is unwise for a number of reasons. If the bottom pipe, is much larger, or has more free space, then the conservative congestion avoidance strategy can waste this by leaving it unused. In the opposite case, where the bottom pipe is much smaller than the top pipe, then using the old equilibrium state in the new pipe can lead to congestion losses. These problems are demonstrated quantitatively in Section 4. For now, we argue from the model and principles that slow start should be re-initiated after the change to the new path, from the connection's initial congestion window, and not the congestion window at the time of transition.

2.2 Problems with Hard Handoffs

Some network layer mobility protocols may cause up to an entire window of segments or acknowledgements to be lost. If this degree of loss occurs, then TCP senders may be forced to wait idly for a full retransmission timeout before beginning the process of repairing the losses and recalibrating the congestion window to the new network. The retransmission timeout is a rather long time frame for a sender with queued application data to pause for, typically representing several RTTs [21] (as measured in the old network before the transition). If a TCP sender has an amount of outstanding data that fills its congestion window, and acknowledgements for this data are lost due to the change in network paths, then this entire time is wasted.

The RTO wait period is particularly a problem in wireless networks because the RTO duration tends to be longer than when traditional wired links are used. One reason for high RTO values in wireless networks is that they often use link layer retransmissions to mitigate the effects of high bit error rates [7]. Because of link layer retransmissions, the measured RTT varies significantly causing the RTT-variance to increase. Since the RTO depends upon the RTT-variance, the wait periods tend to be rather large. Additionally, severe over-buffering of wireless links is a common practice, which leads to longer RTOs [9, 17]. In a real EGPRS test network at Nokia, retransmission timeouts have been routinely observed after handoffs between subnetworks.

Since outstanding data on the old network path does not contribute to congestion on the new path, where the TCP connection has no in-flight data, then this outstanding data should not prevent new (acknowledgement-generating and RTO-avoiding) segments from being sent over the new path. The ability to send data on the new network allows acknowledgements to come back which will indicate whether or not losses occurred during the transition and need to be repaired, and in either case, will allow a wasteful timeout to be avoided. In this case, avoiding the RTO itself is a

more effective approach than trying to detect and correct for spurious RTOs after the fact, as many techniques have been proposed to do [16, 23].

2.3 Invalid *ssthresh* After Handoff

The congestion control state of a TCP connection includes a variable, *ssthresh* (for the slow start threshold), which sets the boundary congestion window between TCP's exponential and linear increases in sending rate. When the congestion window is under *ssthresh*, TCP rapidly probes available capacity using slow start. When the congestion window reaches *ssthresh*, the more conservative congestion avoidance algorithm takes over. Initially, at the beginning of a connection, *ssthresh* is set to a high value. When a loss is first inferred via the fast retransmit mechanism, *ssthresh* is set to half of the present amount of outstanding data. After retransmission timeouts, TCP resets the congestion window to a single segment, uses slow start up to *ssthresh* and then enters congestion avoidance.

Initially, the high *ssthresh* value allows the exponential increase strategy during slow start to operate until the network's limit is reached. After this point, *ssthresh* is always set to some previously attained rate, so slow start is never again used to probe for fresh network capacity, but rather to simply get up to a previously known "safe" speed. This strategy assumes that the amount of available capacity remains somewhat close to the previously estimated values. With mobility between diverse networks, this may not be the case. Newly attached networks may offer multiple orders of magnitude in higher rates, which TCP congestion control will be unable to utilize. Particularly with long RTTs and high network capacities, the additive increase strategy is slow to explore higher rates.

Consider the case of a TCP connection that begins while two hosts are connected via a 384 kbps link with 100 ms of one-way propagation delay[2]. This scenario is designed to be simple for demonstration, not necessarily realistic. Such a TCP connection will have its *ssthresh* set to roughly 6 kB, assuming an RTT of around 250 ms, which is sufficient for keeping its congestion window in the range to reasonably utilize this particular network.

If, at some point, one host changes connection points, such that the new link (or path) between the two is identically configured as the old one, aside from the available capacity, which increases to 54 Mbps, keeping the stale *ssthresh* established on the old link prevents the new capacity from being efficiently used. To fully utilize the new network, the congestion window would need to reach nearly 1700 kB. Even with multiple kB segments, the linear march from 6 kB to 1700 kB would take an inordinate number of round-trip times. Figure 3 plots this time as a function of the ratio between RTT and segment size. Even with an RTT of only propagation delay (200 ms) and a segment size of 8 kB, this takes over 42 seconds. Using slow start, this time could be reduced to under 12 RTTs, or around 3 seconds.

While the given example is somewhat contrived, similar real-world scenarios are not altogether inconceivable. For example, systems supporting communications for space exploration or air traffic management may have such widely varying types of links available to them, and frequently transition connections between links due to fading, line-of-sight blocking, noise in a frequency band, or other link disturbances.

The slow start threshold is a TCP state variable whose purpose is to prevent the large burst of losses that slow start can cause. The adjustment rules do not allow for rapid probing of newly available

[2]Throughout this paper, all link buffers are configured using the $B = (\overline{RTT} \times C)/\sqrt{n}$ rule [4]

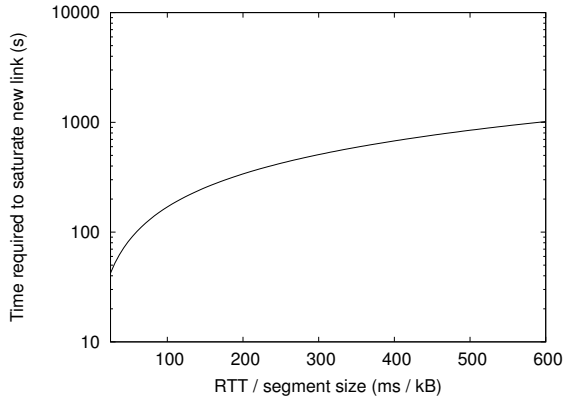


Figure 3: Relation between segment size, RTT, and minimum time to reach 54 Mbps when *ssthresh* is 6 kB

