The Journal of Supercomputing, 23, 225–244, 2002

TCP Performance and Behaviors with Local Retransmissions

JIANPING PAN, JON W. MARK AND SHERMAN X. SHEN

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, ON N2L3G1, Canada

Abstract. TCP has been the dominant transport protocol over the global Internet, and its performance over a hybrid wireless/wireline network has attracted much attention in recent years. This paper investigates the end-to-end TCP performance, in terms of normalized throughput, effective goodput, and packet delay, over wireless lossy links with local retransmissions. The results reveal that local retransmissions can increase the normalized TCP throughput in different wireless bandwidth, delay, and error settings, at the cost of a decrease in effective goodput and an increased packet delay. The performance observation is explained by the explored TCP endpoint behaviors, including the spurious timeout and duplicated acknowledgment. Analysis shows that spurious timeouts with local retransmissions are rare due to the conservative TCP timeout algorithm. However, spurious duplicated acknowledgments have negative impact and a further improvement with the D-SACK proposal is evaluated.

Keywords: Internet, cellular systems, local retransmission, TCP performance and behavior, simulation

1. Introduction

Although traditional cellular systems were primarily designed and extensively optimized for voice communications, certain circuit or packet switching data services [1, 2] have been offered for some time. Currently there are some observations of interest. With the Internet popularization, data traffic has already exceeded, in volume, the voice traffic in wired networks, and wireless networks are anticipated to experience the same process soon. Furthermore, cellular systems now are evolving into their third generation with a higher speed radio interface, better differentiated services, and a more efficient packet switching network backboned by the global Internet.

However, the performance of Internet protocols, particularly TCP/IP, over the current and future cellular systems [3, 4] is still quite uncertain [5]. Many new protocol stacks and tunneling protocols [6, 7] have been proposed for cellular data services, but their horizontal interface to and the possible interference with the dominant TCP/IP protocols remain open issues. Although in recent years there are extensive research efforts [9] in this and other related areas, the main targets are macro mobility support over the global Internet and TCP/IP performance over wireless **Local Area Network**s (LAN) or satellite links [8]. Cellular systems have their own mobility support mechanisms in the link layer. Therefore, the incorporation of **Mobile IP** [13], a network layer mobility solution, and cellular systems with the additional requirements of micro mobility support, security, and accountability certainly is a real challenge. Furthermore, cellular systems have their own complicated

Radio Layer Protocol (RLP) with the link layer flow and error controls, and their vertical interference [18] with the upper layer control logics deserves further study.

These challenges and open questions motivate the work reported in this paper. It focuses on the local retransmission mechanism which is a widely used technique in cellular systems and other wireless networks. Triggered by the selective acknowledgment, this technique applies not only in the lower layer, *e.g.*, by **Media Access and Link Control** (MAC/LLC) protocols [20, 21], but also in the higher layer, *e.g.*, by a *Snoop* [22] agent in the transport layer at the wireless/wireline interworking node. Instead of simply concluding that the local retransmissions technique is good or bad for the overall system in a general setting, which was heavily debated before [18, 19, 21], several representative networking scenarios with different wireless bandwidth, delay, and error settings in current and incoming cellular systems are studied here. The purpose of this work is to examine in which scenarios this technique is beneficial to the TCP performance most and in which scenarios it is not, and more importantly to understand what and how it has to trade off. The observed performance should be explainable by and consistent with the TCP endpoint behaviors, *e.g.*, timeout and **duplicated acknowledgment** (*dupack*). It reveals that spurious timeout is rare with local retransmissions due to the conservative nature of the modern TCP timeout algorithm. However, spurious *dupack*s do have negative impact on the overall performance and further improvement can be achieved with the D-SACK [17] proposal.

The rest of this paper is organized as follows. Related work on improving the TCP/IP performance over wireless links with the link or transport layer approach are outlined in Section 2. Section 3 presents the system model for the hybrid wireless and wireline networks, as well as the networking scenarios, TCP variants, and application traffic under consideration. The end-to-end TCP performance, in terms of normalized throughput, effective goodput, and packet delay, over wireless lossy links with local retransmissions in cellular systems are studied in detail in Section 4. The TCP endpoint behaviors, including the spurious timeout and *dupack*, are explored in Section 5 to explain the observed performance and to motivate a performance improvement by integrating the recent D-SACK proposal. Summary and the future work in Section 6 finally conclude the entire paper.

2. Related work

TCP was carefully designed to work reasonably well over any IP-capable networking technologies, no matter wired or wireless networks. However, the extensive follow-on re-engineering on TCP during the Internet evolution, most noticeably the TCP congestion control [23], has made it vulnerable over any lossy links. Current TCP congestion control simply assumes that any packet loss is due to network congestion. Therefore it misbehaves when the packet loss is indeed due to transmission errors, which is commonly found in many emerging technologies, *e.g.*, infrared and radio networks, cellular systems, and satellite links. There are many schemes proposed recently in the literature [9, 10] that follow either the link or transport layer approaches to mitigate the TCP performance degradation over wireless lossy links.

Other features in these systems which become more and more popular in the heterogeneous Internet today, *e.g.*, asymmetric MAC schemes and large transmission delay, also have impacts on the TCP performance. Mobility support in wireless and Ad Hoc networks is another related issue.

The link layer approach usually proposes a better and smarter link protocol, *e.g.*, utilizing Forward Error Correction, retransmitting locally triggered by selective acknowledgments, differentiating reliable and unreliable services, or just being TCP aware, to mitigate the impact of link impairment to the upper layer performance. However, their ability attracts a lot of debates, *e.g.*, when the link outage occurs, the upper layer still cannot behave adaptively as expected. Furthermore, their vertical interference to upper layers, *e.g.*, competitive retransmissions in both transport and link layers, is still questionable [18, 19]. The transport layer approach, on the contrary, tries to help the TCP endpoint to discriminate the loss source by introducing additional functionality in the receiver, sender, or wireless/wireline interworking node. The receiver or interworking node can postpone [27] or even disable the triple-*dupack* and timeout when the link impairment or user mobility contributes the packet loss, by reusing some existing TCP flow control mechanisms, *e.g.*, persistent mode [11], or by introducing some new explicit signaling schemes [12]. The sender also can behave more friendly [26] to packet losses, particularly when it only has a small number of in-flight packets.

There are also some proposals [9] in the network layer to eliminate the packet loss due to the user mobility, *e.g.*, hierarchical registering, controlled multicast, and packet forwarding. The ability of these higher layer proposals to cooperate with the lower layer flow and error controls which are mandatory in most cellular systems is still an open question and deserves further investigations. An even braver approach begins to propose some brand new protocols [7] or protocol stacks [6] especially for the networking scenario where TCP meets its built-in imperfection. They either cover the wireless portion based on split connection, or replace the whole end-to-end connection, and some even eliminate the need of the IP mobility support. However, the horizontal interface and possible interference for these new approaches with the *de facto* TCP/IP protocols over the global Internet are still not very clear.

This paper differentiates itself from other related work in several respects. First, instead of proposing some new schemes and algorithms in different layers and facing the compatibility issues, it studies the interaction between the current TCP/IP algorithms and the local retransmission techniques which are both widely deployed already. Secondly, this paper systematically investigates the end-to-end TCP performance with different performance metrics which are significant for users, networks, and protocols, respectively, in several representative networking scenarios to evaluate the tradeoffs with local retransmissions. The performance observations are explained by the explored endpoint TCP behaviors, in terms of spurious timeout and *dupack*, which further reveal the implication due to local retransmissions. With the knowledge that spurious *dupack* is a major factor for the unnecessary end-to-end TCP retransmissions, potential performance improvement by integrating the recent D-SACK proposal can be proposed and evaluated with the *undo* feature in this context. Finally, this paper gives an intuitive analysis on the conservative TCP

timeout algorithm which supports the rare observation of spurious timeouts in this paper and some other work.

