A TCP with Guaranteed Performance in Networks with Dynamic Congestion and Random Wireless Losses

Stefan Schmid, Roger Wattenhofer
{schmiste, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory
ETH Zurich, CH-8092 Zurich, Switzerland

***Abstract*— It is well-known that TCP performs poorly in the presence of packet losses happening for reasons other than Internet congestion. One increasingly important source of such errors are wireless links. In this paper, networks are studied where the congestion—and thus the bandwidth available to a host—varies over time, and where in addition packets are lost at random. First, we propose a network comprising both dynamically varying congestion and random errors. Second, our model is extended with network calculus concepts in order to study bursty changes of congestion. Finally, we present the transfer protocol TCP "Wichita" (TCPW) which achieves a provable worst-case performance in this environment.**

I. Introduction

TCP is the prevailing transport protocol of the Internet. It features a variety of properties such as reliable data transport or flow and congestion control. Particularly, the TCP congestion control mechanism is seen as a cornerstone of the functionality of todays Internet, as it restricts hosts to transmit their data with the maximum rate that can be handled by the Internet backbone.

The approach taken by TCP is to have each sender limit the transmission rate as a function of the perceived network congestion: If there is little congestion on the path between the sender and the destination, the sending rate will be increased, and vice versa.

In order to control the transmission rate, TCP employs a *window-based scheme*. Thereby, each sender maintains a send buffer called the *congestion window*. Basically, this window contains the packets which have already been sent, but which have not yet been acknowledged by the receiver. Therefore, the size of this window indirectly limits the sending rate: Roughly, at the beginning of every round trip time (RTT), the sender can send a window full of packets into the connection. At the end of the RTT, the sender receives the acknowledgements (ACKs) and can fill the window with new data.

In order to avoid congestion, a TCP sender seeks to change the window's size—and thus the sending rate—according to the presently available bandwidth. However, the only feedback a sender gets about the current state of the Internet is the number of packet losses it experiences, and hence it is clear that the size of the congestion window can only approximately reflect the current bandwidth. Most TCP versions work as follows: As long as no packets are lost, the window is increased multiplicatively up to a certain *threshold*, after which it is increased linearly. On the other hand, if a packet is lost, the sending window is cut in half. This so-called *Additive Increase - Multiplicative Decrease* (AIMD) strategy has proved to be an effective means to prevent congestion collapses as they happened in the 1980s [3]: If routers start dropping packets, the different senders reduce their sending rates multiplicatively and alleviate the load in a collaborative manner.

However, the Internet has undergone many changes over the last 10 years, both in terms of its size and of its composition. Of prime importance is the evolution of wireless networking technology, which has led to a large number of mobile Internet users. A crucial difference between wireline and wireless networks is the presence of random wireless losses in the latter. Specifically, the effective bit error rates in wireless networks are significantly higher than in wireline networks: e.g., due to higher cochannel interference, host mobility, multipath fading, or disconnections because of coverage limitations, etc. [28]

The higher packet error rates in wireless networks inherently lower the performance experienced by connections traversing such networks. Unfortunately, however, they cause an even more severe degradation in the throughput of connections using TCP as the transport protocol: TCP does not have any mechanisms to differentiate between congestion-induced and other losses. Consequently, when a TCP sender experiences wireless losses, it wrongly interprets such losses as congestion losses, and cuts down its window and thus the throughput of the connection.

In this paper, we introduce a network model which comprises both congestion and random losses. Thereby, the bandwidth available to a host changes over time, for instance due to the dynamic bandwidth demand of other hosts. We will assume a conservative perspective and consider *worst-case changes* of the available bandwidth. In particular, we apply concepts of *network calculus* [19] and allow for *bursty* dynamics. Moreover, in our model, packets are lost at random due to wireless links, or due to intermittent faults in hardware elements (e.g., Ethernet, FDDI adapters, etc.) or incorrect handling of arriving packets by routers. We then look at a single TCP sender and investigate algorithms which achieve a provable throughput in this environment.

The rest of this paper is organized as follows. Related work is reviewed in Section II. In Section III, we describe the model and introduce our definitions. Section IV presents

and analyzes our transfer protocol TCP Wichita (TCPW). We extend our model in Section V and allow for bursty changes of congestion. The average-case performance of TCPW is studied by simulation in Section VI. Finally, the paper is concluded in Section VII.

II. Related Work

TCP lies at the heart of today's Internet, and still many aspects of TCP are subject to active research, e.g. [1], [2], [10], [11], [12], [13], [16], [22]. Unfortunately, we can only review some of these contributions. For an overview of TCP, we refer the reader to [17], or—for a more technical introduction—to [27].

With the increasing number of mobile and wireless devices populating the Internet, researchers have started studying the impact of wireless networking technologies on the different layers of the protocol stack, including physical, data-link, medium-access, network, transport, and application layers [14], [20], [21], [23], [26], [29].

One of the first papers to address TCP performance under random losses is due to Lakshman and Madhow [18]. The authors investigate a model where each packet is lost independently with probability q. It is shown that—even in the absence of congestion and if the available bandwidth is infinite—the TCP's throughput is roughly limited by $1/\sqrt{q}$.

Many solutions have been proposed in literature to improve TCP's performance over wireless links. For a good overview on the topic, see [5][24][28]. These approaches fall into three categories [28]: *Link layer approaches* which enhance TCP's performance without requiring any change at the transport layer (e.g., the snoop protocol [6]); *indirect approaches* which also mask the characteristics of the wireless portion of the connection, but split the TCP connection at the base station (e.g., I-TCP [4]); and *end-to-end approaches* which require changes of the protocol stack at both the sender and the receiver (e.g., WTCP [26]).