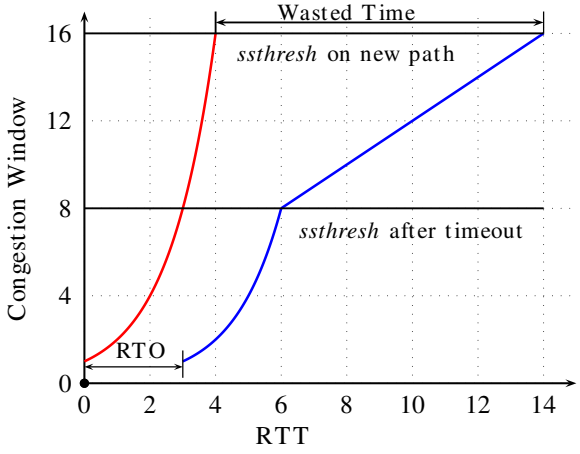


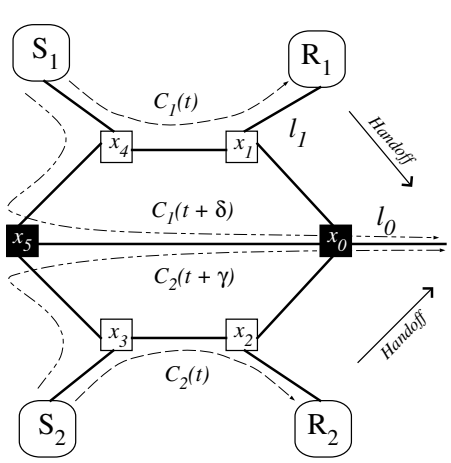
Figure 4: Effect of RTO wait period and stale *ssthresh* on congestion window, after a hard handoff.

capacity after the initial estimation. Since the available capacity can significantly change with network layer mobility between distinct types of networks; when transitions take place, the slow start threshold should be re-initialized to a large value to permit for rapid probing of available capacity in the new network. This is perfectly allowable under present congestion control principles, as it produces the exact same effect as a new flow starting up. Figure 4 illustrates an example time benefit that can be achieved by avoiding an RTO and reseting *ssthresh* after a mobility event where more capacity becomes available.

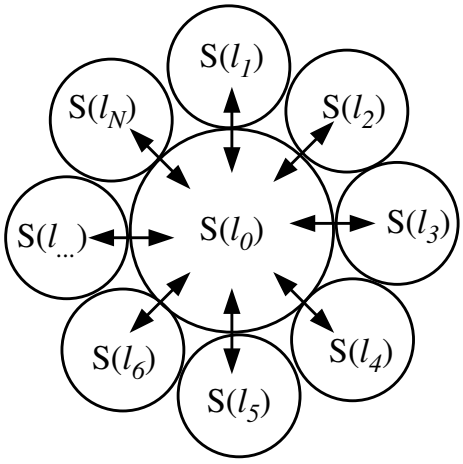
2.4 Network Design, Provisioning, and Stability

We have assumed mobility events to be infrequent over the lifetime of an individual connection, but within the entire network there may be many mobile hosts, each with several connections. While a single host's congestion control behavior will certainly influence the performance of its own connections and others sharing the same links, the design and stability of the network as a whole may be an issue as the number of mobile nodes or connections increases.

Figure 5(a) shows two connections, C_1 and C_2 that are termi-



(a) Set of routers that are affected by a path change.



(b) System view when multiple connections cross subnet boundaries.

Figure 5: System View of Subnet Change

nated at mobile nodes R_1 and R_2. At time t, C_1's segments flow through routers x_4 and x_1, but at time $t+\delta$, R_1 moves from link l_1 to link l_0. The new path which segments take is through routers x_4, x_5, and x_0. With soft-handoff support, x_1 tunnels segments for R_1 to x_0. The link between these two routers must be adequately provisioned to accept the sudden burst of tunneled traffic, which leads the design of the network to include expensive high-speed links between access routers, which would otherwise not be needed.

The tunneled segments from the soft-handoff will generate acknowledgements and cause data to be clocked out through the new path, where some of the routers (x_0 and x_5) have not seen previous segments from the connection, and may already be carrying a mix of flows that can't immediately accommodate the demands of C_1. While x_0 handles soft-handoffs and could be specially designed for such situations, x_5 could be a generic router with no relation to the portion of the network specifically designed for mobile nodes. Over-designing the edge networks and routers (like x_0) only pushes the problem upstream to other routers (like x_5).

Some networkers have argued that this is really not a problem, as the core of the Internet is known to be well over-provisioned anyways. To rely on this property is dangerous and potentially expensive. We cannot accurately extrapolate the current state of network utilization to the future, as history has shown that new applications and ways to use the network emerge and are adopted very quickly. Furthermore, the Internet protocol suite is used in other realms than the global Internet (for instance, military and space exploration networks), which have different requirements and may not be as overdesigned. We desire congestion controllers that behave reasonably across all potential paths, not just the kinds that are presently most likely on the Internet.

The problem of network design becomes even more difficult, considering that in figure 5(a), connection C_2 also sends segments through x_5 and x_0 after R_2 moves at time $t+\gamma$. Even if the network could handle C_1, or if the disturbance from C_1 were to subside after a short period of time, another disturbance would occur from C_2 some time later ($\gamma - \delta$). Since the mobility of R_1 and R_2 may be independent, there is no way to ensure that γ and δ are sufficiently far apart for the network to handle. With even more mobile nodes, or more connections per mobile host, the overall stability of portions of the network could be at stake.

With some additional knowledge of average connections from mobile nodes and frequencies of movement, the network design problem is still not easy, even at the edges or between the access routers that handle soft-handoffs. If $S(l_i)$ is the set of mobile nodes connected to some link l_i, any of those nodes may become connected to other links, and any nodes from other links may become members of $S(l_i)$. Figure 5(b) shows the relation between a network l_0 and N surrounding networks $l_1 ... l_N$, for the purpose of designing l_0 to accommodate mobility to and from each of the nearby networks. Not only would the soft-handoff links between all these networks be expensive, and likely redundant, but for cost-effectiveness, the provisioning would only be sufficient based on some estimates of average (or worst-case) motion between l_0 and each neighbor and the number of new flows that start and old flows that stop as nodes stay in each network. Given that with wireless networks, the neighbor set can easily grow and that the system is already provisioned based on several dicey estimates, the future adequacy of the overall system design is questionable, even if it is sufficient at some point in time.