3. System model

3.1. Networking scenarios

The physical wireless/wireline interworking topology is illustrated in Figure 1(a), where cellular systems are backboned by the wired Internet. A **Mobile Host** (MH) communicates with its **Correspondent Host** (CH) via the serving **Base Station** (BS), **Foreign Agent** (FA), and **Home Agent** (HA). IETF Mobile IP (MIP) [13] is used in this context to provide macro mobility support over the Internet, while other recently proposed schemes, *e.g.*, Cellular IP [14] or HAWAII [15], can be used to support micro mobility within cellular systems which are interconnected through the **Gateway** (GW) to the global Internet. Two logical connection modes, shown in Figure 1(b), are of interest here. The first one is so-called *end-to-end* where the MH talks to its CH directly via one TCP connection over the triangle MIP routing architecture. The second one utilizes a *split connection* paradigm where the GW acts as the application proxy to the CH and the TCP connection from the MH is constrained within the cellular systems. The first mode has been in use for many years, while the second mode is getting more popular recently also for other concerns, *e.g.*, packet filtering, address translation, and content rendering. Therefore two representative networking scenarios are studied here: *a*) Wireless **Wide Area Network** (WAN) + Wired WAN; *b*) Wireless WAN + Wired LAN. A single wireless link, *i.e.*, a special case of the second scenario, is also simulated for comparison purposes.

The raw radio link between the MH and its serving BS in cellular systems is characterized as comparatively small bandwidth due to limited radio spectrum, large transmission delay due to heavy modulation and interleaving, and noticeable error rate due to unavoidable path loss, fading, multiple path, and shadowing. Some representative 2.5G and 3G cellular technologies are approximated by using a generic

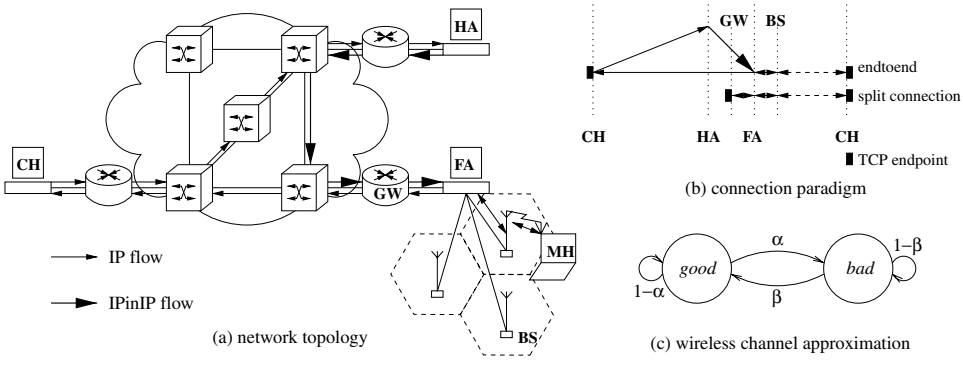

Figure 1. Networking scenarios.

two-state link model, depicted in Figure 1(c), with the configurable *good* or *bad* state bandwidth, delay, and error parameters. Usually the radio link suffers lower bandwidth, longer delay, and higher error rate in the *bad* state. The uplink and downlink bandwidth, delay, and error asymmetry are captured, and the overhead due to the lower layer encapsulation and segmentation is also considered. The residence time in each state is assumed exponentially distributed with a predefined mean value which can be calculated from the empirical measurement on real systems. Transmission errors in different states are assumed independent, while the error characteristics within a given state is highly correlated due to the residual channel memory.

Wired LAN has negligible transmission delay and congestion losses, while the wired WAN, particularly the Internet, has comparatively large delay and noticeable packet losses. Congestion losses are randomly distributed and only recoverable by the TCP endpoints. Therefore, in this study, there are mixed packet loss sources, *i.e.,* network congestion and transmission errors, and only the latter one can be recovered by local retransmissions.

3.2. TCP variants

Reno and NewReno are now two mainstream TCP variants implemented in various platforms, while SACK [16] and its further extensions, *e.g.*, D-SACK [17], are getting more and more attention. Since Tahoe, although an old TCP variant, reportedly has more stable behavior than Reno over wireless lossy links [26], it is also studied here.

All these TCP variants assume that all packet losses, either signaled by a timeout at the sender, a triple-*dupack* returned by receiver, or an **Explicit Congestion Notification** (ECN) [25] flagged by routers, are the result of network congestion, independent of what is the origin of a particular loss. Any TCP variant, initially or after being idle for a while, has to probe the network capacity by increasing its sending rate, first exponentially (known as **Slow Start**) and then linearly (**Congestion Avoidance**) after the **congestion window** (*cwnd*) reaches the **Slow Start threshold** (*ssthresh*). The load-gain curve in Figure 2(a) implies that the network will be eventually overloaded or congested with the aggressive TCP sender, and packets are discarded when buffer overflow occurs in routers. While retransmitting the lost packet, shown in Figure 2(c), the sender also exercises a family of coupled congestion algorithms [23] which differentiate these TCP variants. Tahoe treats a triple-*dupack* equivalently as a timeout (**Fast Retransmit**) and throttles the *cwnd* to its initial size, while Reno just optimistically halves *cwnd* (**Fast Recovery**, accurately a half of current outstanding data) on a triple-*dupack*, as Figure 2(b) illustrates.

The TCP receiver only acknowledges incoming packets accumulatively and the sender does not have any knowledge after the first lost packet indicated by a triple-*dupack*. Therefore usually neither Tahoe nor Reno has the ability to detect and recover multiple packet losses in the same sender window without waiting for a timeout. Figure 2(b) shows clearly that the timeout has much bigger impact than the triple-*dupack* on the achievable performance after packet losses, and the situation

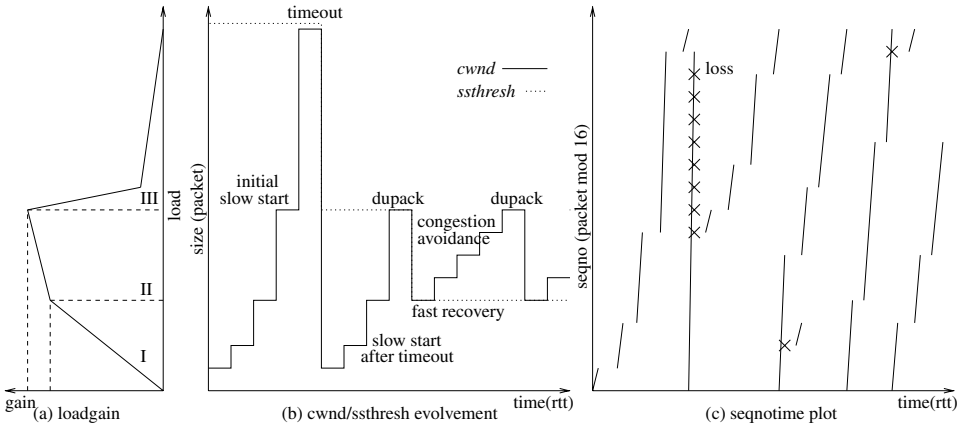

Figure 2. TCP congestion control.

is even worse when the packet loss indeed is not due to severe network congestion, *e.g.*, transmission errors over wireless links. To avoid the timeout, NewReno has an improvement based on the **Partial Acknowledgment** algorithm which can recover multiple losses in a "one per round-trip time (*rtt*)" manner, while SACK uses a more promising **Selective Acknowledgment** option, which requires both the TCP sender and receiver SACK capable, to recover multiple packet losses in a faster manner. However, it is found [26] that the ability of NewReno and SACK to recover multiple packet losses is still largely limited when there is a small amount of in-flight packets and the receiver can not generate enough feedback to trigger the expected algorithms.