Our approach is different from the papers mentioned above as we study an explicit network model. Our model is based on the work by Karp et al. [15] who investigate several optimization problems for Internet congestion. We also consider an enhancement of this model: in [25], a novel model inspired by *network calculus* concepts [19] for the congestion's dynamics has been introduced which allows for *bursty* changes of the available bandwidth. We extend the wireline approaches of [15] and [25] by taking random losses into account besides dynamic congestion, seeking to shed light on the wireless case.

III. Model

In this paper, it is assumed that the routers in the Internet drop packets when the congestion increases, and that some additional packets are lost at random for other reasons, for instance due to a high bit error rate on certain wireless links. For such a network, we consider the problem of regulating the sending rate of a unicast flow from one host to another such that the throughput is maximized.

First, consider the following static model where the bandwidth available to a flow is constant, and where a host aims at choosing its sending rate x in such a way that its throughput is maximized. If there was only a constant loss probability (*independently* of the sending rate), the best policy would clearly be to send at the highest possible rate given the host's hardware resources and network connection. However, if we assume the losses due to congestion to increase with the host's sending rate, this may no longer be the best solution, but there may be one or more optimal *operation points*. Let f be a function that defines, for each sending rate x, a corresponding packet loss probability $f(x)$. Then, if the loss probability increases monotonically— i.e., the second derivative of f is positive: $\forall x : \delta^2 f(x)/\delta^2 x > 0$—there is exactly one optimal operation point. It is easy to devise transport protocols that find such a point quickly by probing different sending rates; for example, one could make use of *Brent's algorithm* [9] which has a logarithmic execution time.

Henceforth, however, only *dynamic* models are considered where the bandwidth changes over time. This paper assumes that the bandwidth available to the flow fluctuates according to the varying requirements for bandwidth of other competing flows rather than on the flow considered. Thus, in our model, the dynamics of the congestion is given externally. Additionally, there are random packet losses in the network because of wireless links.

We consider a synchronous model where time is divided into an infinite number of *rounds*. The duration of a round is approximately given by the round-trip-time (RTT) between the sending and the receiving host of the flow. At the beginning of each round, the sender transmits all the data in the sending window. By the end of the round, the sender has got the acknowledgments (ACKs) from the receiver and can detect whether there have been any packet losses. Based on this information, it will then increase or reduce the size of its sending window, which determines the sending rate of the next round. Recall that the sender is not able to distinguish whether packets have been lost because of congestion or because of wireless errors.

Basically, this setting can be regarded as a *game*: The congestion changes are controlled by an adversary ADV which, in every round t, selects the available bandwidth u_t. The task of a transfer protocol ALG is to decide the rate x_t with which to send in round t such that the throughput is maximal. Thereby, ALG has no knowledge about u_t and not even about the old values u_s for $s < t$. All it knows is its own sending rates x_s for all $s < t$, and whether there have been packet losses in these rounds.

In order to evaluate the performance of a transfer protocol in such a game, cost or gain functions are needed. The cost function used in this paper is described next. If in round t, ALG sends with a rate x_t, and if there are no losses, we say that the *gain* of ALG in round t is x_t. That is, ALG has a gain x_t if there is no congestion in round t and if there are no random errors.

We now look at the two sources for packet losses in more detail. If in round t the available bandwidth is smaller than the sending rate of the transfer protocol, i.e., $u_t < x_t$, ALG cannot

transmit its packets due to congestion and the gain is 0 in round t. Moreover, with probability p, a channel is unavailable to the flow in a round t regardless of the sending rate x_t, and ALG cannot increase its gain either. Thus, we have

$$gain_{ALG}(x_t, u_t) := \begin{cases} x_t, \text{ if } x_t \leq u_t \textit{ and} \text{ no random error} \\ 0, \text{ otherwise} \end{cases}$$

At first sight, this cost function seems to be quite severe: For instance, there is no gain at all in congested rounds and *all* data is lost. Although there would be many reasonable alternative cost functions, this solution takes into account that a transfer protocol usually has to start a retransmission mechanism after a time-out and therefore experiences a certain overhead. Note that our cost function also incorporates an *opportunity cost*: If a transfer protocol conservatively chooses too low sending rates, i.e., $x_t << u_t$, the gain is small as well. The goal of ALG is therefore to use sending rates x_t which are always close to, but never above u_t.

Observe that with the model described so far, in the worst case, no algorithm ALG can achieve a positive gain: After ALG has decided about its rate x_t, a powerful clairvoyant adversary can always choose the available bandwidth to be slightly smaller than this rate, i.e., $u_t = x_t - \varepsilon$ for some arbitrary small $\varepsilon > 0$. Thus, the sender suffers congestion in every round and the total gain of ALG is 0. However, since congestion is determined by the policies of the competing flows, and since the flows normally do not change their rates too abruptly, also congestion does not change too drastically from one moment to the other. Therefore, it is reasonable to assume certain constraints on the bandwidth's dynamics. In our case, we assume that the bandwidth can only increase by a certain *percentage* per round; we do not need any bound for the bandwidth's decrease. Formally, the adversary ADV is allowed to choose u_{t+1} from the interval $[0, u_t \cdot \mu]$. In Section V, we will extend this model with *bursts*.

Having defined the gain of a transfer protocol ALG and the worst-case congestion changes controlled by ADV, it is possible to evaluate ALG's performance formally. However, quantifying the gain by an absolute number itself does not provide many insights. The approach taken here is to compare ALG's gain to the gain the best transfer protocol OPT would have achieved for the same input. Thereby, OPT knows the available bandwidth in advance, i.e., it can always choose $x_t := u_t$ which clearly yields the optimal throughput. Hence, OPT is inherently more powerful than ALG: While ALG is an *online algorithm* and has to select x_t without the knowledge of u_s for $s \geq t$ (and only based on binary feedbacks about u_s for $s < t$), OPT can solve the optimization problem *offline*. Concretely, OPT only experiences random losses, but no opportunity or congestion costs, and thus its gain in round t is given by

$$gain_{OPT}(x_t, u_t) := \begin{cases} u_t, \text{ if there is no random error} \\ 0, \text{ otherwise} \end{cases}$$

To quantify ALG's performance, we consider the *competitive ratio* [8], i.e., the worst-case ratio (over all possible executions) of the total gain achieved by OPT divided by the total gain of ALG (cf Definition 1). The objective of ALG is to minimize this ratio.