Fixing end-host congestion control algorithms to respond to mobility events seems like a far more fruitful, and cheaper, approach than attempting to design the network infrastructure to accommodate sudden changes in offered loads. This puts the onus of responsibility on the end hosts where mobility actually occurs and takes it off innocent links and routers. Losses at properly configured static links and routers are the fault of end hosts' sending patterns, and can be more easily prevented by the end hosts than inside the network.

2.5 Other Path Properties

Aside from the congestion window and *ssthresh* variables that store estimated path state information, TCP also makes an estimate of the RTT. This RTT estimate is used to compute the retransmission timeout. Because instantaneously measured RTTs may vary

widely due to network buffering for congestion, previous estimates contribute to the current RTT estimate at any given time. This is a desirable practice when the measured RTTs have only transient or insignificant differences due to queueing; however, if a recently measured RTT differs from the current estimate significantly due to a path change, using the standard RTT update strategy is a hindrance. Given significant differences, convergence of the RTT estimate to a value representative of the new path may be slow. Updating the RTT estimate based on stale acknowledgements can also lead to a poor estimate.

An invalid RTT estimate makes the retransmission timeout either too long or too short, depending on the direction the RTT estimate is in error. If the RTT estimate is overly long for the new network, then the RTO timer is slow to fire and the amount of idle time before retransmission is needlessly long. If the estimate is too short, then segments may be spuriously retransmitted, with the congestion window unnecessarily reduced. In either case, this is harmful to TCP throughput. Both overly conservative and overly aggressive RTOs can be mostly avoided in the case of mobility, though. If a mobile host can simply re-initialize its RTT estimate with the first available information it gets from a new path, then its RTO timer will be reasonably set. When a path change is known to have occurred, there is no logic behind letting old estimates from a defunct path cloud TCP's judgment.

To avoid IP fragmentation, some TCP implementations execute path MTU discovery algorithms, although this can be problematic [15]. Since the path MTU can change when the path does, the discovery procedure should restart after mobility events. In addition, other more experimental path properties that are measured by a particular TCP implementation should also be re-estimated. For example, some experimental TCPs use estimations of a path's packet loss rate to alter congestion control behavior [2, 6]. In this case, biasing estimates in the new network based on data collected either in the old network or from stale acknowledgements could be troublesome. These types of considerations are not nearly as important as reseting the congestion window, *ssthresh*, and avoiding RTOs, but not to be ignored.

3. LIGHTWEIGHT MOBILITY DETECTION AND RESPONSE

The Lightweight Mobility Detection and Response (LMDR) algorithm has been proposed within the IETF as a way of avoiding the TCP problems described in the previous section. LMDR is designed to be independent of the underlying mobility management protocol. LMDR's only requirement is that a mobile node has some means of detecting its own mobility. In most cases, this requirement is easily satisfied. For example, standard neighbor discovery [19] procedures may be sufficient for this purpose[3].

Although a mobile node can detect its own mobility, its peers (e. g. the CN in figure 1) will often be unaware of this movement. Since a path change influences the congestion state in both directions of the connection, a mobile node must somehow inform its remote peers of local mobility events. To achieve this, LMDR uses a TCP option, that is specified to be reliable, even in the case where both nodes simultaneously move. This is the "mobility detection" portion of LMDR, as described in detail in Section 3.1.

[3]In some cellular networks such as (E)GPRS networks, the mobility information is not only hidden from the transport layer but also from the IP layer. A path change in these networks will not be detected by simple techniques like neighbor discovery. However, even in these networks it is possible to detect mobility events with the help of link layer protocols. The exact mechanism of how mobility events are detected is outside the scope of this document.

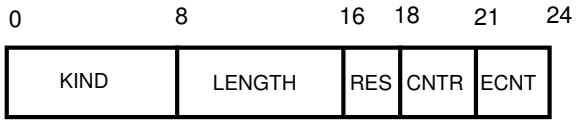


Figure 6: Wire-format of the LMDR TCP option

Once a host is able to detect potential path changes, either by monitoring its own mobility, or by receiving an LMDR TCP option, it can easily take corrective measures to address the problems sketched in Section 2. The "response" component of LMDR consists of performing these actions, and is detailed in Section 3.2. We consider LMDR to be a "lightweight" mechanism for several reasons. First, it requires no additions to the network architecture, and no changes to existing infrastructure components. Second, LMDR does not burden the network with additional probes, heartbeat timers, etc. And finally, LMDR does not introduce any new protocols, but merely a simple TCP option whose processing occurs infrequently and requires only a small number of state variables, and no expensive data structures or operations.

3.1 Mobility Detection

Detection of local mobility can be accomplished in numerous ways, depending on the underlying network layer and mobility management protocols. For example, there are neighbor, destination, and ARP caches in various protocols that can be consulted to infer if a host has moved. Alternatively, changes in a default router or observation of router advertisements might be used. Ideally, the lower layer mobility code would propagate information on mobility events up the protocol stack via some form of message passing or data sharing, but this is not how current kernel implementations of protocols work. How a transport layer infers mobility information from lower layers is beyond the scope of this paper.

A single host can unilaterally detect its own mobility and locally respond to it by fixing its own TCP state for the half-connection that it sends data over. This can be accomplished without any changes to the TCP wire protocol. However, if the half-connection in the reverse direction carries a large amount of data, the remote peer needs to be notified so that it can reset its TCP state. A new TCP option (shown in figure 6) is introduced for this purpose.

As standard for multi-byte TCP options, the first byte identifies the type (whose value is presently unassigned), and the second byte gives its length — an invariant 3 bytes. The next two bits of the third byte are reserved for future use, and should be set to zero by senders and ignored by receivers. Possible uses for these two bits are left for future work. The remaining 6 bits are divided into two 3-bit counters: `CNTR` and `ECNT`.

As with other TCP options, use of the LMDR option is negotiated at startup time on the SYN and SYN-ACK segments. A host wishing to use LMDR places an LMDR option on its SYNs. If the remote host supports LMDR, it responds to received LMDR options on SYNs, by placing an LMDR option on SYN-ACKs. Upon connection startup, `CNTR` is initialized to some random value, and `ECNT` is left uninitialized until the first `CNTR` value is received from the remote host. The `ECNT` field and variable are used to echo received `CNTR` values. Each time a host moves, it decrements `CNTR` (modulo 8), and advertises this by continuously transmitting LMDR options on its outgoing segments (both data-bearing and pure acknowledgements) until it receives back an LMDR option with an `ECNT` value that matches the local `CNTR` value. Upon receiving LMDR options, a host sets its `ECNT` variable to the received

CNTR value.