3.3. Application traffic

Most Internet applications adopt the client/server model. In this model, the transport endpoint (*client*) first issues a small amount of requests to the other one (*server*), which in turn responds with a large amount of replies. It is further found that most user-initiated TCP connections mainly attract data towards the initiator. Given these observations, many emerging link technologies, *e.g.*, Asymmetric Digital Subscriber Line, Cable Modem, broadcast satellite, and some cellular systems, are also engineered on purpose to provide more bandwidth in the downlink to better fit the traffic asymmetry and to utilize the link resource efficiently. In this work, both download and upload applications are studied, but more attention is given to the asymmetric downloads since they currently dominate the Internet traffic workload. As shown in Figure 1(b), the MH is the user client that initiates a TCP connection to its CH in the end-to-end mode or to the GW in the split mode, sends a short request, and receives a large amount of reply data. Typical applications with this paradigm include the following: a mobile user using the handheld computer with cellular interfaces for Web surfing, file downloading, or database inquiry. Background traffic is also considered, and it competes with a TCP download flow that is under the main investigation in this paper. This traffic, generated

by other mobile users, is either TCP-transported, *e.g.*, long-lived FTP sessions, which is definitely TCP friendly, or UDP-transported unreliable **Constant Bit Rate** (CBR) streams without the built-in TCP friendliness.

4. Local retransmissions

The end-to-end TCP performance with local retransmissions is studied within the *ns*-2 [24] simulation framework. The original *ns-2* link object has been extended to accommodate the local retransmissions technique. Some important simulation parameters are listed in Table 1. A wired network has a comparatively high data rate, but might experience congestion losses over the global Internet. Data rates in wireless WAN correspond to those specified for *vehicle*, *pedestrian*, and *in-door* or *stationary* mobile hosts in the 2.5G and 3G cellular systems, and some even lower rates in the current systems are also simulated. Access delays include the propagation latency in the *macro*, *micro*, and *pico* cell configurations, and also include the time consumed by the lower layer signaling and processing. Uplink and downlink bandwidth ratio is assumed to be 1 : 4, as most commercial systems offer in such range, and the delay ratio is 2 : 1 due to the uplink MAC contentions and other overheads. The lower layer segmentation/reassembly overhead is assumed to consume about 15% of the raw capacity. The mean residence times in the *good* and *bad* states are approximated by the Doppler frequencies, which is a function of the mobile speed and the operating radio frequency. The raw average packet error rate varies from 0.1% to 1% for *light error* profile and from 1% to 10% for *heavy error* profile, and the error-free situation is also simulated for comparison purpose.

Every simulation run lasts 110 seconds, and each run with the same network and traffic settings repeats 20 times with various initial random seeds and background traffic patterns to eliminate the simulation dynamic. Background UDP traffic consumes roughly 25% of the link resource, and there are up to 8 concurrent TCP connections competing for the left resource with a TCP-transported flow of the main investigation. Logging facilities at the link object with local retransmissions and at the TCP source and sink objects are activated 5 seconds after the simulation starts and are deactivated 5 seconds before it finishes to eliminate the dynamic when the TCP connection is in its transit phase. Logged datasets are processed offline to extract the statistical mean of the desired metrics, *e.g.*, normalized throughput, effective goodput, and packet delay, which are detailed in the sequel.

Table 1. Some simulation parameters

	Data rate (Kbps)	Delay (ms)	Raw packet loss/error prob. (%)		Mean residence time (ms)
Wired WAN	1K	10	$1 \sim 2$		—
Wired LAN	10K	0.01	0		—
Wireless	144	100	good	0.1, 1	10, 50, 100
WAN	386	50	bad	$0.1 \sim 10$	1, 5, 10
downlink	1920	10	$1 \sim 100$		

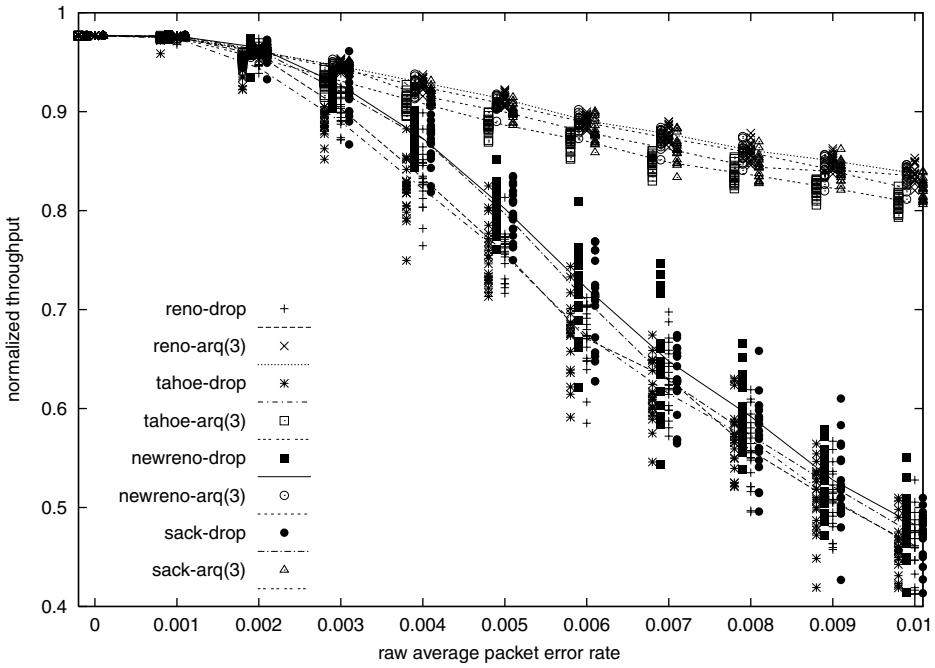

Figure 3. Normalized throughput for various TCP variants.

4.1. Normalized throughput

Normalized throughput in this context is defined by the ratio, in the range of 0 and 1, of the application throughput and the effective bandwidth available to that application. Here the effective bandwidth excludes the transmission error and the transmission capacity occupied by other traffic. Therefore the normalized throughput is independent of the background traffic. This is a very important user-oriented performance metric, as it reflects how much a user can obtain from what the network provider offers by using the TCP/IP protocol stack.

Figure 3 gives a throughput comparison of different TCP variants with and without local retransmissions, when the wireless delay is 10 ms and the mean *bad* state residence time is 1 ms. Other settings have similar results unless otherwise stated. Measured samples are depicted in dots and their mean values by curves. Samples are scattered in the horizontal axis for presentation clarity. With local retransmissions (*arq*) where the default retry limit is 3, the effective packet loss rate is 0 when the raw packet error rate ranges from 0.1% to 1%. Almost no timeout, even with a fine grained timer of 100 ms, at the TCP sender is observed. Therefore Tahoe achieves the lowest throughput as it treats a *dupack* the same as a timeout and re-initializes its *cwnd*. Reno and NewReno have a similar performance. Since SACK is more aggressive to recover the packets that are actually still retried by local retransmissions, it has lower normalized throughput than NewReno. Without local retransmission (*drop*), TCP experiences pseudo-random packet losses. Therefore, NewReno

and SACK achieve better throughput than Tahoe and Reno as they have the ability to recover multiple packet losses in one sender window. Since Tahoe conservatively follows the Slow Start algorithm after the first packet loss, it is reportedly more robust than Reno, which first follows Congestion Avoidance on a triple-*dupack* and then follows Slow Start on a timeout, if there are consecutive losses due to the burst transmission errors over wireless links. Figure 3 supports such a claim when the error rate is comparatively high. In what follows the main focus is given to the cases with and without local retransmissions, and Reno is chosen as the representative TCP variant for presentation simplicity.