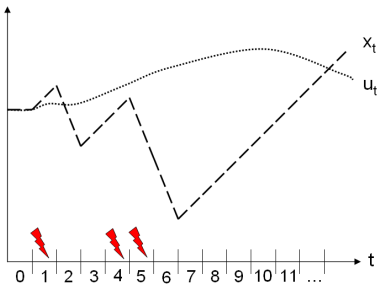

Fig. 1. Sample sequence: The available bandwidth u_t changes over time in a limited manner. The goal of ALG is to choose x_t slightly smaller than u_t. The first round is a success ($u_0 = x_0$), and ALG increases its sending rate. In the next round however, there is no gain due to congestion. In rounds 4 and 5 there are random errors, and the transfer protocol shown in the figure wrongfully reduces its sending rate although the potential bandwidth is large.

Definition 1 (Competitive Ratio ρ): An algorithm ALG is *ρ-competitive* compared to an optimal offline algorithm OPT if for all input sequences I, it holds that

$$gain_{OPT}(I) \leq \rho \cdot gain_{ALG}(I) + k$$

where k is a constant independent of the input.

In the following, we assume that there is no loss in round 0 and $x_0 = u_0$. Figure 1 depicts an example: The available bandwidth changes multiplicatively over time, and ALG can increase its gain only if $x_t \leq u_t$ (e.g., in round 1 where $x_0 = u_0$). Moreover, there are random errors once in a while, for example in rounds 4 and 5. The algorithm shown in the figure decreases its sending rate in rounds 5 and 6, since it has wrongfully interpreted the losses as congestion losses.

Before concluding this section, we introduce some definitions. In the following, we will distinguish between *good and bad rounds*.

*Definition 2 (**Good and Bad Rounds**):* Rounds where the probing rate x_t is smaller or equal the available bandwidth u_t are called *good rounds*. A round t where $x_t > u_t$ is called a *bad round*.

Moreover, we will refer to rounds where random losses happen as *loss rounds*.

*Definition 3 (**Loss Rounds**):* Rounds where neither ALG nor OPT can transmit any data because of random errors are called *loss rounds*.

Note that by our definitions, ALG may not be able to transmit any data in good rounds, because there are random losses. However, in such a round OPT cannot increase its gain either. This motivates the definition of successful and non-successful rounds.

*Definition 4 (**Successful/Non-successful Rounds**):* Good rounds where there are no losses are called *successful*, all other rounds are called *non-successful*.

Thus, successful rounds are exactly those rounds where ALG *successfully* transmits data and hence increases its gain.

IV. A COMPETITIVE TCP

Having introduced the model, we now present and analyze our transfer algorithm *TCP Wichita*, short *TCPW*.[1] Basically, the idea of TCPW is to compensate wrongful rate reductions due to random errors by increasing the rate more after successful rounds. TCPW is described in Algorithm 1.

Algorithm 1 TCP Wichita (TCPW)

```
(* Round t *)
if round t − 1 successful then
    x_t := 2μ² · x_{t−1};
else
    x_t := x_{t−1}/2;
end if
```

The sending rate of TCPW in round t solely depends on the binary feedback as to whether the previous round was successful or not. Moreover, TCPW cuts the sending window in half after a loss. This is similar to many existing TCP versions. However, after successful rounds, TCPW increases the sending rate also multiplicatively. This increase is larger than μ—the maximal change factor of the available bandwidth—in order to compensate wrong reductions in the past.

So how does TCPW's perform in the dynamic network introduced in Section III? First, we compute the missed bandwidth in rounds where the sending rate of TCPW is too high, i.e., higher than the available bandwidth (case $x_t > u_t$).

Lemma 4.1: In the bad rounds, the gain of an optimal transfer protocol OPT is at most a factor $4\mu^2$ larger than the gain TCPW achieves in the good rounds, i.e.,

$$gain_{OPT}(bad) \leq 4\mu^2 \cdot gain_{TCPW}(good).$$

Proof: First, assume that there are no loss rounds, i.e., good rounds are also successful rounds, and vice versa. After having shown that the claim holds in this case, we will tackle the general case.

Consider the last good round t before a sequence of bad rounds t, $t+1$, ..., $t+\tau$ for some (maybe infinite) $\tau > 0$. We have $x_t \leq u_t$, and by the definition of TCPW, $x_{t+1} = 2\mu^2 x_t > u_{t+1}$, $x_{t+2} = x_{t+1}/2 = \mu^2 x_t > u_{t+2}$, $x_{t+3} = x_{t+2}/2 = \mu^2/2 x_t > u_{t+3}$, etc.

Thus, OPT has to reduce its sending rate geometrically, and the gain in a sequence of bad rounds after a good round t is limited by a factor $4\mu^2$ times the last successful transmission of TCPW:

$$2\mu^2 x_t \cdot \sum_{i=0}^{\infty} 2^{-i} = 4\mu^2 x_t.$$

It remains to study the case of random losses. We consider the last good *and successful* round t before a sequence of non-successful rounds $t+1$, $t+2$, ..., $t+\tau$ for some $\tau > 0$. TCPW does not distinguish errors due to congestion from random errors and reduces the probing rate geometrically as observed above. But OPT can never send with a rate larger than x_i for $i \in [t+1, t+\tau]$, because in non-successful good rounds, it also experiences the random loss, and in bad rounds its sending rate is upper bounded by x_i for $i \in [t+1, t+\tau]$. ■

[1]The *W* in Wichita stands for wireless. Moreover, in the tradition of TCP names, Wichita is also a city (in Kansas, USA).