LMDR options are not sent on all segments. The only times when hosts need to send LMDR options are when they are informing peers of their own mobility, or confirming the reception of a peer's mobility notification. During normal exchange of data between mobility events, there is no need to transmit LMDR options on segments.

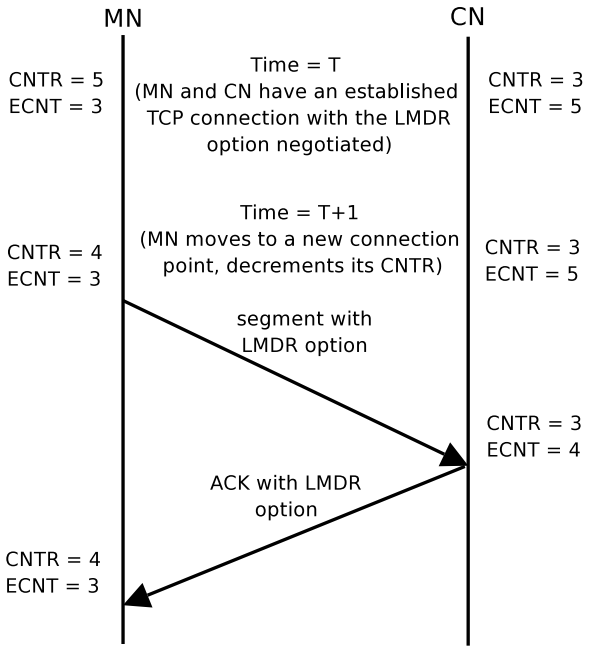


Figure 7: Example of MN notifying CN of a change in its attachment point

The effectiveness of LMDR is limited to situations where end hosts are the only mobile nodes, and the network infrastructure remains fixed. If network path changes are not caused by the mobility of an end host, but through attachment to a mobile router, then the simplistic means we have described for detecting mobility will be ineffective. The LMDR techniques could still be used, only if there were some means for mobile routers to notify attached end hosts of mobility events.

3.2 Mobility Response

Upon detection of local or remote mobility, several actions need to be taken. We have previously discussed these, but lay them out again here in a specific order for clarity.

1. Temporarily pause outgoing transmissions. Cancel the RTO and delayed acknowledgement timers.
2. Update values of CNTR and ECNT as necessary.
3. Record the highest sent sequence number as *stale*. Received acknowledgements for segments underneath *stale* will be considered stale and ignored for congestion control, RTT estimation, and data clocking purposes.
4. Reset the congestion window, *ssthresh*, RTO, RTT estimate, RTT variance, and other path properties as if this were a new connection. This automatically puts the connection into slow start territory, and allows at least one segment to be sent. This segment carries the LMDR option that either notifies the remote host of local mobility, or acknowledges receipt of an LMDR option generated by remote mobility.
5. Resume normal outgoing transmissions

Even, if use of the LMDR option is not successfully negotiated at connection startup, an individual host can still take most of the LMDR mobility response (everything but sending LMDR options) when it detects its own movement. This at least allows the half-connection it sources data on to better deal with rough path transitions.

Since this mobility response involves slow starting from the initial window, the impact of a mobile flow on the new network is the same as that of a totally new flow starting up, which even heavily congested networks are robust to. Furthermore, since slow start exponentially increases the congestion window, for a bulk-transfer flow, the amount of time required to reach the previous congestion window should be negligible. For more interactive flows, the temporary dip in the sending rate could be a problem, however few interactive applications that require smooth high rates use TCP.

4. SIMULATION-BASED EVALUATION

In this section, we use simulations to show that stock TCP (without LMDR) can behave undesirably after mobility events, and that adding LMDR simultaneously mitigates the potential for both overly aggressive and overly conservative behaviors.

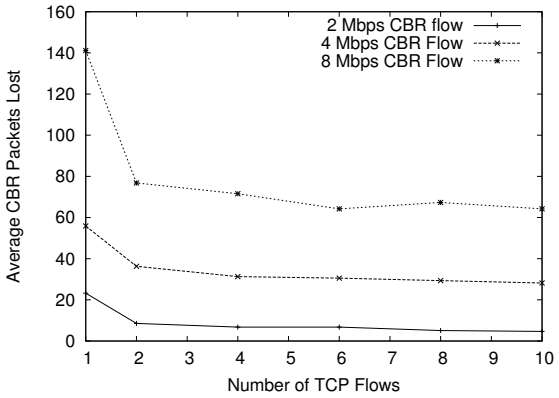
4.1 Improving Friendliness to Competing Traffic

Consider the case where a TCP connection endpoint moves such that in the new path, less capacity is available to it than in the previous path. With an inappropriately high congestion window, the connection will likely cause and experience some packet losses. While TCP's reaction to losses is a quick reduction in the congestion window, whether by half or down to a single segment, the reduction is delayed until losses are *inferred*, and does not take place either immediately when or before they occur. Mobility events that result in substantial path changes can make previous congestion window estimates invalid. Delaying the congestion window reduction until the first loss detection, rather than reducing it immediately after the change in attachment points, can lead to temporary increases in loss rates observed by both the TCP flow and other competing traffic on the new path.

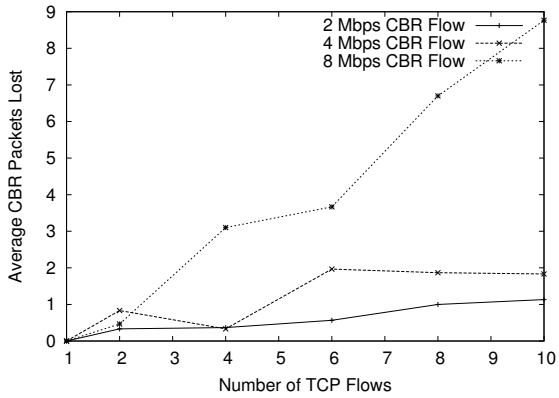
Since TCP senses the losses caused by its inappropriate congestion window, and adjusts it in a manner that quickly converges to an acceptable level, a single flow transitioning between networks is not a long-term threat to the congestion level or stability of the end-to-end path. Transient bursts of packet loss are expected and coped with relatively well by modern transport protocols. However, a node that moves once is likely to move again later, and if one node is allowed to move, than many nodes may also move. This makes the transient problem more serious, as it is bound to reoccur, perhaps frequently. This is harmful to the performance and fairness of TCP-like congestion control, the steadiness and friendliness of equation-based congestion control (like TFRC), and particularly damaging to unreliable real-time protocols that rely on stable path conditions.