With local retransmissions, it is shown that TCP can gain higher normalized throughput. However, when the wireless delay and bandwidth grows, the gain becomes smaller, although still positive, since local retransmissions become costly (Figure 4) and even local retransmissions cannot fully compensate for the capacity loss in a high speed link. Also when there are mixed packet loss sources in the end-to-end mode, the gain due to local retransmissions becomes smaller as a major component of packet losses is only recoverable by the end-to-end TCP retransmissions.

4.2. Effective goodput

Local retransmissions cannot achieve the positive gain in normalized throughput without any tradeoff. Here the effective goodput and packet delay are considered

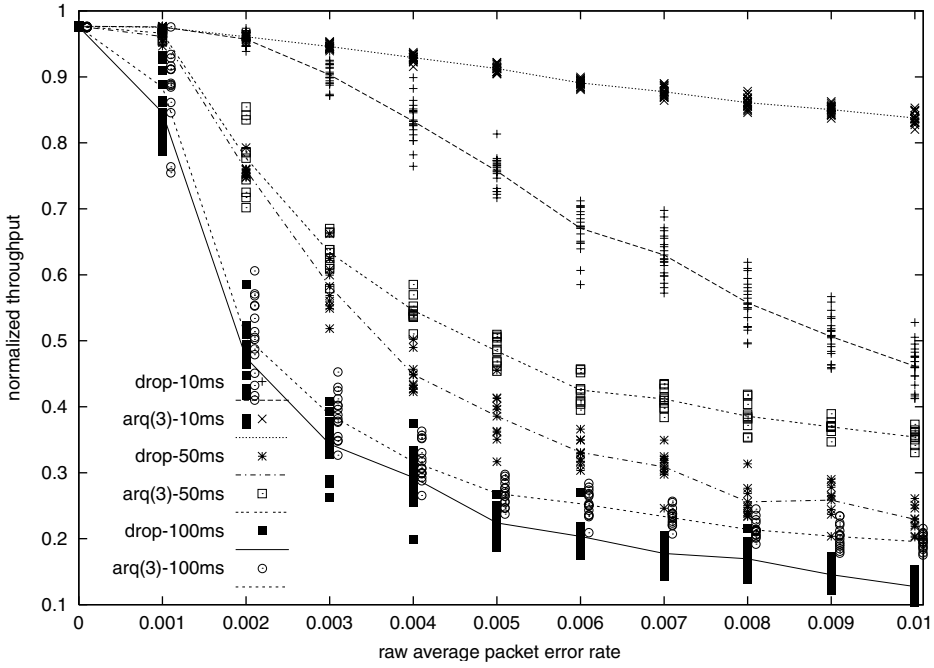

Figure 4. Normalized throughput with various wireless delays.

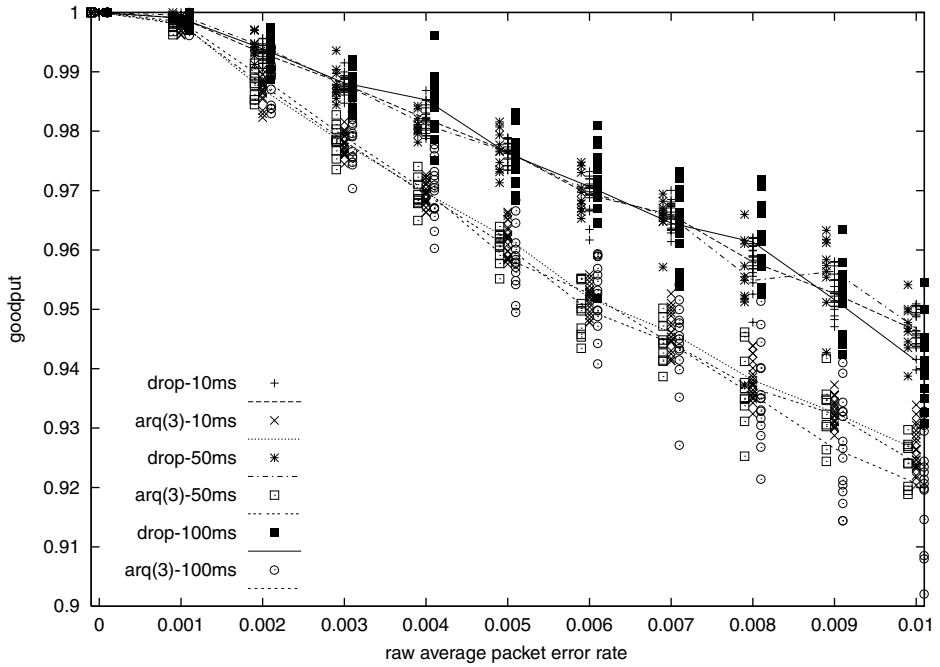

Figure 5. End-to-end goodput with various wireless delays.

as another two important performance metrics. Effective goodput in this context is defined by the ratio, in the range of 0 and 1, of the amount of data transferred from the TCP sender to the receiver and the amount of total traffic generated in the same direction. Both end-to-end and local retransmissions can generate additional overhead traffic. This metric is more significant in cellular systems, as it is proportional to the efficiency of the energy consumed by mobile hosts which are usually powered by batteries. It is also proportional to the effective utilization of the limited radio resource which is shared by a group of mobile hosts.

Figure 5 shows the effective TCP goodput with and without local retransmissions, with various wireless delay settings. It is interesting to find that the goodput is almost independent of the different wireless delay and bandwidth settings unless the error rate is very high, as it is invariant for the given end-to-end TCP algorithms and local link mechanisms. The case with local retransmissions has lower goodput since more traffic is generated locally over the wireless link. For examples, when the raw average error rate is 1% and the wireless bandwidth and delay are 384 Kbps and 10 ms, respectively, the goodput is about 2% lower for the case with local retransmissions whose throughput is about 37.6% higher. Therefore, in this case, the increased overhead due to local retransmissions is tolerable. However, when the wireless delay grows to 100 ms, a similar loss of 2.1% in goodput only brings the gain about 6.8% in throughput with local retransmissions. Therefore, the increased overhead due to local retransmissions in this case becomes non-negligible and needs to be balanced along with other concerns. Similar tradeoffs are also found when the wireless bandwidth grows and when there are congestion losses in the wired WAN.

4.3. Packet delay

Packet delay in this context is defined as the time experienced by a particular data packet from the TCP sender to the receiver, including the delay due to the transmission, propagation, queueing, and local retransmission if any. Although the throughput is a primary metric for bulk data transfer applications, the packet delay is also measured in this experiment to quantify the impact due to local retransmissions. Since TCP uses the timeout mechanism as the final guard to detect and recover packet losses, a change in packet delay will have impact on the efficiency of the timeout estimation algorithm. Also, this metric can provide a guideline for the non-TCP traffic which is delay and jitter sensitive but exercises the TCP-friendly flow control mechanisms.

