Analysis, Simulation, and Implementation of VCP: A Wireless Profiling

Xiaolong Li, Student Member IEEE Homayoun Yousefi'zadeh, Senior Member IEEE

***Abstract*—Every congestion control protocol operating in wireless networks is potentially faced with two major challenges of performance degradation. These sources are (a) the coupling of fairness and efficiency, and (b) not properly differentiating between congestion-caused loss associated with network buffering and error-caused loss associated with fading effects. In this paper, we provide a VCP-based cross-layer framework of congestion control that can address both challenges noted above. As a part of our framework, we introduce a loss differentiation heuristic algorithm that can be used with a variety of congestion control protocols. Then using analysis, simulation, implementation, and emulation, we profile the performance of a number of congestion control alternatives in wireless networks. We describe the first implementation of VCP as a collection of loadable kernel modules along with fine-tuned implementations of XCP and TCP/AQM+ECN in Linux. We utilize NS2 as our simulation tool and a wired Linux testbed emulating wireless link effects as our experimental tool. We implement a finite-state Markov chain in both NS2 and our testbed in order to model error-caused loss over wireless links. We further use link layer FEC codes on a per packet basis to compensate against such loss. Our profiling results demonstrate that VCP equipped with our loss differentiation heuristic and link layer FEC represents a well-performing yet practical alternative of wireless congestion control. We also identify some of the shortcomings of VCP including its oscillatory behavior in the presence of link estimation errors and poor fairness characteristic in multi-bottleneck networks.**

***Index Terms*—Congestion Control, Wireless Networks, Fading Links, Markov chain, TCP, AQM, RED, REM, ECN, XCP, VCP.**

I. Introduction

As surveyed by [1], the coupling of fairness and efficiency is a known source of performance degradation for TCP and end-to-end TCP-based Active Queue Management (AQM) schemes [2], [3], [4], [5], [6]. The problem is further pronounced in networks including high Bandwidth-Delay Product (BDP) links such as wireless and satellite networks. In the past few years, a wide variety of techniques have emerged to increase the efficiency of congestion control protocols in high BDP networks. The works of [7], [8], [9] adaptively adjust the sending window size of TCP by amending the parameters of Additive-Increase Multiplicative-Decrease (AIMD) [10], while the works of [11], [12], [13], [14] use different congestion signals and the works of [15], [16] explicitly signal the congestion information to the sender. Since all of the proposed works above retain an integrated controller design, they often fail to achieve both efficiency and fairness.

In contrast, recently proposed eXplicit Congestion-control Protocol (XCP) [17] and Variable-structure Congestion-control Protocol (VCP) [18] attempt at decoupling fairness from efficiency in order to address performance degradation of TCP. While XCP uses Multiplicative-Increase Multiplicative-Decrease (MIMD) for efficiency and AIMD for fairness control, VCP applies Multiplicative-Increase, Additive-Increase, and Multiplicative-Decrease (MIAIMD) policies in three regions of congestion known as low-load, high-load, overload regions, respectively. By encapsulating congestion related information into packet headers, both protocols exhibit high utilization and great fairness characteristics while maintaining low persistent queue lengths and reducing congestion-caused loss in wired networks. However, XCP calls for using multiple bits in a packet header introducing significant deployment obstacles. To the contrary, VCP is able to achieve a performance comparable to XCP using only two ECN bits while keeping compatibility with a variety of existing protocols.

Transmission over wireless links is subject to both error- and congestion-caused loss. Thus, the performance of any protocol utilizing IP header bits to carry congestion information may be seriously crippled in wireless networks without proper compensation against error-caused loss. While a spectrum of research works as reviewed by [19] have studied congestion control in wireless networks, they often focus on capturing the effects of data link layer in the performance of congestion control. For example, [20] shows that retransmission at the link layer may result in delivering feedback that is misleading to TCP.

In this paper, we propose a cross-layer framework to study the performance of TCP/AQM+ECN, XCP, and VCP as viable end-to-end alternatives of congestion control in wireless networks. First, we apply our framework to compare the performance of TCP/AQM+ECN, XCP, and VCP over simulated wireless and satellite links. Second, we conduct experimental studies performed over a wired Linux testbed emulating wireless link effects and utilizing our Linux implementation of VCP along with fine-tuned implementations of other protocols.

Specifically, the key contributions of this paper include the followings.

- *The first implementation of VCP:* To the best of our knowledge, this paper reports the first implementation of VCP in the Linux kernel. The implementation consists of a number of Loadable Kernel Modules (LKMs) associated with transmitting, intermediate, and receiving

Parts of this paper appeared in the Proc. of IEEE WCNC'06 and IEEE MILCOM'07. The authors are with the Department of EECS, University of California, Irvine. e-mail: ([xiaolonl,hyousefi]@uci.edu). This work was sponsored by grant no. COM06-10223 from the Boeing Company and UC Discovery Industry-University Cooperative Research Program.

nodes of VCP. The implementation allows for the co-existence of VCP with TCP and UDP. The implementation approach is also generalized to provide a transparent way of implementing other congestion control protocols in Linux.

- *Implementation of Gilbert-Elliott (GE) error model:* Relying on the Markov chain modeling of a wireless link, the paper develops loss models matching the observed loss pattern of wireless links. It also demonstrates how the use of FEC can mitigate some of the error-caused loss effects. It implements the two-state GE Markov chain modeling the error pattern of a wireless link in the Linux kernel. The use of GE model allows for realistically emulating wireless link effects caused by fading as well as other parameters of the physical and link layers.
- *A cross-layer framework:* The paper provides a cross-layer framework of analysis, modeling, simulation, and emulation of congestion control in wireless networks. The framework integrates the topics noted above, along with a heuristic algorithm that allows for differentiating error- and congestion-caused loss in wireless networks.
- *Performance evaluation of congestion control protocols:* The paper performs extensive simulation and experimental studies contrasting VCP versus other alternatives of congestion control. It demonstrates (i) the need for protecting protocols' metadata as well as data against bit errors, and (ii) that combined with our loss differentiation heuristic, VCP represents a high performing yet practical congestion control protocol for use in encrypted wireless networks.

The rest of the paper is organized as follows. Section II presents the fundamentals of VCP and related work. In Section III, we conduct simulation studies comparing the performance of a variety of congestion control protocols in our framework. In Section IV, we present some details of VCP operation and our implementation approach. Experimental studies with our VCP Linux implementation are presented in Section V. In Section VI, we discuss potential shortcomings of VCP and provide a review of related work. Finally, we present several conclusions and discuss our future work in Section VII.

II. Background and Related Work

In this section, we briefly introduce the fundamentals of VCP and summarize the related work.

A. Fundamentals of VCP

Fundamentally, VCP remains a window-based protocol and is designed to regulate the $cwnd$ with different congestion control policies according to the level of congestion in the network. VCP defines three levels of congestion: low-load, high-load, and overload allowing for encoding the level of congestion into two ECN bits in the IP packet header. Upon arrival of a packet, each VCP-capable router does: (i) compute the load factor of each of its links, (ii) map each computed load factor to one of the three congestion levels, and (iii) update the ECN bits in the packet header. A more congested downstream router can further change the level of congestion by overwriting it. Finally, the receiver signals the sender with the congestion information via acknowledgment (ACK) packets. Consequently, VCP applies three congestion control policies: MI in the low-load region, AI in the high-load region, and MD in the overload region. The MI region is utilized to eliminate the slow start characteristic of TCP while AI and MD regions preserve the fairness characteristics of TCP.

B. Related Work

A wide variety of approaches have been proposed to improve the performance of TCP in wireless and satellite networks. The majority of these approaches can be roughly grouped into three categories: (i) split connection protocols such as [21], [22], (ii) link layer protocols such as [23], [24], and (iii) end-to-end protocols such as [25], [26], [20], [27], [28]. The protocols in the first two categories attempt to hide the errors of the wireless link from the TCP sender. For example, both I-TCP [21] and M-TCP [22] split one TCP connection into two separate TCP connections and perform local retransmission over a wireless link. The major difference between I-TCP and MTCP is that the latter preserves TCP End-to-End semantic while the former does not. SNOOP [29] relies on an agent running in a base station to monitor the ACKs in order to perform local error recovery thus avoiding conflicting local retransmission. AIRMAIL [23] and TULIP [24] perform error recovery at the link layer. While AIRMAIL utilizes Forward Error Correction (FEC) and Automatic Repeat Request (ARQ) to compensate transmission errors, TULIP serves as a bridge between TCP and MAC layer and performs local error recovery. Some recent works further studied the interaction between FEC applied at different layers of protocol stack and TCP. While the works of [30], [31] study performance of TCP over links offering FEC or hybrid ARQ/FEC, LT-TCP [32] introduces a FEC mechanism at the transport layer and uses ECN bits to help differentiate error-caused loss from congestion-caused loss. Although hybrid ARQ/FEC schemes can significantly improve the inefficiency associated with pure ARQ/FEC schemes, they remain ineffective for links introducing high loss rates. In contrast, utilizing proactive and reactive FEC at the transport layer, LT-TCP can perform significantly better than the above mentioned protocols even under packet error rates as high as 30-50%. The protocols in the third category rely on the TCP sender for error handling. Examples include TCP Westwood [26] and TCP-Peach [33] that rely on the TCP sender to perform bandwidth estimation, or TCP ELN [27], [28] that rely on intermediate routers for explicit signaling of packet errors to the TCP sender. Other examples include SACK [34] and SMART [35] that are proposed to reduce the delay for error recovery. Utilizing TCP ACKs flow shaping, ACK Regulator [36] is proposed to enable fast error recovery by controlling the number of losses in one sending window.

Beyond these techniques, Satellite Transport Protocol (STP) [37] and Space Communications Protocol Specifications-Transport Protocol (SCPS-TP) [38] are protocols particularly optimized for satellite links. While STP utilizes sender-requested ACK mechanism and receiver-driven retransmission

to reduce ACK traffic and avoid timeouts, SCPS-TP uses a different congestion control algorithm that can explicitly react to signals of the sources of packet loss. Furthermore, XCP and VCP are a pair of recently proposed new congestion control protocols that can perform well in high BDP networks. While both XCP and VCP are designed for wired networks, their good performance in high BDP networks make them appealing for use over satellite networks.