Figure 8 shows the results from simulations[4] where some number of TCP flows establish themselves on a 10 Mbps path with

[4]Throughout this paper, all simulations are run using the ns-2 network simulator, with TCP/Sack1 senders, TCPSink/Sack1/Delack receivers, and all results averaged over 30 trial simulation runs. Simulation scripts are available from the authors upon request.

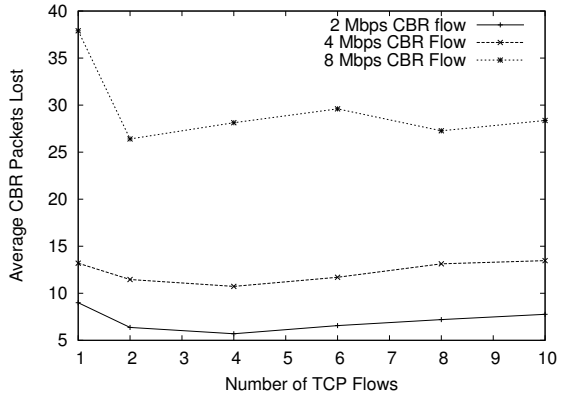


(a) Average number of CBR flow's packets lost during 120th second of simulation time, immediately following path change

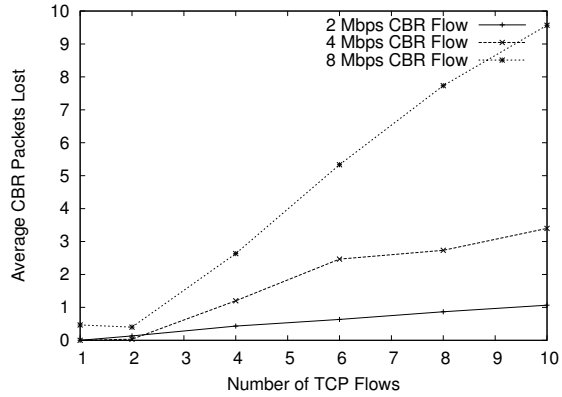


(b) Average number of CBR flow's packets lost during 180th second of simulation time, 60 seconds after path change

Figure 8: Increases in CBR loss rate due to temporarily unreasonable TCP congestion windows



(a) Average number of CBR flow's packets lost during 120th second of simulation time, immediately following path change



(b) Average number of CBR flow's packets lost during 180th second of simulation time, 60 seconds after path change

Figure 9: Demonstration of LMDR effectiveness in reducing temporary congestion spikes

100 ms of one-way propagation delay. After 2 minutes, the flows are then moved to another identical path, where a constant bit-rate (CBR) flow is already consuming some of the new path's capacity. Figure 8(a) plots the number of losses imposed on the existing CBR flow, during the first second after the transition, versus the number of TCP flows. The transitions are implemented such that there is no disconnection time between networks, and there is no loss of data segments, but all stale ACKs are lost (this is a hard handoff with the mobile node as data source). For comparison, the number of the constant bit-rate flow's packets that are lost during a single second a full minute after the transition, when the TCP congestion windows have converged to appropriate values for the new path, is plotted as well in 8(b). The difference between each set of lines is due to the inappropriate TCP congestion windows immediately after the transition.

One of the things this simulation shows us is that the problem is less severe when a large number of flows simultaneously make the path transition. This effect is due to the division of capacity between the flows. If more flows utilize the same capacity, then each individual flow has a smaller congestion window. The smaller congestion windows are more likely to result in RTOs on losses, which causes them to be idle for much longer periods of time than a lesser number of flows, which have large enough congestion windows to repair most losses via fast retransmissions. Another important effect of having more flows, is that the acknowledgements are less regularly spaced, causing the data segments that are lost and trigger congestion window reductions, to be sent less synchronously between flows. This allows flows to detect losses in a more staggered fashion, keeping the overall congestion level a bit lower and more stable than when all flows in unison blast out flights of packets into a congested network.

Figure 9 shows the results of the exact same simulations summarized in figure 8, only in this case, the TCP flows all use LMDR to rapidly adjust themselves to the new path. Comparing the results shown in figure 9 to those in figure 8, the number of losses imposed on competing smooth traffic in the second of time immediately after the transition is greatly reduced in all cases with LMDR, while the steady state behavior 60 seconds later is similar in both cases. With the 4 and 8 Mbps CBR flows, the raw number of packet losses was reduced by over 50% during the initial second after the transition. This makes the transition much less disruptive to existing traffic on the new network path, while attaining the same steady-state result.

4.2 Faster Utilization of Newly Available Capacity

In Section 2.3, we demonstrated how not reseting *ssthresh* can cause under-performance if a node moves into a new network path that can accommodate a much greater rate. Although it is fairly clear that reseting *ssthresh* in such a case allows for faster probing of available capacity, we provide simulation results here as an example of the quantifiable gain that can be expected when using LMDR in such situations, and look at the impact on performance over several time-scales. Our simulation reproduces the scenario described in Section 2.3, where a mobile node moves from a point where 384 kbps is available, to a point where 54 Mbps is available, with the same round-trip latency. This simulation is repeated 30 times with varying numbers of TCP flows simultaneously making the transition, and with both LMDR support, and unmodified TCP.

Figure 10 illustrates the results of these simulations by plotting the improvement in an average flow's throughput when LMDR is used by all connections compared to when it is not used by any of the connections. This is computed using the total throughput over various amounts of simulation time, with the transition between networks happening after 2 minutes. If less time after the transition is taken into account, then the LMDR advantage is more significant, as shown in figure 10, This plots the gain in total throughput, as computed 30, 60, and 120 seconds after the transition. Gains are due to LMDR's much more rapid convergence (exponential versus linear) on the new available capacity. After stock TCP probes the newly available capacity, there is no difference in behavior between it and LMDR, and each reach nearly the same capacity.