Figure 6 shows the average packet delay with and without local retransmissions, with various wireless delay settings, when the wireless bandwidth is 384 Kbps. When the raw error rate is low, the smaller the wireless delay, the quicker the TCP sender increases its *cwnd,* and the more packets queued at the wireline/wireless interface since the wireless link is the bottleneck in the packet forwarding path. Therefore, packets experience large delay with a small wireless delay, since in this case the queueing delay dominates the packet transit delay. When the error rate goes high, the queue length at the wireline/wireless interface decreases dramatically, and the transmission delay over the wireless link begins to dominate the packet delay. Therefore, packets experience large delay with a large wireless delay when the wireless error rate is high. Average packet delay increases for the case with local retransmissions, since a packet or a portion of the packet has to travel the wireless link multiple times if there is an error in the previous journey. However, as also shown in Figure 6, the maximum packet delay, which characterizes the worst case, increases dramatically with the large wireless delay when the error rate grows. When the wireless delay is 100 ms and the raw average error rate is 1%, the maximum packet delay can increase about 4 times with local retransmissions, while the throughput gain is only about 6.8% as mentioned. Therefore the tradeoff between the gain in throughput and the losses in effective goodput and packet delay deserves more consideration in certain networking scenarios.

5. Endpoint behaviors

The end-to-end TCP performance observed in the previous section is a result of the internal protocol behaviors at the TCP endpoints. ECN [25] is a proactive approach for the TCP sender possibly to detect the incoming congestion even before any packet loss occurs. However, ECN requires the participation of the intermediate router with the active queue management which is not widely deployed yet. Therefore this paper still focuses on the traditional timeout and triple-*dupack* behaviors at the TCP sender. With local retransmissions, packets experience changing delay and get re-ordered when arriving at the receiver, which might trigger spurious timeout and *dupack*.

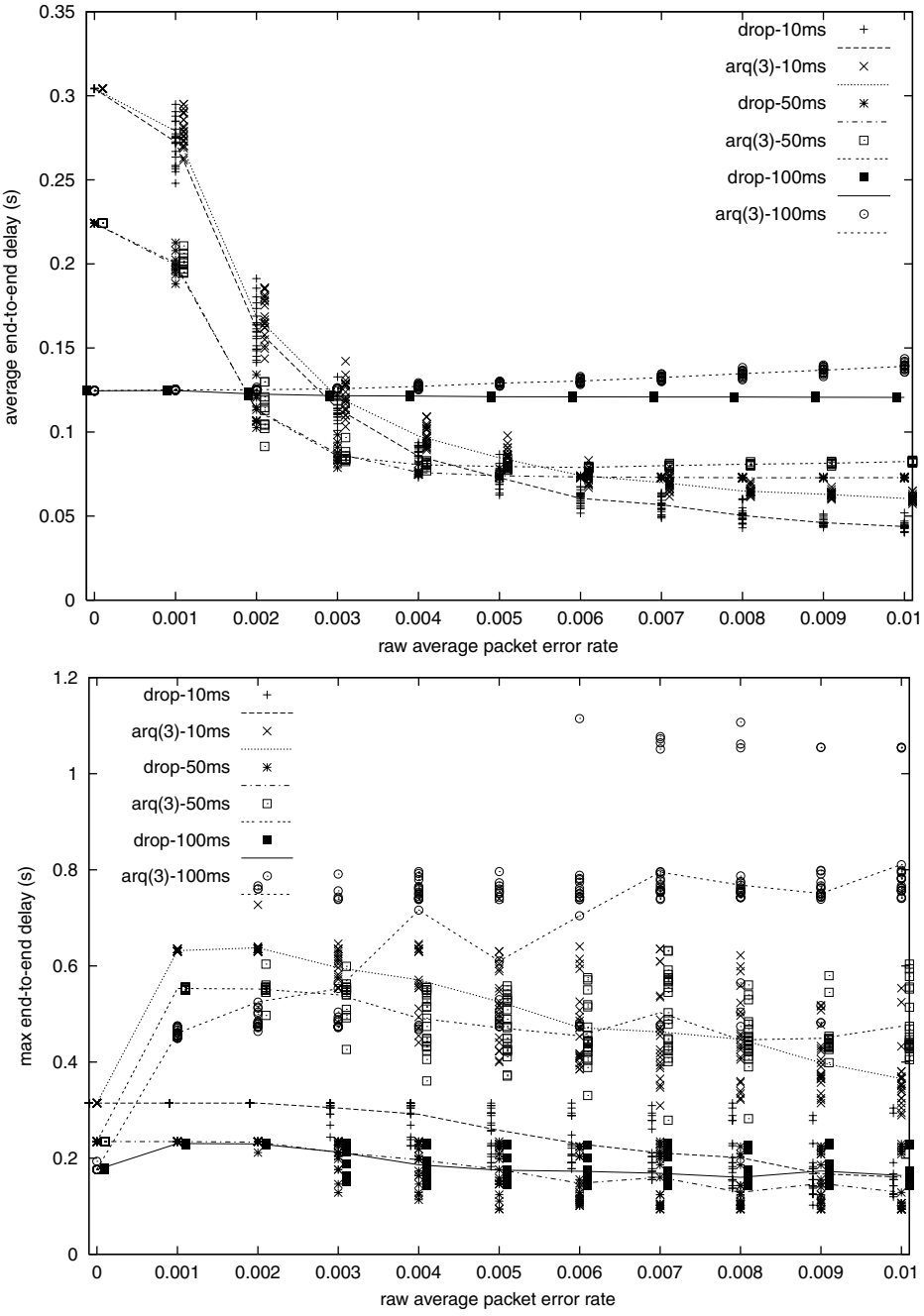

Figure 6. Average and maximum packet delay with various wireless delays.

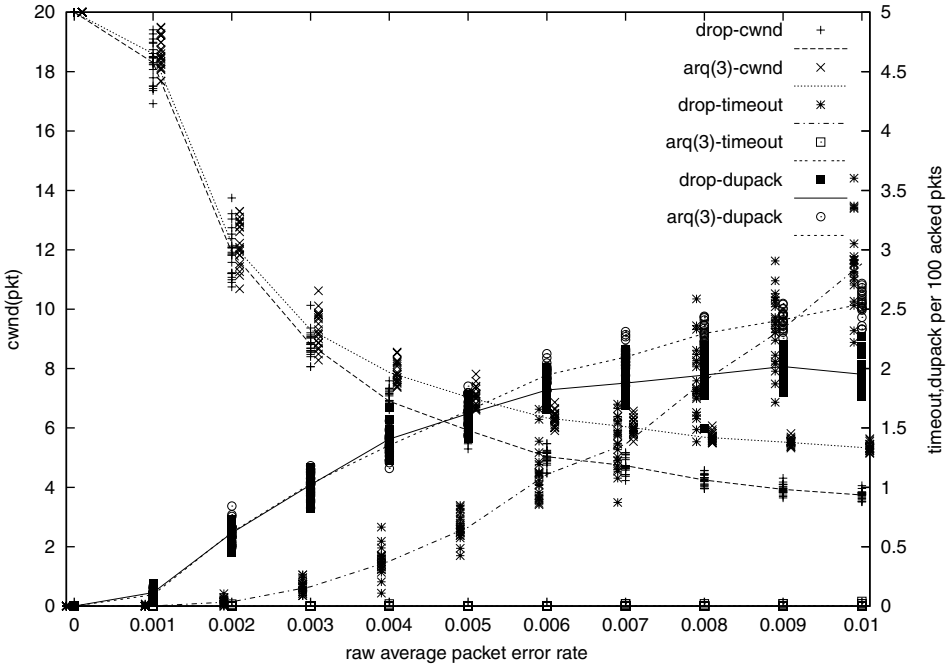

Figure 7. Reno *cwnd*, *timeout*, and *dupack*.

Figure 7 illustrates the *cwnd* size and the number of timeouts and *dupack*s that trigger the end-to-end TCP retransmissions in Figure 3. The *cwnd* size is logged roughly every *rtt* during the simulation and is averaged when it finishes. The numbers of timeout and *dupack* triggers are counted and normalized for every 100 acknowledged packets. Even the case with local retransmissions has slightly more *dupack* triggered TCP retransmissions; the case without local retransmissions has non-negligible timeout triggered ones. Since the timeout takes much more time to recover than the *dupack* and decreases *cwnd* severely as Figure 2(b) shows, it explains why *cwnd* is larger with local retransmissions when the raw error rate goes high. A large *cwnd* implies a higher achievable throughput in Figure 3.