Notably, significant research work has been devoted to evaluate the performance of the above mentioned approaches in a variety of wireless scenarios. For example, the work of [19] compares the performance of a group of mechanisms including SNOOP, I-TCP, and various TCP flavors [35], [34], [19] in conventional cellular wireless environments. In [39] and [40], the performance of TCP Reno and Tahoe over wireless links characterized by rich scattering and Rayleigh fading is examined. The work in XCP-b [41] mainly investigates the performance of an extension of XCP in the presence of bandwidth variations in wireless networks. The work of [42] evaluates the performance of XCP over lossy links where link loss patterns follow a uniform distribution. Relying on a cross-layer analysis, this work focuses on the performance comparison of XCP, VCP, and TCP/AQM+ECN in high BDP wireless networks formed by rich scattering and line-of-sight links. Most importantly, this work concentrates on exploring the performance of VCP due to its potential for transparent deployment.

III. Simulation Studies

This section describes performance profiling results of the congestion control protocols of interest to our study over wireless networks utilizing Multiple-Input Multiple-Output (MIMO) links. We utilize NS2 discrete event simulation tool and experiment with a variety of parameter settings. In all of the experiments, each data packet has a size of 1040 bytes and each acknowledgment (ACK) packet has a size of 40 bytes. All of the links operate in full-duplex mode and link bandwidths are shared among data and ACK packets for two way traffic patterns. In all of the topologies of our experiments, we capture the error pattern of a wireless link utilizing the two-state GE Markov chain. Appendix I describes the details of the modeling of a wireless link with the GE chain including a GOOD and a BAD state. According to the discussion of Appendix I, the GE chain is fully described by two independent pairs of per state parameters. The first pair of parameters are self transition probabilities γ associated with the GOOD state and β associated with the BAD state. The second pair of parameters are per state error probabilities SNR_G associated with the GOOD state and SNR_B associated with the BAD state. We set $SNR_G = 10SNR_B$ to differentiate between the qualities of a wireless link in the GOOD and BAD states. Further, the transition probabilities of the GE model are set in one set of experiments as $\gamma = 0.99875$ and $\beta = 0.875$ representing average burst lengths of 800 and 8 bits for the GOOD and BAD state, respectively. For the same γ, we also experiment with $\beta = 0.91667$ in another set of experiments representing an average burst length of 12 bits. Four different combinations of antennas are considered for a bottleneck link as (1) single transmit single receive (1×1); (2) double transmit single receive (2×1); (3) single transmit double receive (1×2); and (4) double transmit double receive (2×2) antennas. We utilize BPSK modulation along with Reed-Solomon (RS) channel coding scheme, also described in Appendix I, to compensate against fading-caused bit errors. We use a symbol size of one byte when using RS coding and note that the channel coding rate is defined as $r = k/b$ when using RS coding.

A. Performance Analysis over A Single Bottleneck Topology

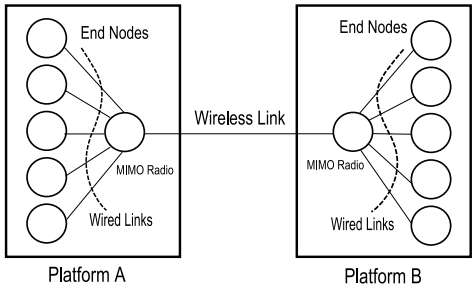


Fig. 1. A single bottleneck dumbbell topology.

Fig. 1 shows the single bottleneck dumbbell topology of the experiments of this subsection. The topology consists of a pair of platforms where each platform includes a number of nodes wired to a multiple antenna radio router playing the role of the gateway for that platform. We note that the bottleneck link could be thought of either as a wireless or a satellite link. A bottleneck wireless link is identified by a bandwidth and a delay of 54 Mbps and 80 msec, respectively. A bottleneck satellite link is assumed to have a bandwidth and a delay of 2 Mbps and 0.5 sec, respectively. The capacity of each wired link is always assumed to be twice as high as the capacity of the bottleneck link with a delay of 8 msec. The parameters of VCP follow what is reported in Table 1 of [18]. In the case of TCP/AQM+ECN, we utilize TCP Reno along with Random Early Marking (REM) [6]. Particularly, the parameters of REM follow the settings of [43] at the two ends of the bottleneck link. All of the simulations are repeated 10 times and averaged with individual runs over a total simulation period of up to 1000 seconds.

We select 10 FTP sources in Platform A sending packets to destinations in Platform B and 10 FTP sources in Platform B sending packets to destinations in Platform A. When active, each source can generate packets at the rate of its dedicated link. In the duration of an individual run, each flow starts as soon as its session is established and continues until either its session is torn down due to fading-, blockage-, and congestion-related loss or the simulation experiment has come to an end. As such, each flow follows a random start time with a random duration. We do not collect the statistics in the first 10 seconds of each run to allow for stabilizing the transitioning behavior of NS2. While we only report a small number of our results due to the shortage of space, we note that our findings are fairly consistent across a wide range of experiments.

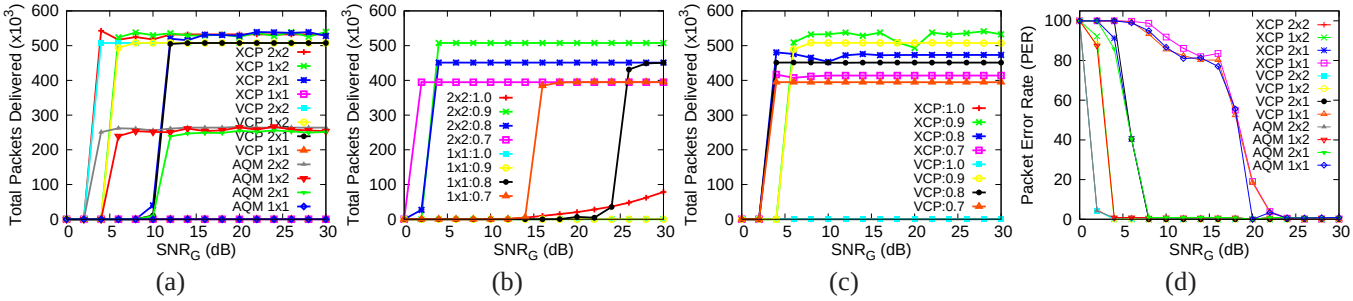


Fig. 2. A performance comparison of the protocols of our study over a bottleneck wireless link with an average burst length of 800 bits for the GOOD state. The average burst length of the BAD state is set at 8 bits for the results reported in (a) and (d), and 12 bits for the results reported in (b) and (c).

Plotting the total number of delivered packets versus SNR_G for a wireless bottleneck link scenario, Fig. 2(a) compares the performance of the three protocols for a BAD state burst length of 12 bits, a channel coding rate of $r = 0.9$, and different choices of antenna configurations.

The interesting observations are that (1) the use of different antenna configurations affects the transient behavior of each protocol as a function of link quality but not the steady-state behavior as a function of link quality and (2) the steady-state performance of XCP and VCP are much better than that of TCP/REM+ECN. The first observation is justified as both SNR_G and the antenna configurations affect the quality of the link. Each protocol exhibits its steady-state behavior once the quality of the link gets passed a certain threshold. When comparing the results of different antenna configurations, we observe that the performance of 1×1, 2×1, 1×2, and 2×2 links are in an ascending order as the result of improving link quality from the former to the latter configuration. The second observation shows that both XCP and VCP perform much better in high BDP environments with XCP taking the lead when the increase in the BDP is mostly due to bandwidth rather than delay.

Fig. 2(b) examines the effects of applying different FEC rates to data packets of VCP for 1×1 and 2×2 antenna configurations. Fig. 2(c) measures the effects of applying different FEC rates to data packets in the cases of VCP and XCP for a 1×2 antenna configuration. In all cases, the FEC rates of data and ACK packets are set to 0.93 and 0.5, respectively. The settings represent our best practical findings. It is observed that introducing a small percentage of FEC to the data packets and a more significant percentage of FEC to ACK packets at the link layer can significantly improve the performance of both VCP and XCP. The latter is due to the fact that VCP and XCP sources collect information about the status of the network from the ACK packets. Since ACK packets are only 40 bytes long and fading related random errors happen in bursts, they can corrupt the shorter ACK packets much easier than longer data packets. We note that the FEC strength must be chosen with the consideration of antenna configuration, link quality, and protocols. It is our opinion based on experiments that providing stronger protection for ACK packets as opposed to data packets can properly address the tradeoff between protocol efficiency and bandwidth overhead. It is also important to note that the sensitivity of XCP, VCP, and TCP/REM+ECN to the loss of ACK packets vary in a descending order. Upon the loss of an ACK, all protocols eventually timeout and drop their sending rate. The drop is followed by applying MI or slow start policies. It is observed that the order of sensitivity varies reversely proportional to the protocol bandwidth consumption rate.

Next, we compare the performance of the three protocols when the bottleneck link is identified by an average burst length of 8 bits for the BAD state. Fig. 2(d) sketches Packet Error Rate (PER) versus SNR_G for the three protocols and with different choices of antenna configurations. The results are fairly consistent with those shown in Fig. 2(a) indicating that packet delivery ratios are improved for smaller PERs.