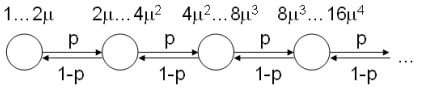

Fig. 2. In the first good round, the competitive ratio of TCPW is at most 2μ. In general, TCPW catches up by a factor of at least 2μ with probability $1-p$, and loses a factor of at most 2μ with probability p.

The study of the competitive ratio of good rounds is more involved. By our definition, the sending rate of TCPW is lower than the available bandwidth in these rounds. Moreover, TCPW reduces its sending rate even further after loss rounds, because it assumes the loss to be due to congestion.

By modeling the situation as a Markov chain (for an introduction, cf [7]), it can be seen that if the random loss probability is sufficiently small compared to the bandwidth changes, the expected competitive ratio in good rounds is small as well, i.e., TCPW sends at rates that are almost as high as the optimal sending rate.

Lemma 4.2: If $\mu \leq \frac{1-p}{4p}$, the expected gain of TCPW in good rounds is at most a factor 4μ smaller than the expected gain of an optimal offline algorithm OPT, i.e.,

$$E[gain_{OPT}(good)] \leq 4\mu \cdot E[gain_{TCPW}(good)].$$

Proof: Consider the first good round t after a sequence of bad rounds. Since $x_{t-1} > u_{t-1}$, $x_t = x_{t-1}/2$, and $u_t \leq \mu u_{t-1}$, the competitive ratio is at most 2μ in round t. However, with probability p, round t is non-successful, and TCPW cuts its sending rate in half while the actual bandwidth increases by a factor of at most μ, yielding a new potential competitive ratio of $4\mu^2$. On the other hand, if there is no random loss, TCPW catches up by a factor of at least 2μ, and the competitive ratio is 1 (or round $t+1$ is bad). Now consider round $t+1$ and assume that there was a random loss in round t, i.e., the TCPW's sending rate is at most a factor $4\mu^2$ smaller than the one of OPT. With probability p, there is yet another random loss, and the competitive ratio in round $t+2$ can be as large as $8\mu^3$. However, with probability $1-p$, TCPW catches up by a factor of at least 2μ since the available bandwidth increases at most by a factor μ per round. And so on.

Thus, a competitive ratio of good rounds can be modeled with the infinite, 1-dimensional Markov chain shown in Figure 2: With probability p, TCPW loses a factor of at most 2μ in each round compared to the real bandwidth, and catches up by a factor at least 2μ with probability $1-p$.

Let us refer to the states in the Markov chain of Figure 2—from left to right—as s_0, s_1, etc., and let $\pi(s_i)$ be the steady-state probability of state s_i. We have $p \cdot \pi(s_0) = (1-p) \cdot \pi(s_1)$, and hence $\pi(s_1) = p/(1-p) \cdot \pi(s_0)$, $p \cdot \pi(s_1) = (1-p) \cdot \pi(s_2)$ and hence $\pi(s_2) = p/(1-p) \cdot \pi(s_1) = p^2/(1-p)^2 \cdot \pi(s_0)$, etc. By induction, it follows that, for $i > 0$,

$$\pi(s_i) = \left(\frac{p}{1-p}\right)^i \cdot \pi(s_0)$$

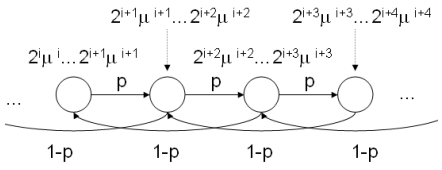

Fig. 3. Markov chain of a modified TCPW.

Using the fact that the steady-state probabilities must sum up to 1, it is possible to compute $\pi(s_0)$:

$$\pi(s_0)\sum_{i=0}^{\infty}\left(\frac{p}{1-p}\right)^i \stackrel{!}{=} 1 \Rightarrow \pi(s_0) = \frac{2p-1}{p-1}.$$

Knowing the probability distribution, the expected competitive ratio in the good rounds becomes

$$E[\rho_{good}] \leq \sum_{i=0}^{\infty}\pi(s_i)(2\mu)^{i+1} = 2\mu\pi(s_0)\cdot\sum_{i=0}^{\infty}\left(\frac{2\mu p}{1-p}\right)^i.$$

As stated in the lemma, we assume that $\mu \leq \frac{1-p}{4p}$, and the claim follows. ∎

Lemma 4.2 requires that μ and p adhere to a certain constraint, that is, if the random loss probability p is large, the bandwidth can only change by small factors μ. Although it is possible to alleviate this restriction by using other algorithms than TCPW, it is clear that p and μ must always be correlated in some way. Moreover, the performance of such alternative transfer protocols—which can be analyzed with the same techniques as introduced for TCPW—can be significantly worse than TCPW. For example, the Markov chain of a modified TCPW which increases its bandwidth by a factor of $4\mu^2$ after successful rounds but which still halves the window size is depicted in Figure 3. The constraints for μ and p are slacker in this case, but the competitive ratio is worse.

Finally, we can combine Lemmata 4.1 and 4.2, and get the following theorem.

Theorem 4.3: The expected competitive ratio of TCP Wichita (TCPW) is at most $4(\mu^2+\mu)$ if $\mu \leq \frac{1-p}{4p}$.