5.1. Spurious timeout

However, the TCP behaviors explored here do not fully conform to the results reported in the literature [18]. Some observe that local retransmissions can bring many sender timeouts and competitive retransmissions in both transport and link layers, which is only found here with a very fine timer granularity.

The symbols used in this subsection are tabulated in Table 2. The TCP timeout mechanism uses an exponentially weighted moving average (**EWMA**) algorithm, which is outlined in this recursive expressions: $rto_i = srtt_i + g_3 rttv_i$, where $srtt_i = srtt_{i-1} + g_1 err_i$, $rttv_i = rttv_{i-1} + g_2(|err_i| - rttv_{i-1})$, and $err_i = rtt_i - srtt_{i-1}$. The parameter *rtt* is measured once in each sender window and *rto* is calculated accordingly.

Table 2. TCP timeout algorithm parameters

rtt_i	round-trip time measurement	$srtt_i$	smoothed rtt_i
$rttv_i$	rtt_i variance	rto_i	round-trip timeout value
b	burst factor	$g_{1,2,3}$	EWMA weight parameters

By comparing the networking scenarios and simulation settings, it is found that many previous studies use an old TCP Tahoe variant in their analysis and experiments. Originally TCP Tahoe adopts $g_3 = 2$. Although this variant is still found in many operating systems, modern TCP has already adopted $g_3 = 4$ for a more conservative *rto*. The following is an intuitive analysis that explains the conflicting results observed in other research work.

Assume in the worst case that $rtt_i = t$ for $0 < i < k$ and $rtt_k = b * t$ with a large k and a finest timer granularity, where t is the smallest *rtt* and b is the burst factor. Therefore it is a good approximation that $srtt_{k-1} = t, rttv_{k-1} = 0$, and $rto_{k-1} = t$. Since $rto_{k-1} < rtt_k$ when $b > 1$, there is an unavoidable timeout at k. TCP exponentially backs off when a timeout occurs and discards that *rtt* sample during the retransmission. If there is a valid *rtt* available after the retransmission, given $err_k = (b-1)t, rto$ is computed by the following algorithm,

$$srtt_k = t + g_1(b-1)t,$$
$$rttv_k = g_2(b-1)t,$$
$$rto_k = t + g_1(b-1)t + g_3g_2(b-1)t.$$

To avoid possible consecutive timeouts, rto_k has to be conservatively larger than rtt_{k+1}, which, in the worst case, is another burst of size $b * t$. Therefore to satisfy

$$t + g_1(b-1)t + g_3g_2(b-1)t > b * t,$$

it is obvious that

$$g_1 + g_2g_3 > 1.$$

Since in TCP $g_1 = \frac{1}{8}$ and $g_2 = \frac{1}{4}$, then $g_3 > \frac{7}{2}$ should be a safety requirement to ensure a conservative *rto* algorithm. Figure 8 gives a numerical illustration on *rtt* and *rto* evolutions, where $t = 100$ ms, $b = 10$, and $srtt_{\text{init}} = 2rttv_{\text{init}} = 1$ sec, with the default TCP weight parameters. If $g_3 = 2$ as the old Tahoe does, it might be too aggressive and results in many spurious timeouts as observed before. Balancing the fact that a too conservative *rto* is deficient to detect lost packets, $g_3 = 2^2$ is a good choice also for easy bit shifting implementation.

Figure 9 supports the above intuitive analysis by illustrating the number of timeouts with different *rttv* weights, along with the case without having *rttv* in the *rto* equation for comparison purpose. The vertical axis is in logarithmic scale for a better illustration. Without *rttv*, the calculated *rto* is very aggressive and does not tolerate any change in *rtt*, so it is quite sensitive to packet losses and local retransmissions. The resultant spurious timeouts are almost comparable to the case without

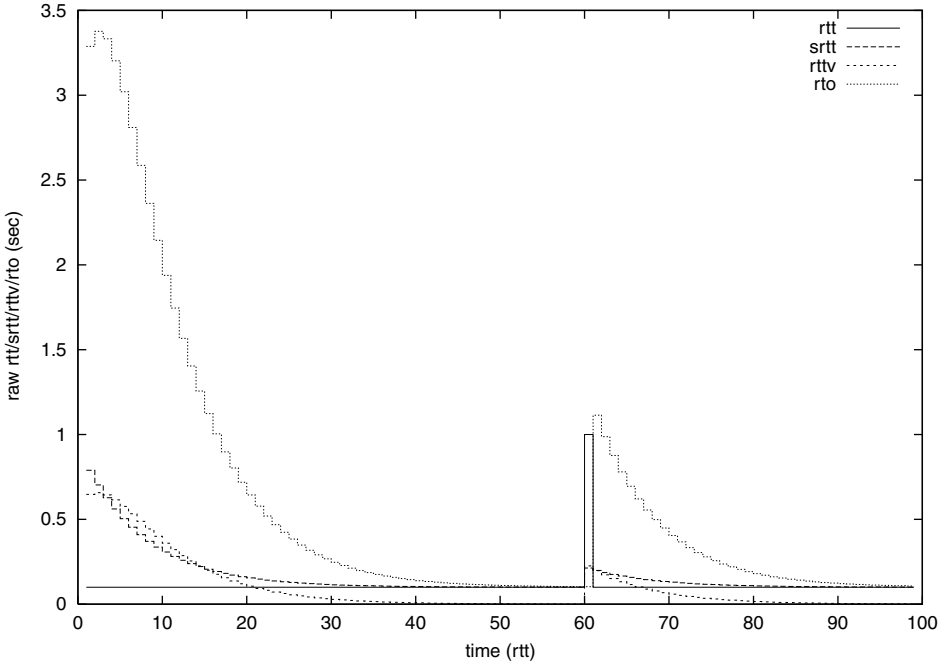

Figure 8. TCP *rto* calculation.

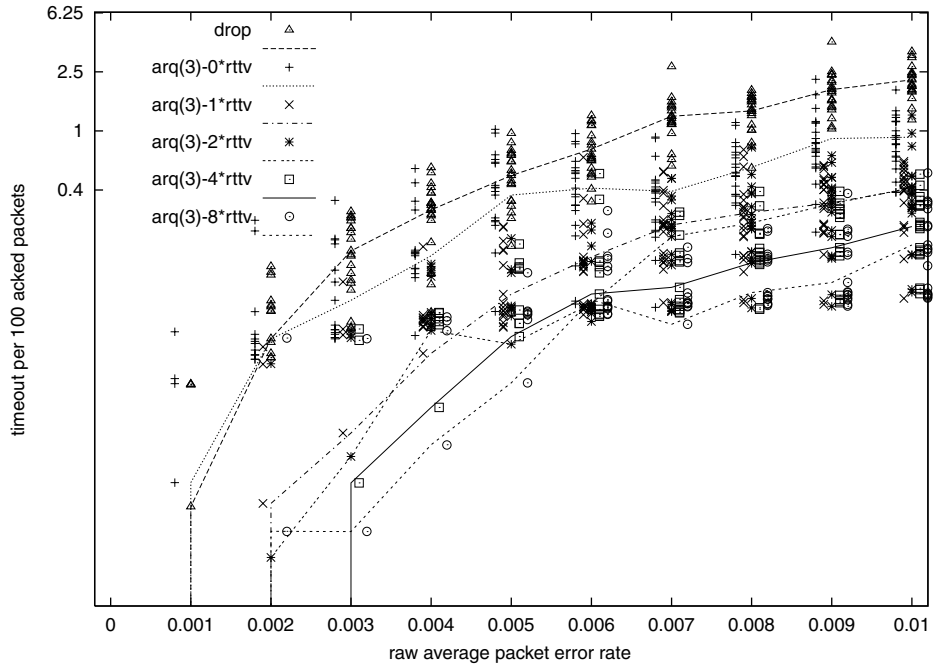

Figure 9. Timeout with various *rttv* weights.

local retransmissions. With $g_3 = 1$ or 2, the calculated *rto* is still aggressive and introduces almost twice as many timeouts as the case with $g_3 = 4$ when the raw packet error rate grows. However, $g_3 = 8$ does not have too much difference from the current default g_3, and a too conservative *rto* is deficient to detect lost packets. This figure shows $g_3 = 4$ is a better choice than is either 2 or 8.