Up until this point, we have reported our evaluation results over wireless bottleneck links. Next, we focus on the performance evaluation over satellite bottleneck links. We note that the main difference of a satellite link scenario with a wireless link scenario is that the increase in the value of BDP in a satellite link scenario is mostly due to an increased delay rather than bandwidth. Since the behavior of the protocols remains in part and qualitatively the same as what we have reported so far, we only focus on the major differences. Fig. 3 reports the percentage utilization of a satellite bottleneck link versus the simulation time of 1000 seconds. The data traffic is one way from Platform A to Platform B. The very interesting observation is that VCP outperforms the other two protocols. It converges rapidly and remains at a high utilization without much oscillation. To the contrary, XCP is not as well-behaved in this scenario. While XCP's oscillatory behavior is still better than that of TCP/REM+ECN, it shows wide variations in utilization. Given a bottleneck link with a larger RTT and a relatively lower bandwidth, XCP tends to overestimate the spare bandwidth and thus grows $cwnd$ too aggressively. To offset the observed overshoot as the result of the aggressive growth of $cwnd$, XCP frequently provides negative reverse feedbacks to the sender. Due to the high delay of the feedback path, XCP exhibits a lag in reacting to congestion and thus yields oscillations. While not shown due to the shortage of space, the average queue length for both XCP and VCP remains low (around 2.9%) throughout the entire experiment. It is also important to note that the results of two way traffic when each platform transmits to the other platform is the exacerbated version of the one way traffic case for both XCP and TCP/REM+ECN. However, VCP still

shows high utilization with very little oscillations.

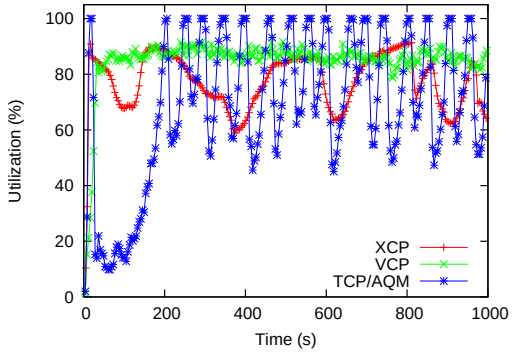


Fig. 3. A performance comparison of XCP, VCP, and TCP/AQM (TCP/REM+ECN) with 2×2 antenna configuration, $SNR_G = 30$ dB, and over a pair of satellite links. The average burst lengths of the GOOD and BAD state are set at 800 and 8 bits. FEC rates of 0.93 and 0.5 are applied to data and ACK packets, respectively.

B. Performance Analysis over A Multiple Bottleneck Topology

In this subsection, we compare the performance of XCP and VCP over a multiple bottleneck topology. The topology of our experiment is created by mixing a number of wired and wireless links where wireless links form the bottleneck links.

The illustration of Fig. 4 shows a parking lot topology used in our experiments. As observed from the figure, the four bottleneck links AB, BC, CD, and DE are assumed to be wireless. Each wired link in the figure is characterized by a capacity of 20 Mbps and a delay of 5 msec and serves a number of sources. Each wireless link is identified by a delay characteristic of 250 msec. The capacity of the links AB, BC, and DE are assumed to be 4 Mbps and the capacity of the link CD is assumed to be 2 Mbps.

While transmission over wired links is assumed to be free of bit errors, each wireless link is presumed to introduce temporally correlated bit errors the pattern of which is specified by the GE model. The average burst lengths of the GOOD and BAD states are universally assumed to be 800 and 8 bits, respectively.

The topology serves a total of five aggregate FTP flows R_1, R_2, R_3, R_4, and F_1. With the exception of F_1, each aggregate flow represents the combined traffic of sixty FTP sources generating packets at the maximum rate of their individual link. The aggregate Flow F_1 represents the combined traffic of thirty FTP sources. We refer to the flows in the aggregate flow F_1 as long flows and the flows in the aggregate flows R_1, R_2, R_3, and R_4 as local flows.

All of the wireless nodes A, B, C, D, and E are equipped with one antenna. In order to address reliability-overhead tradeoff, we set the channel coding rate of the data packets at 0.8 and the channel coding rate of the ACK packets at 0.5. Moreover, all of the simulations are run over a total simulation period of 1500 seconds. All other configurations remain the same as those in earlier experiments if not specified. We perform two sets of experiments. In the first set of our experiments, we measure the total number of delivered packets

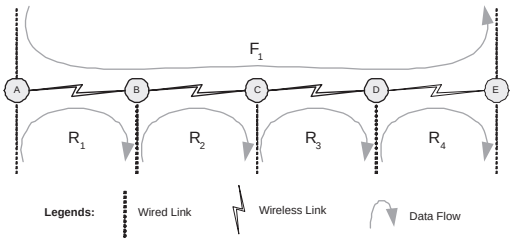


Fig. 4. An illustration of the multiple bottleneck parking lot topology used in our simulations.

as a function of SNR_G for the four wireless links. While not shown due to space limitation, we note that the performance curves of all of the four wireless links resemble hysteresis curves similar to what is shown in the single bottleneck scenario of the previous subsection. The performance of both XCP and VCP improve as the quality of the wireless links improve and the steady-state performance of XCP is slightly better than that of VCP.

In the second set of our experiments, we investigate the transient behavior of XCP and VCP as a function of time. We view our investigation as an assessment of fairness. The following describes the experimentation scenarios associated with the second set of experiments. All of the wireless nodes A, B, C, D, and E are equipped with two antennas. The wireless link qualities are all set at a high quality value of $SNR_G = 30$ dB. We set the channel coding rate of the data packets at 0.9 and the channel coding rate of the ACK packets at 0.5. As the result of our parameter settings, we assure that the wireless links are operating in their steady-state regime introducing no significant bit errors.

The flows associated with the aggregate flows F_1, R_1, R_2, and R_4 all start at the simulation time $t = 0$ and continue until the simulation time $t = 1500$. The flows associated with the aggregate flow R_3 start at the simulation time $t = 1000$ and continue until the simulation time $t = 1500$. Fig. 5(a) and 5(b) show the split of wireless link bandwidth between Local and Long Flows in the case of XCP. To our expectation, the Local Flows and the Long Flows split the bandwidth of each wireless link with a ratio of 2 to 1. As soon as the flows associated with the aggregate flow R_3 start, the 2 to 1 ratio changes to 5 to 1 considering the fact that link CD has a bandwidth of 2 Mbps as oppose to the other wireless links each offering a bandwidth of 4 Mbps.

Similarly, Fig. 5(c) and 5(d) show the split of wireless link bandwidth between local and long flows in the case of VCP. Contrary to the case of XCP, VCP exhibits an extremely poor fairness characteristic splitting the bandwidth of each wireless link with a ratio of 15 to 1. Even when the flows associated with aggregate flowR_3 are turned on, the bandwidth split ratio does not change. This demonstrates that VCP fails to achieve fairness in high BDP multiple bottleneck topologies, whether wired or wireless, serving flows with heterogeneous RTTs.

VCP achieves fairness as the result of utilizing AI and MD policies. Upon detection of an overload scenario, it takes VCP a time period equivalent to an RTT for applying the MD policy. However, it is not guaranteed that VCP changes its

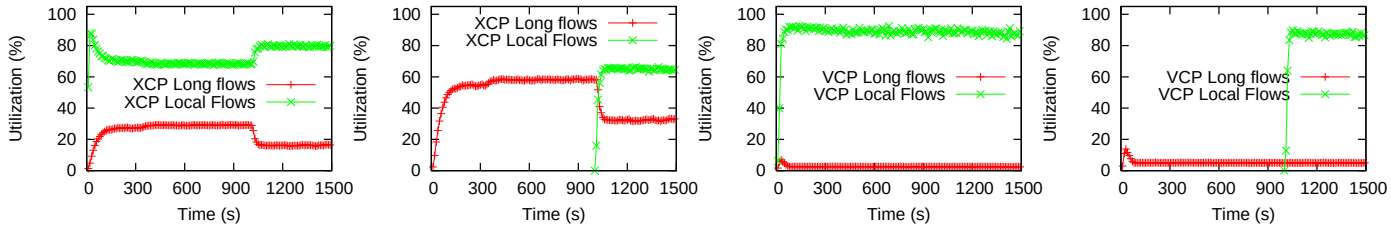


Fig. 5. A performance illustration for (a) XCP over link AB, (b) XCP over link CD, (c) VCP over link AB, and (d) VCP over link CD. Flows F_1, R_1, R_2, and R_4 are on during the simulation interval $[0, 1500]$ and flow R_3 is on during the simulation interval $[1000, 1500]$.

region from the overload to the high load region in a single MD policy enforcement, especially, when the load factor of the bottleneck link is far beyond 100%. As such, local flows can complete many iterations of AI-MD-AI policies before long flows complete the enforcement a single MD policy. Consequently, short flows can maintain their share of link bandwidth over the majority of their life span.

As a mitigation strategy, the behavior of VCP can be improved by increasing the amount of congestion information carried in the packet header and increasing the number of levels of congestion recognized. Fig. 6 shows the average queue length of each bottleneck link. While both VCP and XCP retain a low queue size, VCP builds a queue twice longer in size than XCP does.

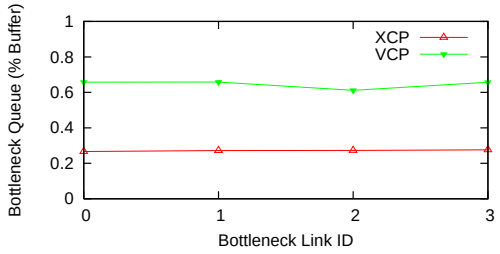


Fig. 6. Average Queue Length over each bottleneck link.

IV. A Linux Implementation of VCP

In this section, we first review the principles of our implementation approach in Linux based on providing some details about the operation of VCP. We will then continue by discussing the details of implementation.

A. Principles of Implementation

Our implementation approach defines four design principles noted below.

- *Transparency to applications:* The protocol should not require any change to applications which is important for a transparent deployment. Namely, all TCP-based applications (e.g. FTP, HTTP) are able to communicate via VCP without being aware of the existence of VCP.
- *Easy integration with Linux kernel:* VCP should be implemented as an LKM. Linux LKMs allow for adding new kernel features without recompiling the existing kernel meaning that a VCP module could be compiled, loaded, and unloaded without rebooting the host system.
- *Preserving backward compatibility:* The implementation should be entirely compatible with existing TCP options and schemes such as fast-recovery and retransmission.
- *Friendliness to other transport protocols:* A VCP-capable router should exhibit friendliness to other standard transport protocols such as TCP and UDP. More importantly, such friendliness must not adversely affect the performance of VCP.