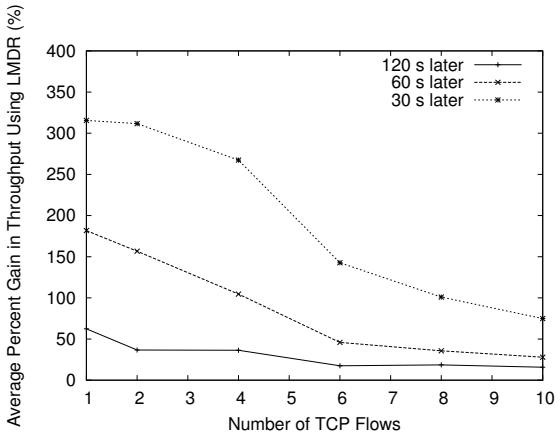


Figure 10: Example throughput gain when using LMDR after moving from 384 kbps link to 54 Mbps link

If we zoom-in to a time-scale covering only several RTTs after the transition, we can see that by resetting to the initial congestion window, LMDR may temporarily send more slowly than stock TCP (if the stock TCP isn't in the midst of a retransmission timeout). This is quickly reversed though, and the temporary degradation in performance is a necessary product of taking the correct action based on congestion control principles derived from the split-pipe model in Section 2.

From these example simulations, we see that LMDR can cause mobile connections to be less offensive to other traffic, and aid performance when transitions create a path with much greater capacity. The LMDR behavior is simultaneously advantageous for both resource conservation and high utilization, satisfying both the community needs, by keeping the network stable, fair, and friendly, and selfish individuals, by providing higher performance when available.

4.3 Multiple Node, Multiple Network Tests

To this point, the results we have presented have been from rather simplistic simulations, with flows changing networks simultaneously. In this section, we introduce a slightly more complex simulation topology and scenario, in which multiple mobile nodes randomly move through multiple networks of differing uplink and downlink capacities. In this particular case, the results still indicate an average net gain from using LMDR, although it is more modest (on the order of several percent).

Figure 11 illustrates the topology used in these simulations. Two corresponding nodes (CN1 and CN2) have TCP connections to three mobile nodes each (MN1 through MN6). The mobile nodes are distributed amongst three access routers (AR1 to AR3). Initially, two mobile nodes connect to each access router, with mobility between access routers occurring randomly, with each mobile node moving once ever 2 minutes on average. The access routers are connected to the corresponding hosts using links of variable capacities ($c1$ through $c3$), and connected to each other for tunneling

($c4$ to $c6$). The simulator's mobile IP support is used over dynamic links, which produces hard handoffs.

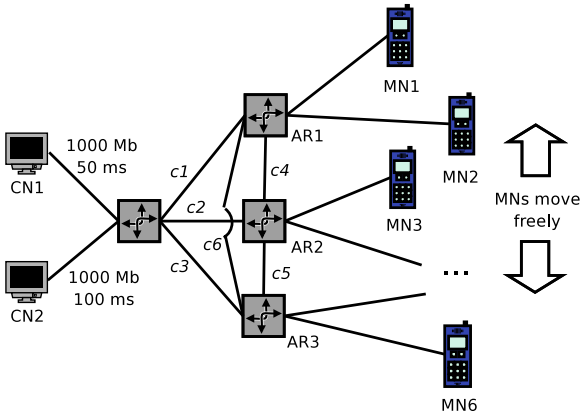


Figure 11: Topology Used for Multiple Node, Multiple Network Tests

We perform three different sets of simulations in this setup. In one set, the MNs source traffic, in another the CNs do, and in the final set, traffic is bidirectional. In each case, the link capacities $c1$ to $c6$ are set to particular values to make the experiment interesting. Each set is run for 30 iterations with the same random seed used in both a fully LMDR-enabled situation, and a fully LMDR-less situation, with each simulation lasting 5 minutes. We record the number of unique user data bytes transferred by each TCP flow over this time period.

In the case where the MNs alone source traffic, $c3$, $c4$, $c5$, and $c6$ were set to 500 Mbps, while $c1$ was set to 5 Mbps, and $c2$ to 50 Mbps. With this configuration, individual flows averaged a 14% gain in throughput by using LMDR, in comparison to their counterparts with the same mobility pattern in the accompanying simulation that did not use LMDR. Figure 12 plots the cumulative distribution function of throughput improvements observed in these simulations. Despite the solid improvement in average performance, clearly LMDR did not always increase a flow's throughput, and in many cases it accounted for a degradation. The sum of the throughputs in each simulation increased by 2.9%, on average.

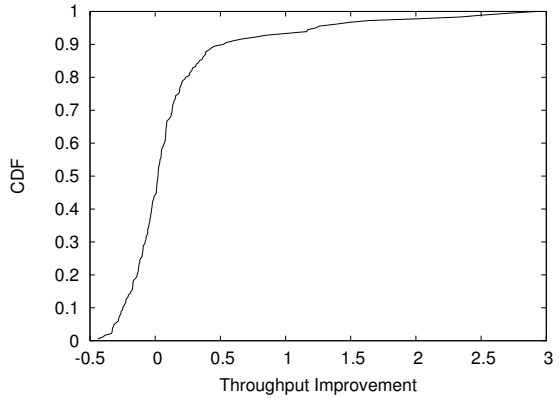


Figure 12: Cumulative Distribution Function of Throughput Gain With MNs Sourcing Data

When configuring the CNs to source data, we changed the topology such that $c1$, $c2$, $c3$, and $c6$ were 500 Mbps, $c4$ was 5 Mbps, and $c5$ was 50 Mbps. This caused all the flows to start out with fast links, and then after mobility events, have their segments tunneled through links of possibly much lesser capacity. This is different from our topology where the MNs sourced data, as in that case the tunneling links were all of high capacity, but the access links varied. In this case, the results still show a positive average gain from a flow using LMDR, of 3.2%. The CDF is not presented here, but is very similar to that of figure 12, except with the upper tenth percentile representing lesser gains. This accounts for the decrease in the mean.