Proof: According to Lemma 4.1, an optimal offline algorithm OPT can increase its gain by a factor at most $4\mu^2$ in the bad rounds following a successful round. The expected competitive ratio of good rounds on the other hand is at most 4μ (Lemma 4.2). Since we assume that $x_0 = u_0$, there is a good round before every sequence of bad rounds, and the claim follows. ∎

V. Congestion with Bursts

It has been pointed out by Willinger et al. [30] that the nature of network traffic is inherently self-similar and *bursty*. In this section, we extend our framework to allow for burst-like changes of the available bandwidth as proposed in [25].

We do not give a complete introduction to network calculus here, but refer the interested reader to the book by Le Boudec and Thiran [19]. However, we will quickly review the concept of *leaky-bucket arrival curves*. In Section V-B, it is then shown how these arrival curves are adapted to model bursty congestion.

A. Leaky-Bucket Arrival Curves in Network Calculus

Network calculus introduces the notion of *arrival curves* which provide some deterministic limitation to the network traffic sent by sources. The idea is that if the data flows indeed adhere to these limitations, it is possible to make statements about the deterministic behavior of networks, for instance about the queue lengths at the different routers.

Arrival curves are defined as follows. Let R be a data flow, and $R(t)$ the total number of bits R has sent until time t. Let α be an increasing function defined for all times $t \geq 0$. R has an arrival curve α if and only if for all $s \leq t$:

$$R(t) - R(s) \leq \alpha(t-s)$$

Thus, the total number of bits sent until time t by flow R can never exceed the amount of bits sent by R until some time s plus $\alpha(t-s)$. As an example, we look at a so-called *leaky bucket arrival curve* which is defined as $\alpha(t) = c_1 t + c_2$ for some non-negative constants c_1, c_2. Figure 4 visualizes the constraints imposed upon a flow R by such an arrival curve: The total number of bits sent can increase by c_2 at once and with a rate c_1 over time, unless there is a conflict with a constraint from a previous round. Informally, the total number of bits must always be less or equal the minimum constraint that arises if the curve α is attached (or added) at all points of $R(t)$.

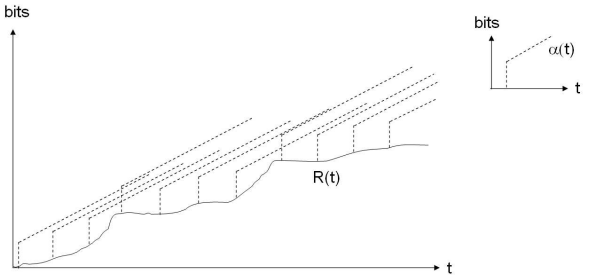

Fig. 4. Leaky bucket arrival curve: The number of bits sent by flow R must never exceed the constraints from earlier times (dashed lines), i.e., $\forall s \leq t$: $R(t) \leq R(s) + \alpha(t-s)$.

Note that such an arrival curve incorporates a limited form of *amortization*: If flow R only sends a few bits for several rounds, the constraints of earlier rounds get weaker and allow R to send up to c_2 bits at once in some later round. In this paper, we port this property of arrival curves to the field of congestion control.

B. Bursty Congestion Changes

How can the concepts of Section V-A be used to model dynamic congestion? The idea is to bound the adversarial bandwidth changes by an arrival curve. Concretely, the new adversary ADV can again increase the available bandwidth by a factor of μ in every round. However, it may also decide to accumulate some power in some rounds and then make more abrupt changes in later rounds. The goal of a transfer protocol ALG is to compete with an optimal offline algorithm OPT which knows all the bandwidth changes in advance.

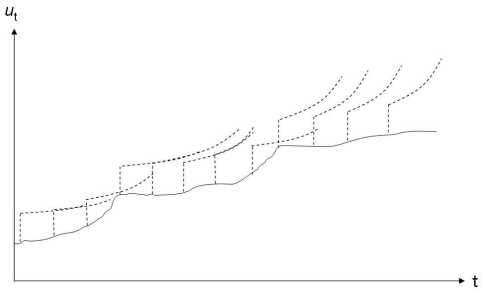

Fig. 5. Visualization for the case $\forall t : u_{t+1} \geq u_t$, i.e., for an adversary which never decreases the bandwidth. The bandwidth can increase multiplicatively in every round, but it must always adhere to the constraints imposed by the previous rounds (dashed lines).

ADV has two parameters: A *rate* $\mu \geq 1$ and *maximum burst factor* $B \geq 1$. In every round, the available bandwidth u_t varies according to these parameters in a multiplicative manner. That is, ADV selects u_{t+1} from the interval $u_{t+1} \in [\frac{u_t}{\beta_t \mu}, u_t \cdot \beta_t \cdot \mu]$. Hence, the available bandwidth can change by a factor of at most $\beta_t \mu$, where β_t is called the *burst factor at time* t. It is explained next.

On average, the available bandwidth changes by a factor at most μ per round. However, there can be times of only small changes, but consequently, the bandwidth can change by factors larger than μ in later rounds. At the beginning, β_t equals B, i.e., $\beta_0 = B$. For $t > 0$, the burst factor β_t is computed depending on β_{t-1} and the actual bandwidth change c_{t-1} that took place in round $t-1$. More precisely,

$$\beta_t = \min\{B, \beta_{t-1} \frac{\mu}{c_{t-1}}\}$$

where

$$c_t := \begin{cases} \frac{u_{t+1}}{u_t} & \text{, if } u_{t+1} > u_t \\ \frac{u_t}{u_{t+1}} & \text{, otherwise} \end{cases}$$

This means that if the available bandwidth has changed by a factor smaller than μ in round t, i.e., $c_t < \mu$, the burst factor will *increase* multiplicatively by $\frac{\mu}{c_t}$, and hence the potential bandwidth change is larger in the next round (and vice versa if $c_t > \mu$). In other words, the adversary is allowed to save power for forthcoming rounds. This amortization is however limited as β_t is upper bounded by B for all rounds t. Also note that $\beta_t \geq 1$ always holds, because $c_t \leq \mu \beta_t$ by the definition of ADV.