B. Details of Implementation

Many congestion control protocols including VCP operate by manipulating the $cwnd$ of the sender. To expedite the development, we take an implementation approach that treats VCP as a protocol independent of TCP while taking advantage of its important features such as fast recovery and retransmission. More specifically, VCP is implemented as a "layer 3.5" protocol between the IP and the transport layer. From the view point of the end nodes, it appears to be a "dummy" layer while transmitting. The dummy property comes from the fact that there is no new protocol header introduced for VCP.

Furthermore, by directly grafting TCP on top of VCP, the overhead introduced by crossing protocol layers is minimized. Under such implementation methodology, TCP remains the underlying transport mechanism except that its congestion control functionality is taken over by VCP. Thus, there is no need to change existing TCP-based applications for the deployment of VCP. Simply put, our design dedicates to enable VCP functionality with a minimum overhead while keeping compatibility with legacy TCP stacks.

Since VCP manipulates cwnd solely based on the router feedback, it can potentially replace the entire Linux congestion control scheme. Based on the reasons discussed below though, we allow Linux to perform congestion control in our implementation. For the purpose of efficiency, the Linux implementation of congestion control is tightly coupled with its implementation of TCP. Since our implementation aims at preserving backward compatibility with TCP, it is neither reasonable nor effective to simply extract out TCP from the Linux implementation of congestion control. Recall that we tend to implement VCP as a layer 3.5 protocol, which means that once a packet arrives at the TCP layer after crossing the VCP layer, the information related to VCP is supposed to be lost.

Fortunately, Linux kernels after version 2.6.13 support pluggable congestion control algorithms enabling seamless integration of various congestion control algorithms with the TCP stack. In this architecture, a variety of congestion control algorithms are organized via the data structure:

```
struct tcp_congestion_ops { ... }
```

By invoking the register function below, a group of TCP congestion control algorithms such as Reno, Vegas, and BIC could be registered to TCP.

```
int tcp_register_congestion_control(
    struct tcp_congestion_ops *ca)
```

The latest registered algorithm would be the active congestion control algorithm. Such architecture provides VCP with an elegant interface to interact with Linuxs native congestion control scheme. Thus, we implement VCP as a pluggable congestion control algorithm of TCP. Doing so results in not only conforming to our principle mentioned earlier but also being very efficient.

Specifically, VCP consists of two components deployed on the *end-host* side and the *router* side. Both components are implemented as LKMs, and hence can be transparently enabled and disabled. In what follows each module is described separately.

We discuss VCP's end-host module first. Our implementation defines a new IP packet type called VCP with the protocol number 200. When installing the VCP host module by calling the function

```
int inet_add_protocol(
        struct net_protocol *prot,
        unsigned char protocol)
```

VCP is registered to IP and signals IP to deliver the packets with protocol number of 200 to the VCP's receiving handler. Meanwhile, the TCP's sending function is changed from *ip_queue_xmit()* to *vcp_queue_xmit()* in order to detour a TCP packet to VCP rather than IP. Since VCP is implemented as one of the congestion control algorithms of TCP, a new congestion control algorithm called *tcp_vcp* is registered with TCP while installing the VCP module. As a result, the algorithm *tcp_vcp* is put in charge of updating the *cwnd* of TCP. In what follows, we explain what happens to outgoing and incoming paths of a TCP packet after enabling VCP.

- Outgoing packet path: While transmitting, a TCP packet generated by the standard TCP stack is bypassed to VCP before it is forwarded to IP. In *vcp_queue_xmit()*, the member variable *sk_protocol* in *socket* data structure is changed from 6 (TCP) to 200 (VCP). Meanwhile, the ECN field is marked as "01". Then, the VCP packet is forwarded to IP for transmission by the direct function call *ip_queue_xmit()*. We note that there is no need for a new VCP packet header although VCP is embedded between TCP and IP as a "layer 3.5" protocol. Furthermore, there is nearly zero overhead introduced in the outgoing path of a TCP packet. Fig. 7(a) compares the outgoing data path of VCP with that of standard TCP.
- Incoming packet path: While receiving, a VCP packet is delivered from IP to the VCP's receiving handler due to the fact that VCP has been registered on top of the IP protocol. After saving the value of ECN bits, VCP forwards the packet to TCP leaving the rest of packet processing to TCP. However, the packet forwarding from VCP packet to TCP represents a tradeoff between efficiency and transparency. Fig. 7(b) compares the incoming data path of VCP with that of standard TCP. Note that a VCP packet needs to pass through $backlog$ and IP twice, which introduces processing overhead.

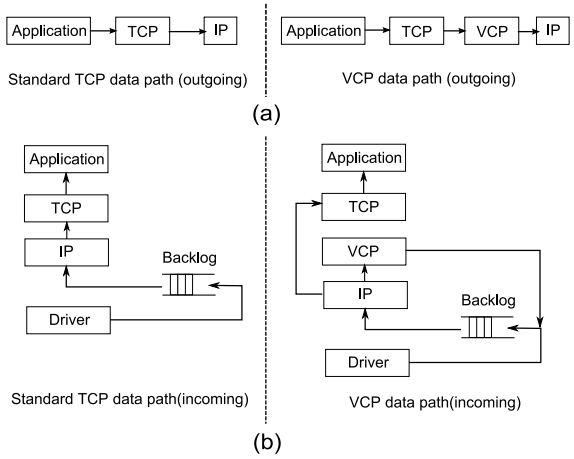


Fig. 7. A comparison of the (a) outgoing and (b) incoming path of standard TCP data packet with that of VCP data packet.

Next, we continue by discussing VCP router module. As presented in [18], a VCP capable router should be able to (i) sample and compute network load on the network link, and (ii) intercept VCP packets and mark ECN bits.

Taking advantage of $qdisc$ and timers, sampling and computing network load could be easily achieved in Linux. With regard to handling VCP packets, $netfilter$ module [44] module, built-in Linux kernel, can be used to intercept and manipulate network packets. Via registering a $hook$ function at the point of "NF_IP_POST_ROUTING" on the router, VCP packets could be intercepted by $netfilter$ and forwarded to the VCP router module wherein the ECN bits of the IP packet header are marked as LOW_LOAD, HIGH_LOAD, or OVER_LOAD.

It is worth noting that such implementation allows for the co-existence of VCP with standard transport protocols such as TCP and UDP. Importantly, the mixture of flows of different protocols (VCP, TCP, UDP) does not adversely affect the performance of VCP.

We note that besides our approach described above, there are a couple of alternative approaches to implement VCP. Those are: (i) introducing a new TCP option similar to what is proposed in [42], or (ii) replacing the entire TCP protocol stack. While the former approach might require a separate implementation for each possible transport protocol, the latter introduces significant compatibility issues. In contrast, our approach is simple and efficient, while keeping compatibility with legacy TCP stacks. Most importantly, any new congestion control algorithm and/or protocol can be transparently implemented within the Linux kernel relying on our approach. In

what follows, we describe the implementation steps.

1) Give a name to the new congestion control algorithm, which will be filled in $char * name[TCP\_CA\_NAME\_MAX]$ of struct tcp_congestion_ops.
2) Define a new protocol number and register it with the IP layer the same way VCP is implemented in this section.
3) Implement functions defined in struct tcp_congestion_ops. Note that,

```
void (*cong_avoid)(...);
unsigned int (*ssthresh)(...);
unsigned int (*min_cwnd)(...);
```

must be implemented.
4) At the router side, register a hook function to $netfilter$ at the point of "NF_IP_POST_ROUTING" to capture packets with the newly defined protocol number. Once a packet is passed to the hook function, necessary operations, such as ECN marking, can be performed.

We will further discuss implementation alternatives in Section VI.

As revealed in Section III, the performance of a congestion control protocol over a lossy link depends on the speed of recovery following a loss. The latter is determined by the choice of the protocol's MI and AI parameters. This fact implies that there might be a better way for congestion control protocols to respond to loss over wireless lossy links. Since all such protocols perform an MD operation after a loss without differentiating the cause of that loss, an error-caused loss forces an unnecessary reduction of $cwnd$ value thus degrading performance. To identify the source of loss, in our implementation, we propose and implement a heuristic scheme presented below.

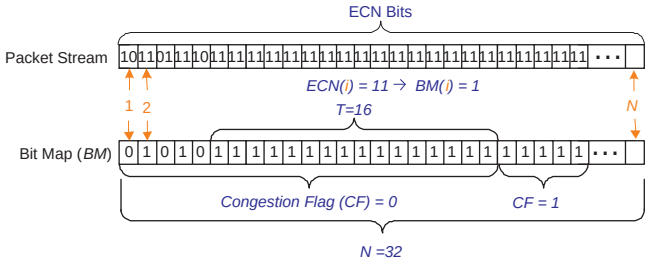


Fig. 8. An illustration of the loss differentiation heuristic algorithm.

Loss differentiation heuristic algorithm: Intuitively, a sender can build knowledge about whether the network is congested as it keeps receiving feedback from its intended receiver. Given the fact that the feedback is updated with the receipt of every ACK, it is reasonable to assume that the congestion status of a network can be continuously tracked by the sender. It is specially important to realize that a congestion-caused loss event has a much longer duration than an error-caused loss event. Relying on the above fact, our heuristic assumes that a sender can identify the cause of a loss by keeping track of the status of the network. In order to track the status of the network, our heuristic algorithm proposes maintaining a revolving congestion history Bit Map (BM) of size N at the sending side. Upon the receipt of an ACK, the bit at position $BM(1)$ is dropped, the bit at position $BM(i)$ with $i \in \{1, \cdots, N\}$ is shifted to the left so it takes the position of bit $BM(i-1)$, and the bit at position $BM(N)$ is set to 1 if the new ACK indicates congestion or otherwise to 0. If at any time, the right most T consecutive bits with $T \leq N$ are set to 1 in the bit map, a binary flag called Congestion Flag (CF) is set to 1. Otherwise, the flag is set to 0. Upon detection of a loss, if CF flag is set, then the loss is safely determined as a congestion-caused loss triggering an MD operation to $cwnd$. Otherwise, the loss is considered to be an error-caused loss and the sender simply maintains the current $cwnd$. As an alternative, T could represent the total (as opposed to the consecutive) number of bits set to 1 in every N bits. Investigating the impact of the choice of parameter N in the performance of the algorithm is the subject of our future work. In the case of VCP, the link load factor is encapsulated in ACK packets and the $OVER\_LOAD$ represents a load factor beyond 100%. Thus, $OVER\_LOAD$ is used as the indicator of congestion. According to our experiments, setting N to 32 and T to 16 represent best practical findings. We note that with our choices of values, maintaining a revolving bit map history only requires 4 bytes of storage on a per flow basis. While N should essentially be a function of flow $cwnd$, we set the value of N to 32 for the convenience of implementation. We also note that the value of $cwnd$ for larger flows could be easily scaled to fit the 32 bits of N. Fig. 8 illustrates the operation of our heuristic algorithm. Similarly, we also integrate this scheme with the implementation of XCP except that in the case of XCP, negative feedback is used as an indicator of congestion. In our experimental studies presented in next section, this scheme is applied to both XCP and VCP.