For bidirectional traffic, we modified the link capacities with the topology again. In this case, $c1$ and $c4$ were set to 500 Mbps, $c2$ and $c6$ to 5 Mbps, and $c3$ and $c5$ to 50 Mbps. This set of simulations is somewhat different from the others in that there can be congestion in both the data and acknowledgement paths. With this configuration, we observed individual flows gain an average of 8.5% in throughput, although there were outliers that experienced large decreases. This further demonstrates that LMDR is not specifically designed to improve the performance of all flows, and may result in losses, but is gainful on average.

5. DISCUSSION

As mentioned previously, even though TCP can maintain connections over transparent network layer mobility protocols, simply maintaining a connection is not the same as providing an efficient level of performance and acting as a good network citizen. TCP continues to function without LMDR, and perhaps in the normal global Internet, the mobility scenarios we have shown to be problematic, will only rarely or never occur. Core networks may continue to be well-provisioned enough, that end-user demand and connectivity will limit the danger to the core's congestion level and stability posed by mobility-induced TCP problems at the edge. In this case, LMDR is probably not crucial to the architecture.

Unlike many other TCP extensions, LMDR *is not* explicitly intended to improve the throughput of TCP connections. Its purpose is to correct what we view as an oversight by the designers of transparent network layer mobility protocols. The Internet protocols are arranged in a stack, and changes to one layer should not be made without consideration of how these changes may affect higher layers. LMDR assumes there is some way by which the network layer's transparency can be reduced slightly so that the transport may know when local point of attachment changes occur. Since the transport layer is typically tasked with keeping estimates of end-to-end path properties, network layer protocols that knowingly change that path in a completely transparent way, are behaving negligently.

Given the knowledge that a path change has occurred, LMDR's behavior can be described as an attempt to take the most correct and conservative transport protocol behavior. As shown, LMDR is a better neighbor to competing traffic than stock TCP, when moving into a congested network path. We have also shown cases where by reseting *ssthresh* and avoiding retransmission timeouts, LMDR can achieve better throughput than stock TCP. There may be cases where using LMDR actually reduces a flow's performance by some small margin, however, this is clearly acceptable given the significance of its positive impact in other cases.

We have raised the point that the magnitudes of capacity differences in some of the path transitions we've looked at is probably unrealistic for today's Internet and common mobile devices. The IP protocol suite can be used in other environments than merely the Internet, though. Transitions between vastly different networks are not only possible, but highly likely for certain classes of military communications and NASA space-exploration missions, for which

TCP and IP may be used. Whether or not LMDR is needed for reasons of reducing congestion or aiding performance in the global Internet, it does not hurt in any way, and to not include it under the assumption that rare cases of problematic mobility scenarios will remain rare, is perhaps a mistake.

6. CONCLUSIONS AND FUTURE WORK

We have studied several problems TCP connections may experience after mobility events, and introduced the LMDR procedures to correct these issues. Simulations have shown that LMDR is an effective technique for combating these problems and improving TCP performance in mobile environments.

Other transport protocols have similar mechanisms to TCP's for estimating various path state properties, and may experience similar negative effects after path changes. The LMDR option for TCP offers no benefit to applications that do not use TCP as a transport. Other transport protocols may require adaptations of LMDR. For example, a "Reset Congestion State" option is present in an underdevelopment version of the DCCP base specification. This option could be sent by a host after detecting its own mobility and reseting its own invalid path state estimates, to ask the remote host to do the same.

We have mentioned that mobility of routers can cause similar path change problems as end host mobility, but that LMDR is helpless in this case. Some provisions for notification or detection of mobility inside the network path, and not just over links adjacent to the end hosts, would be beneficial in such cases, and could likely be easily incorporated into the LMDR detection facility. The LMDR response would likely be able to remain unchanged.

7. REFERENCES

[1] I. F. Akyildiz, J. Xie, and S. Mohanty. A Survey of Mobility Management in Next-Generation All-IP-Based Wireless Systems. *IEEE Wireless Communmications Magazine*, 11(4), Aug. 2004.

[2] M. Allman, W. M. Eddy, and S. Ostermann. Estimating Loss Rates with TCP. *ACM Performance Evaluation Review*, 31(3), Dec. 2003.

[3] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, Apr. 1999. RFC 2581.

[4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. *ACM SIGCOMM 2004, Stanford HPNG Technical Report TR04-HPNG-060800*, Aug. 2004.

[5] F. M. Chiussi, D. A. Khotimsky, and S. Krishnan. Mobility Management in Third-Generation All-IP Networks. *IEEE Communication Magazine*, 40(9), Sept. 2002.

[6] W. M. Eddy, S. Ostermann, and M. Allman. New Techniques for Making Transport Protocols Robust to Corruption-Based Loss. *ACM Computer Communication Review*, 34(5), Oct. 2004.

[7] G. Fairhurst and L. H. Wood. Advice to link designers on link automatic repeat request (ARQ). RFC 3366, Internet Engineering Task Force, Aug. 2002.

[8] S. Floyd. Congestion Control Principles, Sept. 2000. RFC

[9] A. Gurtov, M. Passoja, O. Aalto, and M. Raitola. Multi-layer Protocol Tracing in a GPRS Network. *Proc. of IEEE Vehicular Technology Conference*, Sept. 2002.

[10] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification, Jan. 2003. RFC 3448.

[11] W. Ivancic, D. Stewart, T. Bell, K. Leung, D. Shell, and B. Kachmar. Mobile Router Technology Development. *Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, July 2001.

[12] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM*, Aug. 1988.

[13] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6, June 2004. RFC 3775.

[14] R. Koodli and C. Perkins. Fast Handovers and Context Transfers in Mobile Networks. *ACM Computer Communication Review*, 31(5), Oct. 2001.

[15] K. Lahey. TCP Problems with Path MTU Discovery, Sept. 2000. RFC 2923.

[16] R. Ludwig and R. H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *ACM Computer Communication Review*, 30(1), Jan. 2000.

[17] R. Ludwig, B. Rathonyi, A. Konrad, K. Oden, and A. Joseph. Multi-layer Tracing of TCP Over a Reliable Wireless Link. *Proc. of ACM SIGMETRICS*, May 1999.