Figure 5 visualizes ADV for the case $\forall t : u_{t+1} \geq u_t$, i.e., for an ADV than only increases (but never decreases) bandwidth: The bandwidth can rise by a factor of μB in every round, unless it conflicts with a constraint from a previous round, i.e., $\forall t : u_t \leq \min_{i \in \{0,\ldots,t-1\}} \{u_i \cdot B \cdot \mu^{t-i}\}$.

In the following, we investigate the performance of TCPW. First, observe that the missed gain by TCPW in bad rounds is still upper bounded by $4\mu^2$ times the gain of TCPW in good rounds (cf Lemma 4.1), because the same geometric series argument applies. For the competitive ratio of good rounds, the analysis is similar as well: Over-pessimistically, a Markov chain as depicted in Figure 6 can be assumed. It follows that

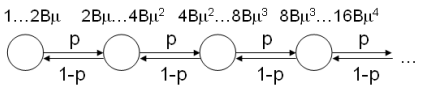

Fig. 6. Markov chain for expected competitive ratio for the case of bursty congestion changes.

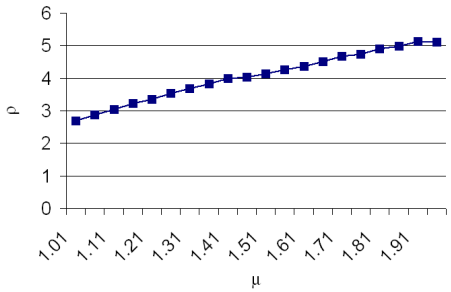

Fig. 7. Competitive ratio of TCPW for random error probability 10% as a function of μ.

the expected competitive ratio is upper bounded by $4(\mu + B\mu^2)$, if $\mu \leq \frac{1-p}{4 \cdot B \cdot p}$.

Theorem 5.1: In the case of worst-case bandwidth changes with bursts, the expected competitive ratio of TCP Wichita (TCPW) is at most $4(\mu + B\mu^2)$ if $\mu \leq \frac{1-p}{4 \cdot B \cdot p}$.

VI. Simulation

Although the main focus of this paper is on worst-case performance, in this section, we want to complete the picture by giving some simulations of the average case behavior of TCPW for random (rather than adversarial) bandwidth changes.

A. Without Congestion Bursts

In Section IV, it has been shown that TCPW achieves an expected competitive ratio of at most $4(\mu^2 + \mu)$ if the congestion changes in a worst-case manner and without bursts, and if $\mu \leq \frac{1-p}{4 \cdot p}$ (cf Theorem 4.3). While this throughput is acceptable, our analysis was pessimistic in several respects. For instance, it is not possible that there is a sequence of bad rounds right after a good round having a high competitive ratio.

So we may ask: What is the "competitive ratio" if the bandwidth changes *randomly*? In the following, we select a change factor c_t uniformly distributed from the interval $c_t \in_R [1, \mu]$. We toss a coin and increase or decrease the bandwidth with probability .5 by a factor c_t accordingly.

In experimental studies [14], [26], packet error rates ranging from 1% in microcell wireless networks up to 10% in macrocell networks have been reported. Moreover, it is reasonable to assume that the bandwidth does also change around 10% or less per round. In Figure 7, the competitive ratio of TCPW is shown for an error probability $p = .1$ as a function of the bandwidth dynamics μ. Apparently, the ratio is near linear in this range. Moreover, as expected, it is much smaller than the bound given in Theorem 4.3.

In a second experiment, we study the impact of the error probability p: Figure 8 plots the competitive ratio as a function of p for bandwidth changes of approximately 10% per round,

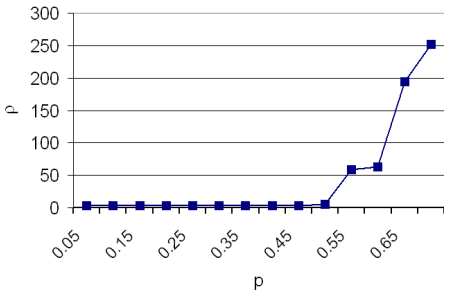

Fig. 8. Competitive ratio of TCPW for $\mu = 1.1$ as a function of p.

i.e., $\mu = 1.1$: After the ratio has been almost constant in the beginning, it increases abruptly if a certain threshold is exceeded. This also highlights the existence of a relationship between μ and p (e.g., $\mu \leq \frac{1-p}{4p}$ in Theorem 4.3).

If the bandwidth changes randomly and if μ and p are small, TCPW performs better if it increases the bandwidth less after successful rounds, for example by a factor $1.1 \cdot \mu^2$. In the following, we will assume such a less aggressive version of TCP Wichita.

We have compared TCP Wichita to a simplified version of the well-known *TCP Tahoe* [17] (cf Algorithm 2). At the beginning, TCP Tahoe increases its sending rate by a factor of two (*slow-start phase*), but changes to linear increase after a certain threshold (*congestion avoidance phase*) is met. If there is an error, the new threshold becomes half the sending rate at which the loss has occurred, and the sending rate is set back to 1. Figure 9 shows a sample execution for $p = .1$ and $\mu = 1.1$. Moreover, we assume that $x_0 = u_0$ and that at the beginning, $threshold := u_0/2$.