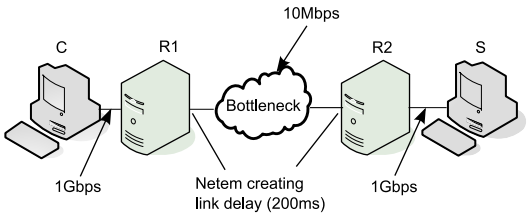


Fig. 9. The setup of our experimental Linux environment.

V. VCP Experimental Evaluation

In this section, we present the results of our experimental study conducted in a Linux testbed. All of the nodes utilize Fedora Core 5 (FC5) distribution of Linux. From among the flavors of TCP, we select TCP BIC for our experiments as it outperforms TCP Westwood, TCP Peach, TCP Reno, TCP Vegas, and others inspected. Utilizing TCP BIC [7], we compare the performance of TCP Drop Tail (TCP/DT), TCP Random Early Drop (TCP/RED), XCP, and VCP. Fig. 9 shows our single bottleneck experimental setup. VCP end-host code runs at the end nodes. VCP router code runs on routers R1 and R2. The 10Mbps bandwidth of the full-duplex bottleneck link between R1 and R2 is controlled by the $ethtool$ interface provided in FC5. A bottleneck link delay of 200 msec

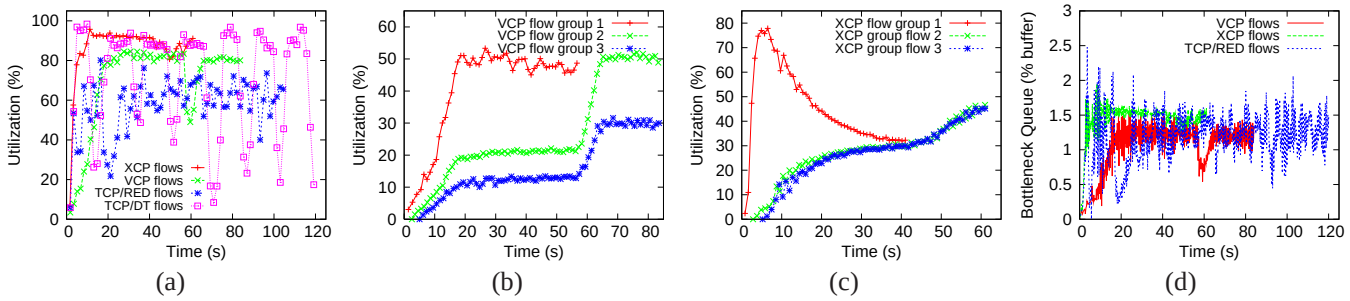


Fig. 10. A bandwidth utilization profiling of TCP/DT, TCP/RED, XCP, and VCP for the experimental setup of Fig. 9 capturing (a) a comparison of the four protocols, (b) VCP, (c) XCP. Queuing dynamics of VCP, XCP and TCP/RED are compared in (d).

is introduced utilizing $netem$ utility [45] on both routers to create a 400ms Round Trip Time (RTT). An extra delay of $0 \sim 200$ msec is introduced at side links to vary RTTs of each VCP flows. FTP servers are set up at both end hosts using $vsftpd$ daemon available in FC5 distribution of Linux. During the first 10 seconds, three groups of FTP requests start from each end host to another at a random time each requesting to transfer a 5 MB file. Three RTT ranges, 400ms, 500ms, 550ms, are associated with these groups and each group consists of five FTP requests. Flows in an individual group have RTTs with slight variation. Both side links from end hosts to routers operate in full-duplex mode and have a bandwidth of 1 Gbps. Furthermore, during transmission, fixed rate two-way UDP and TCP flows are served as background traffic.

To optimize the performance of TCP, a variety of TCP parameters are fine tuned following the TCP performance tuning guides of [46] and [47]. VCP parameters are set as presented in [18]. Moreover, we use our revised version of the implementation of XCP presented in [42]. With regard to RED cases, the drop probability is set to 0.1, minimum and maximum queue size are set to one third and two thirds of the queue buffer size, respectively.

The experimental studies are two fold: one for a wired bottleneck link and another for a wireless bottleneck link. In the case of a wireless bottleneck link, wireless link effects are emulated via another LKM that implements the Gilbert-Elliott error model as a mean of capturing temporally correlated fading characteristics of the link. Using this emulated wireless network, we validate the simulation results of Section III.

A. Performance in Wired Networks

In this subsection, we measure the performance of TCP/DT, TCP/RED, XCP, and VCP in a wired environment to build our baseline and validate our implementation. Next, we explore the performance of the protocols in wireless environments with a variety of parameter settings. Fig. 10 illustrates the results of our experimental study for the wired bottleneck link.

Fig. 10(a) shows a significant performance gap between TCP and VCP. As expected, both VCP and XCP consistently achieve high bandwidth utilization. In contrast, TCP/DT illustrates an unbalanced bandwidth allocation behavior in the presence of competing flows. Although TCP/RED significantly improves the performance of TCP/DT, TCP/DT remains ill-behaved in terms of its fairness characteristic. We notably observe that after flows with minimal RTT finish, VCP takes a longer time to re-converge thus yielding a sudden drop in bandwidth utilization while XCP shows a small oscillation.

Furthermore, in the figure, the end point of each curve denotes the moment at which all FTP flows finish. Table I shows the average FTP completion time of each protocol for all of the flows taken over 10 runs. Notably, XCP outperforms VCP in terms of average completion time. However, it is worth noting that the performance gap comes from the difference in the speed of convergence. In our experiments, XCP takes an average of less than 10 seconds to converge, while VCP has an average close to 20 seconds.

Fig. 10(b) and 10(c) illustrate group-wise utilizations of VCP and XCP, respectively. As illustrated in Section III, VCP tends to allocate more bandwidth to flows with shorter RTT in high delay scenarios. Thus, VCP fails to converge and shows three steps with ratios of 5:2:1 in its steady-state status. In contrast, XCP finally converges. While not shown in the paper, all VCP flows consume bandwidth evenly for the case of flows with the same RTT. To the contrary, XCP flows show slight oscillations. The oscillations of XCP imply that the high convergence speed of XCP comes at the cost of fairness.

TABLE I
AVERAGE FTP COMPLETION TIME OF DIFFERENT PROTOCOLS

Protocols	Average FTP Completion Time (s)
TCP/DT	121
TCP/RED	102.5
XCP	55.2
VCP	72.1

Fig. 10(d) shows the bottleneck queue dynamics of VCP, XCP, and TCP/RED. The queue dynamics of all three protocols show the same pattern as the one shown in Fig. 10(a). While no protocol consumes more than 3% of the buffer size of the bottleneck link, TCP/RED shows a highly oscillatory pattern.

B. Experimental Evaluation with the Wireless Emulator

In this section, we compare the performance of TCP, XCP, and VCP in the same wired testbed as the one utilized in the previous subsection. However, we now assume that the bottleneck link is wireless and emulate wireless link effects through an LKM implementation of the GE error-model described in Section A. Our implementation of the GE model

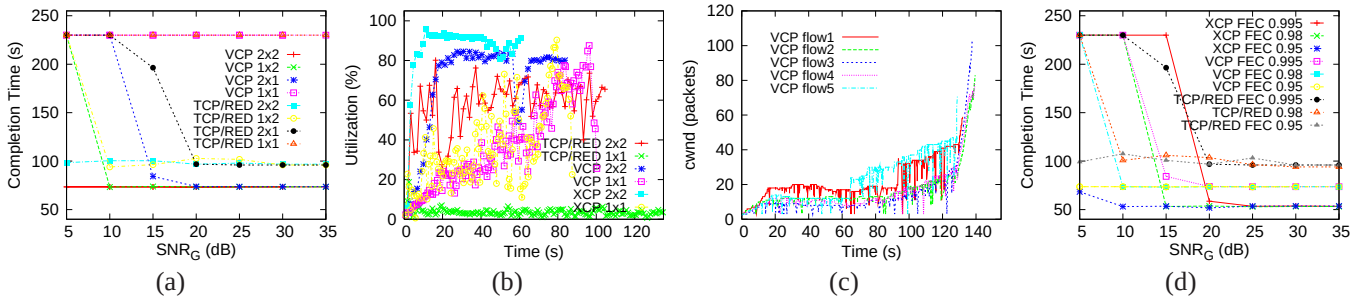


Fig. 11. A performance comparison of the congestion control protocols of Fig. 9 when the bottleneck link is wireless. The figures capture (a) the effects of antenna configurations on FTP completion times, (b) the bandwidth utilization of VCP, XCP, and TCP/RED, (c) congestion window dynamics of VCP, and (d) the effects of FEC rates on FTP completion times.

is independent of our VCP implementation, yet it still takes advantage of the $netfilter$ module in the Linux kernel. While the core iterative processing of the GE model including state transitioning operations is implemented in the kernel space, the parameters of the GE model are calculated by an application in the user space. Further, we implement a new system call in order to forward the parameters of the model from the application operating in the user space to the LKM operating in the kernel space. Due to space limitation, we omit further explanation of the implementation details. In this section, the configuration of error-model and the description of parameters follow those of Section III. In what follows, we proceed with reporting the results of our study.