5.2. *Spurious dupack*

Figure 7 shows that *dupack*, instead of timeout, is a major factor for the unnecessary end-to-end retransmissions in the case with local retransmissions. Figure 10 further illustrates the number of *dupack*-triggered end-to-end retransmissions with various wireless delay settings, with the same networking configuration as used to obtain the results shown in Figure 4. With a large wireless delay, the selective acknowledgment returns much later to trigger local retransmissions. Since more follow-on packets are already in flight, a large number of packets arrive at the receiver out-of-order. Therefore more *dupack*s due to packet reordering are generated at the receiver and returned to the sender, and spurious end-to-end retransmissions are triggered. It is found that the larger the wireless delay, the more sensitive are spurious *dupack*s to the increase of raw packet error rates. This explains the performance observed in the Figure 4, when the wireless delay increases, the normalized throughput and the performance gain over the error dropping policy decrease dramatically. Similar

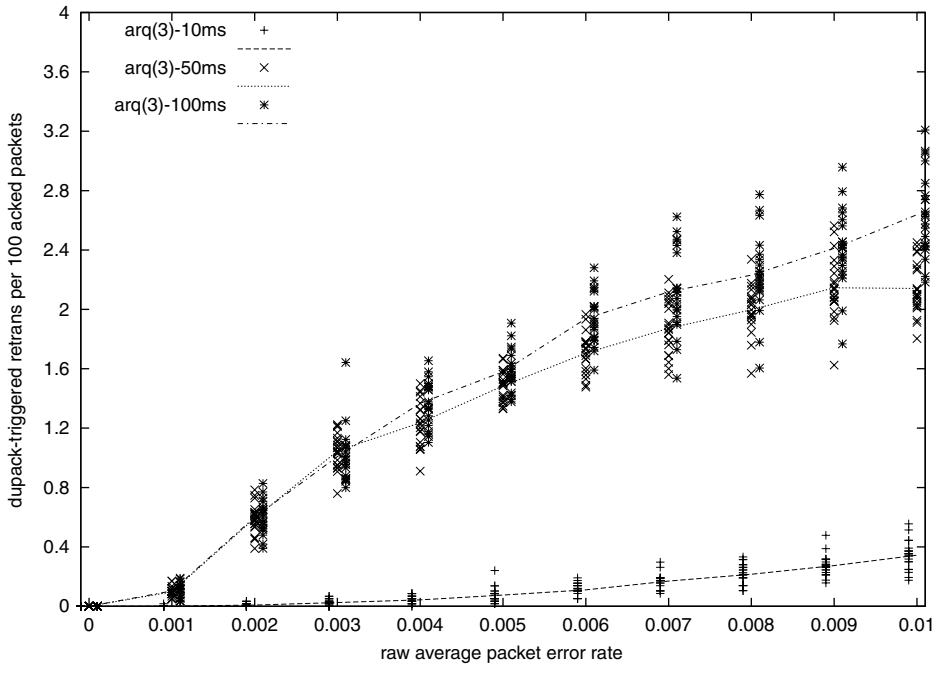

Figure 10. *Dupack*s with various wireless delays.

spurious *dupack* returned from the TCP receiver also explains the performance difference with different wireless bandwidth and residence time settings.

Since spurious *dupack*s contribute most of the unnecessary end-to-end retransmissions, it is obvious that, by avoiding these *dupack*-triggered spurious retransmissions, the performance can be further improved. There are several approaches available with the existing mechanisms. Increasing the *dupack* threshold beyond the default value of 3 is a possibility. But it is very difficult, if not impossible, to choose a suitable threshold for an arbitrary networking scenario, since the TCP sender does not know local retransmissions in remote links. Furthermore, an increased threshold without any adaptation reduces the sender's ability to detect and recover the packet that is already lost. On the other hand, the TCP receiver can delay the third *dupack* [27] and give local retransmissions more chances. Again, finding a suitable *dupack* delay interval is not easier than finding a dynamic *dupack* threshold, since the receiver does not have enough knowledge on how long it can wait without triggering a timeout at the TCP sender.

5.3. D-SACK improvement

SACK [16] was originally proposed to extend the accumulative acknowledgment by using the SACK options which acknowledge the non-contiguous but received packets. With the header length limit, an acknowledgment packet (*ack*) can contain up to 3 pairs of sequence numbers. The first pair outlines the block which contains the packet that triggers this *ack,* and the following two just redundantly repeat the first two SACK pairs in the previous *ack*. This design maintains the robustness if the previous *ack* is lost. The SACK-capable TCP sender then has the ability to determine which packets are missing and which are received at the receiver after the contiguous block. Therefore, the sender can detect lost packets if they are reportedly missing several times and recover them in a more efficient manner by not retransmitting the packets which are selectively acknowledged, unless a timeout occurs.

D-SACK [17] is recently proposed to further extend the SACK mechanism to report the duplicated packets. A packet might get duplicated by the network, or unnecessarily retransmitted by the sender due to *ack* loss, premature timeout, or packet reordering. The receiver uses the first SACK pair to report a full or partially duplicated packet. By comparing the first SACK pair to the accumulative *ack* in the original TCP header, the sender can distinguish whether it is a SACK or D-SACK. If a packet is received in duplicate without any sender transmission, it indicates that probably the network duplicates the packet or its *ack*. Otherwise, it indicates that, a moment ago, there was an unnecessary retransmission triggered by either timeout or *dupack*. If this packet has not been acknowledged accumulatively yet, there might be an *ack* loss; otherwise, the timeout might be too aggressive and occurs prematurely. If there was a *dupack*-triggered retransmission, it might be caused by packet reordering instead of pack losses.

With the D-SACK option returned by the receiver and the history that it maintains, the sender subsequently has the ability to figure out whether a previous *dupack*

is caused by a duplicated packet, a duplicated *ack*, or a gap in the receiver's buffer. If the *dupack* that triggers a retransmission is due to the duplicated packets, it is possible for the sender to *undo* the unnecessary congestion regulation which decreases *cwnd* size. Although the unnecessary retransmission has already wasted certain network resources for a while, its impact in the follow-on periods can be hopefully minimized if *cwnd* and *ssthresh* can be recovered to a proper size.

Since the spurious *dupack* plays a major role in the TCP endpoint when there is local retransmissions, it is expected that by integrating the D-SACK, these unnecessary end-to-end retransmission triggered by spurious *dupack* can be largely avoided. Figure 11 shows the end-to-end TCP performance in terms of normalized throughput (the left vertical axis) and the TCP endpoint behavior in terms of total end-to-end retransmissions (the right vertical axis) with or without the D-SACK option. It uses the worst case parameters shown in Figure 4, *i.e.*, with the longest wireless delay and the finest timer granularity. With local retransmissions, the normalized throughput is increased slightly over the *error-dropping* policy since there are still many spurious end-to-end retransmissions triggered by *dupack*s due to packet reordering. With both the D-SACK option and local retransmissions, the normalized throughput is increased dramatically as most impact due to spurious *dupack*s have been eliminated and the spurious timeouts only have very little performance impact as shown in the previous subsections.