Algorithm 2 TCP Tahoe

```
1:  (* Round t *)
2:  if round t − 1 successful then
3:      if size(window) < threshold then
4:          x_t := 2 · x_{t−1};
5:      else
6:          x_t := x_{t−1} + 1;
7:      end if
8:  else
9:      x_t := 1;
10:     threshold := ⌊x_{t−1}/2⌋;
11: end if
```

TCP Tahoe is in the congestion avoidance phase most of the time, as the slow starts are quite short. Moreover, with this rather high error rate of 10%, TCP Tahoe frequently experiences errors before it reaches the available bandwidth. TCP Wichita on the other hand reacts faster to the changes. It does not have any linear phases, but overshoots sometimes.

Although a comparison of different TCP versions in the average case is beyond the scope of this paper, we just give one more simulation, namely for a simplified version of *TCP Reno* [17], which is a newer—and more widely used—TCP

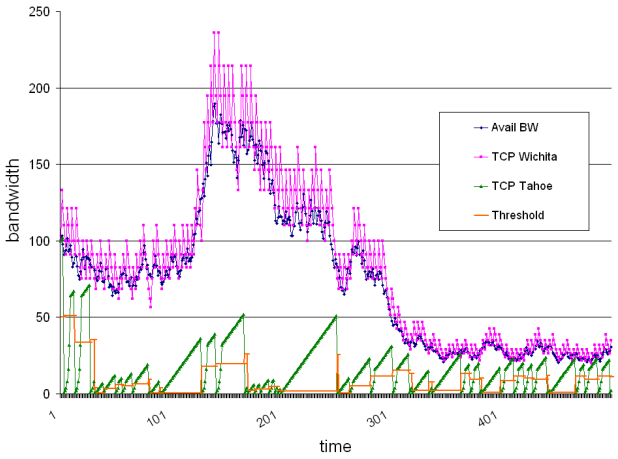

Fig. 9. Sample execution of TCP Wichita and TCP Tahoe for $p = .1$ and $\mu = 1.1$.

Algorithm 3 TCP Reno

```
1: (* Round t *)
2: if round t − 1 successful then
3:     x_t := x_{t−1} + 1;
4: else
5:     x_t := ⌊x_{t−1}/2⌋;
6: end if
```

algorithm. In contrast to TCP Tahoe, TCP Reno cancels the slow-start phase after a triple duplicate ACK (but not after a time-out). The idea is that the arrival of three duplicate ACKs indicates that some packets have been received, implying that the packets may have been lost for other reasons than congestion. This modification is known as *fast recovery*, and it is vital to improve performance in case of random errors.

In our simulation, we assume that there are no time-outs and TCP Reno never performs a slow-start (cf Algorithm 3). Figure 10 depicts a sample execution. It can be seen that while TCP Reno performs better than TCP Tahoe, still many losses occur before the available bandwidth is reached.

Finally, we want to investigate the behavior of TCPW protocols for *non-multiplicative bandwidth changes*. Figure 11 shows a sample execution where the bandwidth varies according to a *normal (Gauss) distribution*. Thereby, the mean is given by the current bandwidth and the standard deviation is 2. For TCPW, we again use $\mu = 1.1$, and the error probability is 10%. Although TCP Wichita's sending rate is often too high, the performance is acceptable.

B. With Congestion Bursts

Finally, in this section, we briefly look at bursty congestion changes [25]. The following random bandwidth changes are considered: In round t, the "adversary" selects a change factor uniformly at random from the interval $c_t \in_R [1, \mu]$ and then increases or decreases (each with probability .5) the bandwidth by c_t. Moreover, ADV saves a factor μ/c_t. In a round where the product of these saved factors exceeds the maximum burst B, the congestion is changes by $B \cdot \mu$, and the burst factor is set

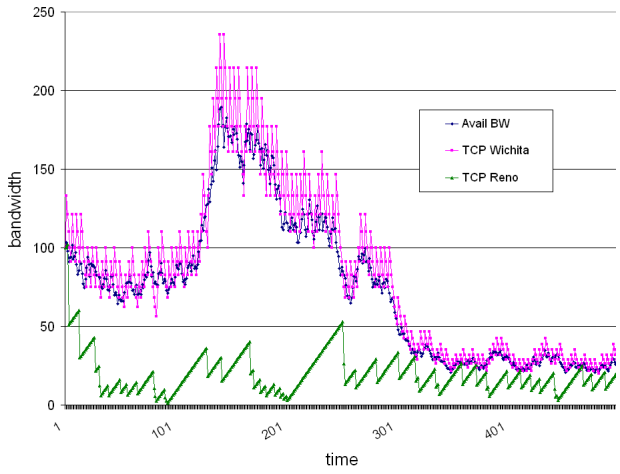

Fig. 10. Sample execution of TCP Wichita and TCP Reno for $p = .1$ and $\mu = 1.1$.

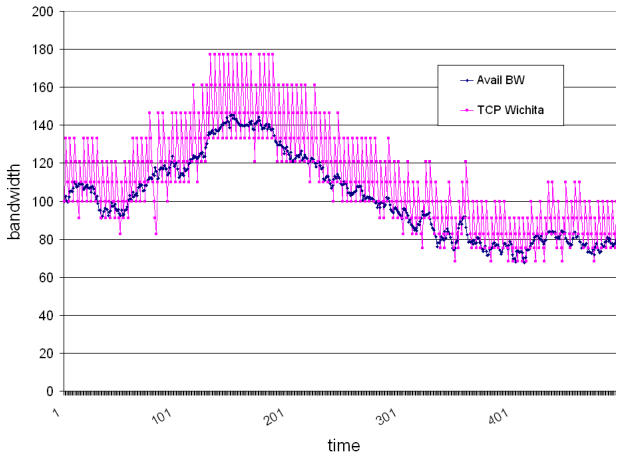

Fig. 11. Sample execution of TCPW for the case of a bandwidth varying according to the normal distribution $\mathcal{N}(u_t, 2)$, and for $p = .1$ and $\mu = 1.1$.

back to 1 (cf Section V). Figure 12 shows a sample execution for $\mu = 1.05$, $B = 1.3$ and $p = .05$.