1) Antenna Configuration Effects: Fig. 11(a) compares the completion times of VCP and TCP/RED for various antenna configurations. In all cases, VCP outperforms TCP/RED completing transmission 20% to 40% faster than TCP/RED in time. While not shown in the figure due to the limitation of space, XCP achieves slightly better completion times than VCP in all cases. As illustrated, the performance is directly related to the quality of the channel. For small values of SNR_G representing a low channel quality, a Packet Error Rate (PER) close to 100% yields a very long completion time. As the quality of channel improves, the PER improves eventually approaching zero. The performance of 1×1, 2×1, 1×2, and 2×2 links are in an ascending order as the result of improving SER from the former to the latter configuration. For proper scaling of the performance gap in the figures, completion times larger than 500 seconds are scaled down to 230 seconds.

We observe a very interesting phenomenon pertaining to an observation that the completion time of TCP/RED is not quite proportional to SNR_G. We observe that the loss associated with congestion decreases as the link quality improves. We justify our observation as follows. While the link quality drops for lower values of SNR_G, random bit errors introduce more packet loss somewhat limiting the explosion of the $cwnd$ and thus achieving a RED-like effect, i.e., by dropping packets earlier congestion losses are reduced. As a result, better completion times are achieved.

In contrast, both VCP and XCP are anticipated to maintain relatively constant completion times. As they do not rely on the loss to regulate the $cwnd$, the throughput of both protocols is not affected much by the wireless loss. Fig. 11(b) verifies this anticipation where in all cases a high quality wireless link with $SNR_G = 30$ dB and no FEC is utilized. As expected, both VCP and XCP outperform TCP/RED in all of the cases. However, TCP is affected not as much as VCP and XCP by the link quality. Instead, the occurrence of random bit errors can even help TCP. Fig. 11(c) shows the congestion window dynamics of VCP over a bottleneck link with a 1×1 antenna configuration utilizing no FEC. The loss associated with the fading effects reduce the speed of VCP convergence as fast recovery and retransmission processes are always triggered upon observing a loss.

2) Forward Error Correction Effects: As reported in Section III, applying a low level of FEC protection to ACK packets can result in a significant ACK packet loss. The latter is due to the fact that short ACK packets are corrupted with a smaller number of bit errors. As verified by our experiments, this phenomenon can significantly affect the performance of both VCP and XCP as they both heavily rely on the congestion information in the ACK packets. Fig. 11(d) shows the effects of enabling link layer FEC for a 2×1 MIMO link. As observed from the figure, applying an appropriate percentage of FEC at the link layer can significantly improve the performance of both VCP and XCP. While not shown due to the shortage of space, we have observed similar results for other configurations of MIMO links.

3) The Effects of Large Link Delays: In this section, we compare the performance of VCP and XCP for large values of RTT. Recall that our simulation results of Section III-A demonstrated VCP outperforms XCP in high delay, moderate bandwidth wireless networks. Fig. 12 shows the experimental results of our emulated testbed for a bottleneck link with an RTT of 1600 msec. It is observed from the figure that the pattern of the results is consistent with the simulation results. Note that, VCP achieves higher bandwidth utilization without much oscillation while XCP shows wide variations in bandwidth utilization. Although XCP has a shorter completion time than VCP, as presented earlier, the performance gap comes from the differences in the convergence speed which will be compensated as the size of downloaded file increases.

4) The Effects of the Heuristic Scheme: Fig. 13 compares the bandwidth utilization of the original VCP and VCP with the heuristic algorithm over a lossy link. In our experiments, we utilize a 2×1 antenna link with an SNR_G of 10dB. The

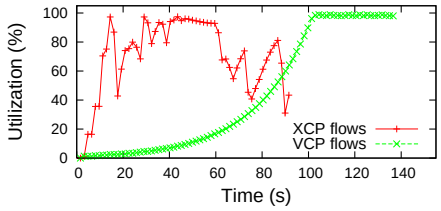


Fig. 12. A performance comparison of XCP and VCP over a bottleneck link with an RTT of 1600 msec. Other configurations are the same as those of Fig. 3.

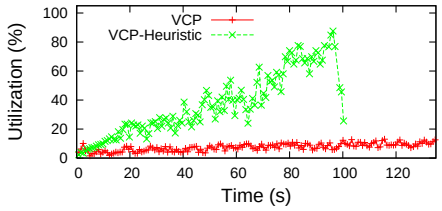


Fig. 13. The effect of the heuristic scheme on the performance of VCP with 2×1 antenna configuration, $SNR_G = 10$ dB, and over a single bottleneck link.

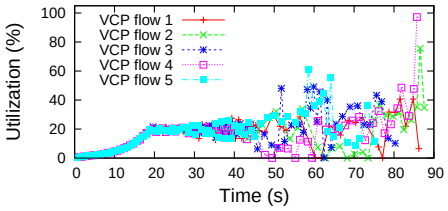


Fig. 14. Bandwidth utilization of VCP in the wireless network of Fig. 9 as the link capacity varies.

average burst lengths of the GOOD and BAD state are set to 800 and 8 bits. FEC rates of 0.95 and 0.5 are applied to data and ACK packets, respectively. As shown by the figure, VCP fails to efficiently increase the value of $cwnd$ in the absence of the heuristic scheme, and therefore exhibits a low utilization characteristic. In contrast, VCP with the heuristic scheme can identify the source of loss, ignore error-caused loss, and thus achieve a significantly better bandwidth utilization than VCP. However, the latter shows oscillations due to the associated retransmissions and timeouts.

VI. Considerations

Although our experiments show that VCP is an efficient congestion control protocol in wireless networks, there are several open issues and weaknesses that bear further studies.

Implementation alternatives: As mentioned earlier, our implementation passes a VCP packet through $backlog$ and IP layer twice in order to preserve transparency to transport layer protocols as well as applications. To evaluate the overhead of our approach, we have also implemented a non-transparent version of VCP which directly forwards all VCP packets to TCP. In our testbed, the measured time overhead of both implementations turns out to be nearly identical and relatively small in comparison with the link delay. Notably, the overhead is introduced only in end hosts. Another alternative is to use an approach similar to that of [42] in which packets will have to be forwarded to TCP at every router along the path between a sender and a receiver introducing a much higher overhead. It is worth noting that none of the implementations of XCP and VCP including this work can completely prevent TCP from changing $cwnd$. Currently, all of the implementations have to reset the $cwnd$ to their desired value after TCP changes it. Fig. 11 (c) shows this effect. While it is not clear to us how this could affect the overall performance of VCP in large scale networks, we note that the prevention of such behavior requires completely eliminating TCP from the stack which may not be feasible in terms of deployment.

Global deployment issues: Consistent with the reported results of [42], VCP fails to converge in heterogeneous network environments where non-VCP routers exist. If the bottleneck link is a non-VCP link, VCP is not able to identify the real bottleneck link and thus makes an over-optimistic feedback. Furthermore, the differentiation between loss caused by fading and loss caused by timeouts remains to be an open issue even for all VCP networks with heterogeneous values of RTT and large deviations in the values of RTT.

Biased fairness: As shown in Section III, VCP exhibits a poor fairness characteristic in multiple bottleneck networks due to the fact that three congestion levels employed by VCP are not accurate enough to represent the congestion phenomenon. While it may be possible to use a larger number of bits in the IP header to increase the accuracy of VCP, such approach introduces significant global deployment issues. Nonetheless, our experiments not reported in this paper show that the use of multiple bits would improve the fairness. To keep compatibility and maintain transparency, using multiple packets to carry more accurate congestion information is deemed a promising solution. Our current ongoing work involves the development of a multi-packet version of VCP.

Bandwidth estimation issues: Although the link capacity never changes in wired networks, it varies dynamically in wireless networks imposing a significant impact on the performance of VCP. As VCP calculates the network load factor based on the link capacity, an over-estimated link capacity could result in an inflated *cwnd* at the sender potentially forming longer queues and causing congestion-related losses. Unfortunately, it is not easy to adaptively adjust the estimation of the link capacity based on the link quality. We demonstrate one such situation in Fig. 14 by dynamically forcing a lower bandwidth of the bottleneck link. In the experiment, the link capacity initially estimated by VCP agrees with the actual link capacity. After 20 seconds, the link capacity drops by a factor of 50%. Fig. 14 shows that VCP fails to converge due to an over-estimation of the link capacity. This is another open issue for any congestion control protocol relying on the estimation of the link capacity in wireless networks. We are currently working towards addressing the issue through the use of a dynamic link bandwidth estimator.

VII. Conclusion

In this paper, we described a cross-layer study focusing on performance profiling of TCP/AQM+ECN, XCP, and VCP in wireless networks. Our study characterized wireless networks with random bit errors caused by fading and packet erasures caused by network buffering. We utilized finite-state Markov chains to model the bit error characteristics of a wireless link and utilized the use of FEC codes at the link layer in order to combat the effects of bit errors. Our performance profiling results illustrated the sensitivity of both XCP and VCP to the loss of congestion information carried in the ACK packets.

Furthermore, we reported the results of our implementation of VCP and experimental study on its performance. Our

implementation was transparent to applications and allowed for the co-existence of VCP with standard transport protocols such as TCP and UDP. We conducted our experimental study in a wired testbed consisting of Linux nodes realistically emulating the fading characteristics of wireless links in the Linux kernel.

Our experimental results demonstrated the necessity for protecting not only a congestion control protocol's data but more importantly its metadata against bit errors. Such protection was provided relying on the use of FEC codes and/or improving the quality of a link by utilizing multiple antennas. Our results further revealed that VCP represents a high performing yet practical congestion control protocol for wireless networks specially for encrypted networks restricting the number of available header bits for use by a congestion control protocol. Given that ACK packets are reasonably protected through the use of link layer FEC, our experimental results also showed that XCP outperforms VCP in high bandwidth moderate delay scenarios. However, in moderate bandwidth high delay scenarios that are typical of satellite links, VCP outperforms XCP. Both VCP and XCP outperform TCP/AQM+ECN in high bandwidth-delay product scenarios of our study.