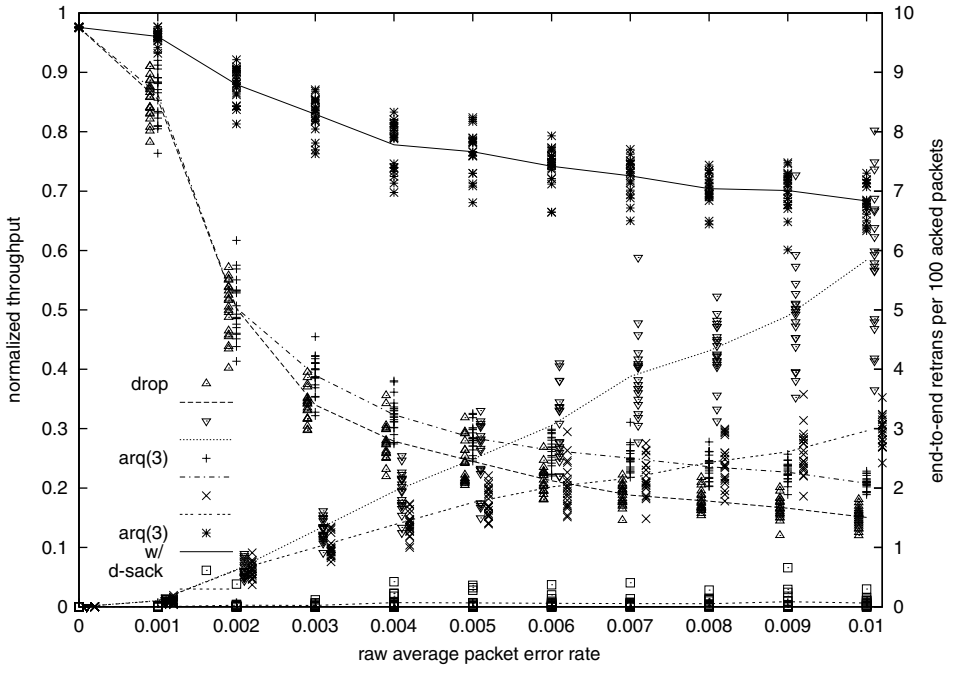

Figure 11. Performance and behaviors with D-SACK.

6. Conclusions

The end-to-end TCP performance with local retransmissions in several representative networking scenarios is assessed in this paper. The performance metrics, *i.e.*, normalized throughput, effective goodput, and packet delay, are significant to users, networks, and protocols, respectively. These metrics show the tradeoff nature with local retransmissions in different wireless bandwidth, delay, and error settings. The observed performance is further explained by the explored TCP endpoint behaviors in terms of spurious timeout and *dupack*. Intuitive analysis on the conservative TCP timeout algorithm supports the observation of the rare spurious timeout with local retransmissions, and the spurious *dupack* is found to have the major implication of the unnecessary end-to-end TCP retransmission. By integrating the recent D-SACK proposal which can distinguish the source of *dupack* alone with the sending history, the TCP performance with local retransmissions is evaluated with further improvement in the same context.

Acknowledgments

This work has been supported in part by a grant from the Canadian Institute for Telecommunications Research (CITR) under the NCE program of the Government of Canada, and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No. RGPIN7779. Views and opinions in this paper are those of the authors.

References

1. A. K. Salkintzis. Packet data over cellular networks: the CDPD approach. *IEEE Communications Magazine*, 37(6):152–159, 1999.
2. C. Scholefield. Evolving GSM data services. *Proceedings of IEEE ICUPC*'97, 2:888–892, 1997.
3. P. Bender, P. Black, M. Grob, et al. CDMA/HDR: a bandwidth efficient high speed wireless data service for nomadic users. *IEEE Communications Magazine*, 38(7):70–77, 2000.
4. A. Furuskar, S. Mazur, F. Muller, and H. Olofsson. EDGE: enhanced data rates for GSM and TDMA/136 evolution. *IEEE Personal Communications*, 6(3):56–66, 1999.
5. H. Inamura, ed. TCP over 2.5G and 3G wireless networks. Internet-Draft, draft-ietf-pilc-2.5g3g-03, 2001.
6. WAP Forum. Wireless application protocol. http://www.wapforum.org, 2001.
7. Z. J. Haas. Mobile-TCP: an asymmetric transport protocol design for mobile systems. *Proceedings of IEEE ICC'97*, pp. 1054–1058, 1997.
8. M. Allman, D. Glover, and L. Sanchez. Enhancing TCP over satellite channels using standard mechanisms. *IETF RFC 2488*, 1999.
9. J. Mark, X. Shen, Y. Zeng, and J. Pan. TCP/IP over wireless/wireline networks. *Proceedings of IEEE 3Gwireless'2000*, 438–445, 2000. see also TCP/IP over air links, http://bbcr.uwaterloo.ca/∼ jpan/tcpair, 2001.
10. H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.
11. A. Chan, D. H. K. Tsang, and S. Gupta. Impact of handoff on TCP performance in mobile wireless computing. *Proceedings of IEEE ICPWC'97*, pp. 184–188, 1997.

12. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving performance of TCP over wireless networks. *Proceedings of IEEE 17th ICDCS'97*, pp. 365–373, 1997.
13. C. E. Perkins. Mobile networking through Mobile IP. *IEEE Internet Computing*. 2(1):58–69, 1998.
14. A. G. Valko. Cellular IP—a new approach to Internet host mobility. *ACM Comp. Comm. Review*, 29(1):50–65, 1999.
15. R. Ramjee, T. F. La Porta, L. Salgarelli, S. Thuel, K. Varadhan, and L. Li. IP-based access network infrastructure for next-generation wireless data networks. *IEEE Personal Comm.*, 7(4):34–41, 2000.
16. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment option. IETF RFC 2018, 1996.
17. S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An extension to the selective acknowledgement (SACK) option for TCP. IETF RFC 2883, 2000.
18. A. DeSmimone, M. Clmah, and O. Yue. Throughput performance of transport layer protocols over wireless LANs. *Proceedings of IEEE GlobeCom'93*, pp. 542–549, 1993.
19. D. A. Eckhardt and P. Steenkiste. Improving wireless LAN performance via adaptive local error control. *Proceedings of IEEE ICNP'98*, pp. 327–338, 1998.
20. R. Ludwig and B. Rathonyi. Link layer enhancements for TCP/IP over GSM. *Proceedings of IEEE INFOCOM '99*, 2:415–420, 1999.
21. J. S. A. Liew, C. W. Ang, and K. F. Ho. Improving TCP performance over multi-slot GSM. *Proceedings of IEEE ICC'99*, 1:329–333, 1999.
22. H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. *Proceedings of ACM MobiCom'95*, pp. 2–11, 1995.
23. M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. IETF RFC 2581, 1999.
24. Vint. The network simulator—ns-2. http://www.isi.edu/nsnam/ns, 2001.
25. S. Floyd. TCP and explicit congestion notification. *ACM Comp. Comm. Review*, 24(5):10–23, 1994.
26. J. Pan, J. W. Mark, and X. Shen. TCP performance and its improvement over lossy wireless links. *Proceedings of IEEE Globe-Com'2000*, pp. 62–66, 2000.
27. N. H. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed duplicated acknowledgments: a TCP-unaware approach to improve performance of TCP over wireless. Texas A&M University, TR-99-003, 1999.