VII. Conclusion

This paper has introduced a model comprising both worst-case bandwidth changes and lossy links. Moreover, algorithms have been investigated which achieve provable throughputs under this model. Our results are based on the assumption that a *single* (selfish) sender strives for higher data rates. Of course, such a behavior (e.g., the more aggressive bandwidth increase)—if concurrently applied by *many senders*—may threaten the stability of a system. However, this work has focused on a scenario where the available bandwidth is considered to be given externally, and the study of multiple TCPW flows remains to be addressed in future work. It would also be interesting to extend our model to incorporate aspects such as buffers or varying round-trip-times.

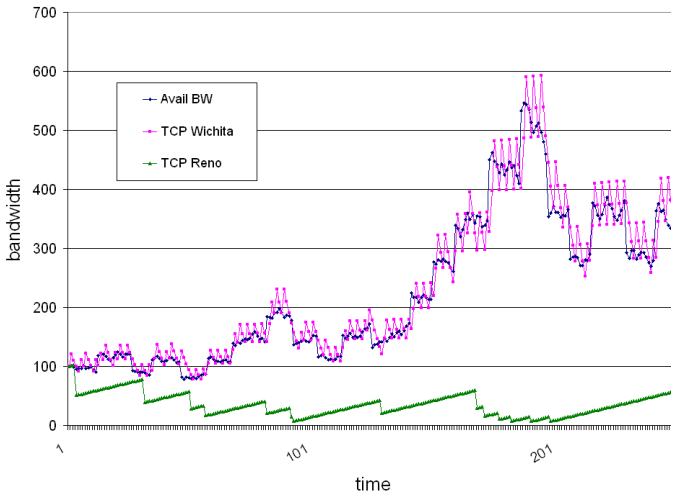

Fig. 12. Behavior of TCPW and TCP Reno for bursty bandwidth changes with $\mu = 1.05$, $B = 1.3$ and $p = .05$.

References

[1] A. Abouzeid and S. Roy and M. Azizoglu. Stochastic Modeling of TCP over Lossy Links. In *IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 1724–33, 2000.

[2] E. Altman, K. Avrachenkov, C. Barakat, A. A. Kherani, and B. J. Prabhu. Analysis of MIMD Congestion Control Algorithm for High Speed Networks. *Comput. Networks*, 48(6):972–989, 2005.

[3] S. Arora and B. Brinkman. A Randomized Online Algorithm for Bandwidth Utilization. In *Proceedings of the 13th Annual ACM Symposium on Discrete Algorithms (SODA)*, pages 535–539, Philadelphia, PA, USA, 2002.

[4] A. Bakre and B. Badrinath. TCP: Indirect TCP for Mobile Hosts. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 1995.

[5] 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.

[6] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proceedings of International Conference on Mobile Computing and Networking (MOBICOM)*, pages 2–11, 1995.

[7] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Inc., 1987.

[8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[9] R. P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, 1973.

[10] C. Casetti and M. Gerla and S. Mascolo and M.Y. Sanadidi and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 287–297, 2001.

[11] Charalampos Samios and Mary K. Vernon. Modeling the Throughput of TCP Vegas. In *ACM SIGMETRICS*, 2003.

[12] Franois Baccelli and Dohy Hong. AIMD, Fairness and Fractal Scaling of TCP Traffic. In *IEEE Conference on Computer Communications (INFOCOM)*, 2002.

[13] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1537–1545, 2000.

[14] Hari Balakrishnan and Srinivasan Seshan and Elan Amir and Randy H. Katz. Improving TCI/IP Performance Over Wireless Networks. In *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking (MobiCom)*, 1995.

[15] R. M. Karp, E. Koutsoupias, C. H. Papadimitriou, and S. Shenker. Optimization Problems in Congestion Control. In *Proceedings of Symposium on Foundations of Computer Science (FOCS)*, pages 66–74, 2000.

[16] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *ACM SIGCOMM Computer Communication Review*, 33(2).

[17] J. F. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[18] T. V. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, 1997.

[19] J.-Y. Le Boudec and P. Thiran. *Network Calculus*. Springer LNCS 2050 Tutorial, 2001.

[20] S. Lu, V. Bharghavan, and R. Srikant. Fair Scheduling in Wireless Packet Networks. *IEEE/ACM Transactions on Networking*, 7(4):473–489, 1999.

[21] P. Karn. MACA—A New Channel Access Method for Packet Radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, 1990.

[22] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 303–314, 1998.

[23] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–7, New York, NY, USA, 1996.

[24] J. Schiller. *Mobile Communications*. Addison-Wesley, 1999.

[25] S. Schmid and R. Wattenhofer. Dynamic Internet Congestion with Bursts. In *Proceedings of the 13th Annual IEEE International Conference on High Performance Computing (HiPC)*, Bangalore, India, 2006.

[26] P. Sinha, N. Venkitaraman, R. Shivakumar, and V. Bharghavan. WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks. *Wireless Networks*, 2002.

[27] R. Stevens and G. R. Wright. *TCP/IP Illustrated Vol. 2 (The Implementation)*. Addison-Wesley, 1995.

[28] I. Stojmenovic. *Handbook of Wireless Networks and Mobile Computing*. Wiley, 2002.

[29] V. Bhaarghavan and A. Demers and S. Shenker and L. Zhang. MACAW: A Media Access Protocol for Wireless LANs. In *Proceedings of the ACM SIGCOMM Conference*, 1994.

[30] W. Willinger, W. Leland, M. Taqqu, and D. Wilson. On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking*, pages 1–15, 1994.