Our future work includes improving the oscillatory behavior of VCP in the presence of link bandwidth estimation errors and enhancing the performance of VCP in encrypted wireless networks by designing and implementing a multi-packet protocol version of the protocol.

APPENDIX
PHYSICAL AND LINK LAYER ANALYSIS

In this section, we analyze the underlying physical and link layers of a wireless link and provide cross-layer hooks for relating the effects of such layers to the performance of network and transport layers where congestion control protocols are hosted.

Our discussion covers the modeling of fading wireless links with finite-state Markov chains and using a channel coding scheme to compensate against the effects of temporally correlated bit errors caused by fading. Our modeling approach of fading links is general and can be applied to both rich scattering and line-of-sight links whose fading characteristics are typically captured through the use of Rayleigh and Rician distributions. While our channel coding discussion proposes the use of RS codes, it can nonetheless be applied to other channel coding schemes such as convolutional codes.

A. Markov Chain Modeling of Wireless Links

As pointed out in [48], [49], [50], and many other articles, a MIMO fading wireless link is characterized by temporally correlated bit errors. Finite-state Markov chains can provide an elegant mathematical model to capture the error behavior of such link. A finite-state Markov chain is fully described by its parameters, state transition probabilities, and per state error rates. The number of states in a Markov chain is typically chosen such that it can balance the tradeoff between the complexity and the accuracy of the model. We propose the use of the two-state GE chain characterized by a GOOD and a BAD state as the best practical alternative of addressing accuracy-complexity tradeoff. In the GE model, the GOOD state introduces a probability γ of staying in the same state and a probability $1-\gamma$ of transitioning to the BAD state while the BAD state introduces a probability β of staying in the same state and a probability $1-\beta$ of transitioning to the GOOD state. The burst length of the GOOD (BAD) state BL_G (BL_B) represents a moving average value capturing the number of symbols received in the GOOD (BAD) state. The parameters γ and β can be measured from the observed average burst lengths of a wireless link as $BL_G = \frac{1}{1-\gamma}$ and $BL_B = \frac{1}{1-\beta}$, [51].

The GOOD state also represents symbol loss with a probability ε_G while the BAD state represents symbol loss with a probability ε_B where $\varepsilon_G << \varepsilon_B$. The pair of probabilities ε_G and ε_B are related to quantities such as signal-to-noise ratio or signal-to-interference-noise ratio. In what follows, we provide a brief discussion of how the quantities ε_G and ε_B can be extracted. The details of calculations for rich scattering links characterized by Rayleigh fading and line-of-sight links characterized by Rician fading are discussed in [49] and [50], respectively.

In [48], closed-form expressions describing the modulation symbol error rate of MIMO fading wireless links are identified in terms of modulation constellation points M and the average received signal-to-noise ratio. We apply the previous results to capture modulation symbol error rates of wireless links which may be accommodating multiple antenna nodes.

We introduce the modulation symbol error rate of the GOOD state for a link associated with single transmit and N receive antennas using Maximum Ratio Combining (MRC) as

$$\begin{array}{l}\varepsilon_G = \frac{M-1}{M} - \frac{1}{\pi}\sqrt{\frac{\vartheta_G}{1+\vartheta_G}}\Big\{(\frac{\pi}{2} + \tan^{-1}\xi) \\ \sum_{j=0}^{N-1}\binom{2j}{j}\frac{1}{[4(1+\vartheta_G)]^j} + \sin(\tan^{-1}\xi_G) \\ \sum_{j=1}^{N-1}\sum_{i=1}^{j}\frac{\sigma_{ij}}{(1+\vartheta_G)^j}[\cos(\tan^{-1}\xi)]^{2(j-i)+1}\Big\}\end{array} \quad (1)$$

where $\vartheta_G = SNR_G \sin^2(\frac{\pi}{M})$, $\xi_G = \sqrt{\frac{\vartheta_G}{1+\vartheta_G}}\cot\frac{\pi}{M}$, and $\sigma_{ij} = \frac{\binom{2j}{j}}{\binom{2(j-i)}{j-i}4^i[2(j-i)+1]}$. By replacing SNR_G with SNR_B in Equation (1) where $SNR_G >> SNR_B$, the modulation symbol error rate of the BAD state for a link associated with single transmit and N receive antennas using MRC is identified.

Relying on a discussion of diversity gains, we also argue that per state modulation symbol error rates of Space-Time Block Codes (STBCs) of [52] and [53] can be calculated from Equation (1) by proper scaling of the values of SNR_G and SNR_B. For example, per state modulation symbol error rates of a link associated with double transmit single receive antenna links can be calculated by replacing SNR_G and SNR_B with $\frac{SNR_G}{2}$ and $\frac{SNR_B}{2}$ and setting N to 2 in Equation (1). Similarly, per state modulation symbol error rates of a link associated with double transmit double receive antenna links can be calculated by replacing SNR_G and SNR_B with $\frac{SNR_G}{2}$ and $\frac{SNR_B}{2}$ and setting N to 4 in Equation (1).

In our experiments, we set $M = 2$ and work with BPSK as a modulation symbol is mapped to a bit for BPSK. While

our choice of BPSK reflects a matter of convenience for our study, it does not affect the generality of our model.

Let $\varphi(t, r, G)$ and $\varphi(t, r, B)$ denote the probability of receiving r bits from t transmitted bits and winding up in the GOOD and the BAD state of the GE model, respectively. Then the overall probability of receiving r bits from t transmitted bits is given by

$$\varphi(t, r) \;=\; \varphi(t, r, G) + \varphi(t, r, B) \tag{2}$$

where the recursive probabilities $\varphi(t, r, G)$ and $\varphi(t, r, B)$ are given [49] by

$$\begin{aligned} &\varphi(t, r, G) \;= \\ &\quad \varepsilon_G \left[\gamma\, \varphi(t-1, r, G) \;+\; (1-\beta)\varphi(t-1, r, B)\right] \\ &\quad (1-\varepsilon_G) \left[\gamma\, \varphi(t-1, r-1, G)\right. \\ &\qquad\qquad \left. +\, (1-\beta)\varphi(t-1, r-1, B)\right] \end{aligned} \tag{3}$$

and

$$\begin{aligned} &\varphi(t, r, B) \;= \\ &\quad \varepsilon_B \left[(1-\gamma)\, \varphi(t-1, r, G) \;+\; \beta\varphi(t-1, r, B)\right] \\ &\quad (1-\varepsilon_B) \left[(1-\gamma)\, \varphi(t-1, r-1, G)\right. \\ &\qquad\qquad \left. +\, \beta\varphi(t-1, r-1, B)\right] \end{aligned} \tag{4}$$

for $t \geq r > 0$ and the initial conditions

$$\begin{aligned} \varphi(0, 0, G) &= g_{ss} \;=\; \tfrac{1-\beta}{2-\gamma-\beta} \\ \varphi(0, 0, B) &= b_{ss} \;=\; \tfrac{1-\gamma}{2-\gamma-\beta} \\ \varphi(1, 0, G) &= \varepsilon_G \left[\gamma\, g_{ss} \;+\; (1-\beta)\, b_{ss}\right] \\ \varphi(1, 0, B) &= \varepsilon_B \left[(1-\gamma)\, g_{ss} \;+\; \beta\, b_{ss}\right] \end{aligned} \tag{5}$$

B. Channel Coding and Associated Error Modeling

As indicated by various standards such as Direct Sequence Multiple Access Code Division (DS-CDMA) [54] and WiMax [55], the use of Forward Error Correction (FEC) schemes at the link layer has been proposed in order to combat the fading effects of wireless links.

In our work, we propose the use of Reed-Solomon (RS) channel coders at the link layer as an optional component to compensate for temporally correlated loss effects of the wireless channel. We note that the use of other channel coders besides RS coders is also viable. That said, an RS channel coder $RS(b, k)$ converts k channel coding symbols into a b-symbol block by appending $(b-k)$ parity symbols. Such channel coder is able to correct as many as $t_C = \lfloor \frac{b-k}{2} \rfloor$ symbol errors in a block.

We distinguish between modulation symbols and channel coding symbols by noting that a channel coding symbol typically consists of a number of modulation symbols. For example, an 8-bit RS channel coding symbol consists of eight BPSK modulation symbols. If at least $b - t_C$ channel coding symbols are correctly received from b transmitted channel coding symbols, the whole block is recoverable. We note that our approach calls for mapping a channel coding block onto a packet. Since an RS channel coder $RS(b, k)$ at the link layer treats an assembled packet as its data block k, errors in both the payload and headers/trailers of the network and higher layers can be mitigated for each packet individually at the receiving side.

Next, we discuss our error modeling approach. Suppose the RS coder generates a set of channel coding symbols where each symbol consists of s bits. A channel coding symbol is received error free if all of its s bits are received free of errors. We assume that the loss pattern of the wireless channel at the bit level is described by the GE model. Thus, the probability of receiving a channel coding symbol free of error under the GE model is obtained from Equation (2) with $t = r = s$ as $\varphi(s, s)$.

Relying on a hybrid loss model in which inter-symbol temporal correlation is not deemed significant compared to intra-symbol temporal correlation and since inter-symbol temporal correlation is captured in the expression $\varphi(s, s)$ for back-to-back transmitted channel coding symbols, we can obtain the probability of channel coding symbol block loss as

$$\Psi(b, t_C, \varepsilon_G, \varepsilon_B) \;=\; \sum_{i=0}^{b-t_C-1} \binom{b}{i} (1-\varphi(s,s))^i \varphi(s,s)^{(b-i)} \tag{6}$$

We refer the interested reader to [56] for validation results of our model.

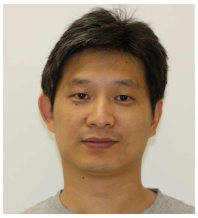
ACKNOWLEDGMENTS

We would like to thank Dr. Yong Xia for his insights on performance fine tuning of VCP. We would also like to thank Dr. Thomas R. Henderson for providing us with the XCP prototype code in Linux.