[18] G. Montenegro. Reverse Tunneling for Mobile IP, Revised, Jan. 2001. RFC 3024.

[19] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6), Dec. 1998. RFC 2461.

[20] P. Nikander, J. Ylitalo, and J. Wall. Integrating Security, Mobility, and Multi-Homing in a HIP Way. *Proceedings of Network and Distributed Systems Security Symposium (NDSS'03)*, Feb. 2003.

[21] V. Paxson and M. Allman. Computing TCP's Retransmission Timer, Nov. 2000. RFC 2988.

[22] C. Perkins. IP Mobility Support for IPv4, Jan. 2002. RFC 3220.

[23] P. Sarolahti, M. Kojo, and K. Raatikainen. F-RTO: An Enhanced Recovery Algorithm for TCP Retransmission Timeouts. *ACM Computer Communication Review*, 33(2), Apr. 2003.

APPENDIX

A. MOBILITY MANAGEMENT SCHEMES

Although most network layer mobility schemes achieve similar basic results, there is substantial variation among these protocols. In this appendix, we provide a brief overview of a small sample of mobility management schemes. We have restricted our discussion to IETF protocols only, although similar technologies exist [5, 1].

MIPv4 Mobile IPv4 hosts have static IP addresses. MIPv4 operation can be thought of as a coordination of three processes. First, a mobile host determines that it has moved from one network (home) to another (foreign), and in the new network, the host obtains a care-of address (either by obtaining a new IP address on the foreign network or by locating a foreign agent). Second, the mobile node registers its care-of address on the new network with its Home Agent. The Home Agent is an indirection point on the network that the mobile node's static

address belongs to. Third, as packets arrive on the home network addressed to the mobile node's static address, the Home Agent intercepts them and tunnels them to the mobile node's current care-of address.

Several factors can slow the time between the node's movement to a new network, and registration with the Home Agent. These include: detection of the network change (through router advertisements, etc), acquisition of a new address (via DHCP and duplicate address detection), authentication with the new network, and latency or packet losses when registering the care-of address with the Home Agent. During this time period, all packets sent to the mobile node's static address will be routed to an old care-of address, and not to the mobile node's current location.

MIPv6 Basically, Mobile IPv6 works in a similar fashion to MIPv4. Unlike MIPv4, MIPv6 includes *route optimization*, which can be used to bypass the Home Agent and send packets directly from corresponding nodes to a mobile node. This requires a two-message exchange (one round trip) for authentication, called the return routability test. This always adds an RTTof delay before packets can be sent on the new path, which can result in a full congestion window of data being sent to a stale address and lost.

The latency involved in establishing a new tunnel in MIPv6 has motivated the design of fast-handover techniques that involve coordination with access routers. This can prevent the burst packet losses that may occur during stock Mobile IP handovers.

HIP Mobility using the Host Identity Protocol is somewhat different, in that each node has another identifier that is independent from its IP address. Transports bind to these identifiers and the HIP layer takes care of mapping these to IP addresses. Changes to a node's IP address are relayed to peers and cryptographically authenticated by the HIP layer. Like Mobile IP's concept of the Home Agent, HIP also requires an indirection point, called the Rendezvous Server. The Rendezvous Server is only used to redirect the first packet to a mobile node. After this, nodes always send packets directly to each other.

The update after an IP address change in HIP requires at least a round trip before packets can be sent to the new address. Since HIP identifiers are not used by routers, no mechanism similar to fast-handovers for MIPv6 is available for HIP. This means that mobility events always result in an RTTwhere any data transmitted to a mobile node is lost.

NEMO The idea behind network mobility (NEMO) is to adapt the Mobile IP protocols to allow not just single hosts, but entire networks to be mobile, via the concept of a mobile router. This approach has many of the same advantages and disadvantages as the Mobile IP protocols, and adds some additional complexity, in that it may be more difficult for hosts to detect their own mobility in a NEMO setting. The IETF's NEMO working group is actively testing NEMO concepts and working on producing a standard.

Despite the many technical differences, we can broadly categorize mobility protocols into two classes: hard handoff and soft handoff. Hard handoff protocols may cause a large number of in-flight packets to be lost as detection, configuration, and registration occur after network transitions. MIPv4, MIPv6 (without fast-handover), and HIP are all hard handoff protocols. With fast-handover, MIPv6 is a soft handoff protocol, because it does not cause packets to the mobile node to be lost during transitions. NEMO handoffs could be made either hard or soft. By influencing packet loss, the distinction between hard and soft handoff protocols can significantly affect TCP's behavior.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2005	Final Contractor Report

4. TITLE AND SUBTITLE

Adapting End Host Congestion Control for Mobility

5. FUNDING NUMBERS

WBS–22–184–10–06
NAS3–03100

6. AUTHOR(S)

Wesley M. Eddy and Yogesh P. Swami

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Verizon Federal Network Systems
21000 Brookpark Road
Cleveland, Ohio 44135

8. PERFORMING ORGANIZATION REPORT NUMBER

E–15208

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—2005–213838

11. SUPPLEMENTARY NOTES

Wesley M. Eddy, Verizon Federal Network Systems, 21000 Brookpark Road, Cleveland, Ohio 44135; and Yogesh P. Swami, Nokia Research Center, 6000 Connection Drive, Irving, Texas 75603. Project Manager, Will Ivancic, Communications Division, NASA Glenn Research Center, organization code RCN, 216–433–3494.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 62

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Network layer mobility allows transport protocols to maintain connection state, despite changes in a node's physical location and point of network connectivity. However, some congestion-controlled transport protocols are not designed to deal with these rapid and potentially significant path changes. In this paper we demonstrate several distinct problems that mobility-induced path changes can create for TCP performance. Our premise is that mobility events indicate path changes that require re-initialization of congestion control state at both connection end points. We present the application of this idea to TCP in the form of a simple solution (the Lightweight Mobility Detection and Response algorithm, that has been proposed in the IETF), and examine its effectiveness. In general, we find that the deficiencies presented are both relatively easily and painlessly fixed using this solution. We also find that this solution has the counter-intuitive property of being both more friendly to competing traffic, and simultaneously more aggressive in utilizing newly available capacity than unmodified TCP.

14. SUBJECT TERMS

Computer networks

15. NUMBER OF PAGES

18

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	