REFERENCES

[1] M. Goutelle, Y. Gu, and E. He, "A Survey of Transport Protocols other than Standard TCP," 2004, available at http://www.gridforum.org/documents/GFD.55.pdf.
[2] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988, pp. 314–329.
[3] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," in *Proc. of the 2001 ACM SIGCOMM Conference*, 2001.
[4] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," in *IETF RFC 3168*, 2001.
[5] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 9, pp. 397–413, Aug 1993.
[6] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active Queue Management," *IEEE Network*, vol. 15, no. 3, pp. 48 – 53, May/Jun. 2001.
[7] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," in *Proc. of the Infocom 04*, 2004.
[8] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," in *Proc. of the PFLDNet'05*, Feb. 2005.
[9] S. Floyd, "HighSpeed TCP for Large Congestion Windows," Aug. 2002.
[10] D. Chiu and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks," *J. of Computer Networks and ISDN*, vol. 17, no. 1, pp. 1 – 14, Jun. 1989.
[11] D. Leith and R. Shorten, "H-TCP: TCP for High-speed and Long-distance Networks," in *Proc. of the PFLDNet'04*, Feb. 2004.
[12] T. Kelly, "Scalable TCP: Improving Performance in HighSpeed Wide Area Networks," Feb. 2003, available at http://wwwlce.eng.cam.ac.uk/ctk21/scalable/.
[13] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. of the Infocom 04*, 2004.
[14] S. Bhandarkar, S. Jain, and A. Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control," in *Proc. of the PFLDNet'05*, Feb. 2005.
[15] B. Wydrowski and M. Zukerman, "MaxNet: A Congestion Control Architecture for Maxmin Fairness," *IEEE Comm. Letters*, vol. 6, no. 11, pp. 512 – 514, Nov. 2002.

[16] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," in *IETF RFC 4782*, Jan. 2007.
[17] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proc. ACM SIGCOMM, 2002*, Aug. 2002.
[18] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One More Bit Is Enough," in *Proc. ACM SIGCOMM, 2005*, Aug. 2005.
[19] 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*, vol. 5, no. 6, pp. 756–769, Dec 1997.
[20] J. Chen, M. Gerla, Y. Z. Lee, and M. Sanadidi, "TCP with Variance Control for Multihop IEEE 802.11 Wireless Networks," in *Proc. of the IEEE MILCOM, 2006*, Oct. 2006.
[21] A. Bakre and B. R. Badrinath, "I-TCP: indirect TCP for mobile hosts," in *Proc. of 15th International Conference on Distributed Computing Systems(ICDCS)*, Vancouver, BC, May 1995, pp. 136 – 143.
[22] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 5, pp. 19–42, Oct. 1997.
[23] E. Ayanoglu, S. P. T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: a link-layer protocol for wireless networks," *ACM Wireless Networks*, vol. 1, no. 1, pp. 47–60, Feb. 1995.
[24] C. Parsa and J. Garcia-Luna-Aceves, "Improving TCP performance over wireless networks at the link layer," *ACM Mobile Networks and Applications*, vol. 5, no. 1, pp. 57–71, Mar. 2000.
[25] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bhargavan, "WTCP: a reliable transport protocol for wireless wide-area networks," in *Proc. ACM MobiCom*, Seattle, WA, Aug. 1999.
[26] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang, "TCP Westwood: end-to-end congestion control for wired/wireless networks," *Wireless Networks (WINET)*, vol. 8, no. 5, pp. 467–479, Sep. 2002.
[27] H. Balakrishnan and R. Katz, "Explicit Loss Notification and Wireless Web Performance," in *Proc. of the IEEE GLOBECOM, 1998*, Nov. 1998.
[28] G. Buchholcz, A. Gricser, T. Ziegler, and T. V. Do, "Explicit Loss Notification to Improve TCP Performance over Wireless Networks," in *Proc. of the 6th IEEE High-Speed Networks and Multimedia Communications (HSNMC), 2003*, 2003.
[29] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995.
[30] C. Barakat and A. A. Fawal, "Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic," *Performance Evaluation Journal*, vol. 57, no. 4, pp. 423–500, Aug. 2004.
[31] C. Barakat and E. Altman, "Bandwidth tradeoff between TCP and link-level FEC," *Computer Networks*, vol. 39, no. 2, pp. 133–150, Jun. 2002.
[32] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan, "LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels," in *Proc. of the IEEE International Workshop on Quality of Service*, 2005.
[33] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Transaction on Networks*, vol. 9, no. 3, pp. 307–321, Jun. 2001.
[34] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "Selective acknowledgment options," in *RFC-2018*, Oct. 1996.
[35] S. Keshav and S. P. Morgan, "SMART Retransmission: Performance with Overload and Random Losses," in *Proc. INFOCOM 1997*, Apr. 1997.
[36] M. C. Chan and R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation," in *Proc. of the ACM Mobicom 02.*, 2002.
[37] T. R. Henderson and R. H. Katz, "TCP performance over satellite channels," *UCP Computer Science Technical Report*, Dec. 1999.
[38] M. J. Zukoski and R. Ramirez, "A transport protocol for space communications," *Network and Communications: The Edge Newsletter*, Nov. 1998.
[39] A. Chockalingam, M. Zorzi, and R. Rao, "Performance of TCP on Wireless Fading Links with Memory," in *Proc. IEEE ICC*, 1998.
[40] M. Zorzi, A. Chockalingam, and R. Rao, "Throughput Analysis of TCP on Channels with Memory," *IEEE JSAC*, vol. 18, pp. 1289 – 1300, Jul. 2000.
[41] F. Abrantes and M. Ricardo, "A simulation study of xcp-b performance in wireless multi-hop networks," in *Proc. of the 3rd ACM workshop on QoS and security for wireless and mobile networks*, 2007.
[42] Y. Zhang and T. Henderson, "An Implementation and Experimental Study of the eXplicit Control Protocol (XCP)," in *Proc. IEEE INFOCOM, 2005*, Mar. 2005.
[43] S. Athuraliya, "A Note on Parameter Values of REM with Reno-like Algorithms," Mar. 2002, available at . http://netlab.caltech.edu/pub/papers/REMparameter.pdf.
[44] R. Russell and H. Welte, "Linux netfilter Hacking HOWTO," Jul. 2002, available at http://www.netfilter.org/documentation/HOWTO//netfilter-hacking-HOWTO.html.
[45] S. Hemminger, "Network Emulation with NetEm," in *Proc. LCA, 2005*, Apr. 2005.
[46] J. Mahdavi, "Enabling high performance data transfers on hosts," *technical note, Pittsburgh Supercomputing Center*, Dec. 1997, available at http://www.psc.edu/networking/perf_tune.html.
[47] -, "TCP Tuning Guide," Nov. 2005, distributed Systems Department, Available at http://dsd.lbl.gov/TCP-tuning/TCP-tuning.html.
[48] H. Yousefi'zadeh, H. Jafarkhani, and M. Moshfeghi, "Power Optimization of Wireless Media Systems with Space-Time Block Codes," *IEEE Trans. Image Processing*, vol. 7, no. 13, pp. 873 – 884, Jul. 2004.
[49] L. Zheng, H. Yousefi'zadeh, and H. Jafarkhani, "Resource Allocation in Fading Wireless Ad-Hoc Networks with Temporally Correlated Loss," in *Proc. IEEE WCNC*, Mar. 2004.
[50] H. Yousefi'zadeh, "A Constrained Resource Allocation Study for LOS MIMO Fading Ad-Hoc Networks," in *Proc. IEEE WCNC*, Apr. 2006.
[51] H. Jafarkhani, P. Ligdas, and N. Farvardin, "Adaptive Rate Allocation in a Joint Source-Channel Coding Framework for Wireless Channels," in *Proc. IEEE VTC*, Apr. 1996.
[52] S. Alamouti, "A Simple Transmitter Diversity Scheme for Wireless Communications," *IEEE JSAC*, vol. 16, pp. 1451 – 1458, Oct. 1998.
[53] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-Time Block Coding from Orthogonal Designs," *IEEE Trans. Information Theory*, vol. 45, no. 5, pp. 1456 – 1467, Jul. 1999.
[54] -, "TIA/EIA/IS-99, Data Services Option Standard for Wideband Spread Spectrum Digital Cellular System," Jul. 1995, telecommunications Industry Association, Available at http://electronics.ihs.com/document/abstract/ EZGW-CAAAAAAAAAAA.
[55] ——, "WiMax Protocol Specification," available at http://ieee802.org/16/published.html.
[56] H. Yousefi'zadeh and A. Habibi, "A Performance Comparison Study of End-to-End Congestion Control Protocols over MIMO Fading Channels," in *Proc. of the IEEE MILCOM, 2006*, Oct. 2006.



Xiaolong Li currently is a Postdoc researcher at the Department of EECS at UC, Irvine. He received his Ph.D. degree at the same department in 2009. He received his MS degree in Computer Science and Engineering from the University of Notre Dame in 2006. His research interests are in the area of wireless congestion control and wireless routing. Xiaolong is a student member of IEEE.

Homayoun Yousefi'zadeh is an Associate Adjunct Professor at the Department of EECS at UC, Irvine. He also holds a Consulting Scientist position at the Boeing Company. Most recently, he was the CTO of TierFleet, a Senior Technical and Business Manager at Procom Technology, and a Technical Consultant at NEC Electronics. He is the inventor of several US patents and the author of many scholarly published articles. Dr. Yousefi'zadeh is with the editorial board of two IEEE journals, has served as the lead guest editor of IEEE JSTSP the issue of April 2008, and has been with the TPC of various IEEE and ACM conferences. He has also served as the founding Chairperson of systems' management workgroup of the Storage Networking Industry Association, as a member of the scientific advisory board of Integrated Media Services Center at the University of Southern of California (USC), and as a member of American Management Association. He received the Ph.D. degree from the Dept. of EE-Systems at USC in 1997.