Approaches to *Congestion Control* in packet networks

L. Mamatas and V. Tsaoussidis
Dept. of Electrical & Computer Engineering
Demokritos University
Xanthi, Greece
{emamatas, vtsaousi}@ee.duth.gr

Chi Zhang
School of Computer Science
Florida International University
Florida, USA
czhang@cs.fiu.edu

ABSTRACT

We discuss congestion control algorithms, using network awareness as a criterion to categorize different approaches. The first category ("the box is black") consists of a group of algorithms that consider the network as black box, assuming no knowledge of its state, other than the binary feedback upon congestion. The second category ("the box is grey") groups approaches that use measurements to estimate available bandwidth, level of contention or even the temporary characteristics of congestion. Due to the possibility of wrong estimations and measurements, the network is considered a grey box. The third category ("the box is green") contains the bimodal congestion control, which calculates explicitly the fair-share, as well as the network-assisted control, where the network communicates its state to the transport layer; the box now is becoming green.

We go beyond a description of the different approaches to discuss the tradeoffs of network parameters, the accuracy of congestion control models and the impact of network and application heterogeneity on congestion itself.

Keywords

Congestion Control, TCP, Performance, TCP-Friendly

1. INTRODUCTION

A network is considered congested when too many packets try to access the same router's buffer, resulting in an amount of packets being dropped. In this state, the load exceeds the network capacity. During congestion, actions need to be taken by both the transmission protocols and the network routers in order to avoid a congestive collapse and furthermore to ensure network stability, throughput efficiency and fair resource allocation to network users. Indeed, during a collapse, only a fraction of the existing bandwidth is utilized by traffic that finally reaches the receiver.

Congestion is considered, in general, as a catastrophic event. However, congestion itself is associated with different properties, depending on the characteristics of the underlying networks, the mechanisms of the transmission protocols, the traffic characteristics of the contenting flows, the level of flow contention, and the functionality of network routers. Therefore, the impact of congestion may be temporary and easily controllable; or it may be catastrophic. Consider, for example, a high speed network which hosts a number of competing flows that increases or decreases. The window of each flow also increases and decreases. However, unlike the traditional networks, the time it takes for the flows to exploit the available bandwidth is certainly longer; the amount of loss upon congestion is certainly higher; and the duration of congestion itself throughout the overall communication time may be relatively smaller.

Since the nature of congestion itself cannot be prescribed or even accurately defined in general, congestion control becomes a complex task. Furthermore, complexity increases due to the multipurpose-task of congestion control algorithms. They need to control congestion and avoid collapses, maximize bandwidth utilization, guarantee network stability, and ensure fair resource allocation.

Considering the network as a *black box* that only provides a binary feedback to network flows upon congestion, shifts all the burden to end users and calls for solutions that are more generic and perhaps less responsive. That is, regardless of the network particularity or the network current state, the algorithm will react similarly in all cases. This is not strange. The goal of each sender is to operate independently but nevertheless to adjust its rate (or window) in a manner that the total bandwidth of the network will be expended fairly and effectively. From its algorithmic perspective the above problem is challenging because the distributed entities (sources) do not have any prior or present knowledge of the other entities' states; nor do they know the system's capacity and the number of competitors. Hence, the goal of fairness and efficiency appears initially difficult to attain. However, if the system is entitled to a prescribed behavior and the entities agree on common transmission tactics, convergence[1] to fairness becomes feasible [21]. AIMD, the traditional congestion control algorithm of the Internet, operates within that scope: it increases additively the rate of

[1]Convergence to fairness should be perceived in this paper as the procedure which enables different flows that consume different amount of resources each, to balance their resource usage.

the senders (by a value α) until the system reaches congestion. Upon congestion, all senders decrease their rate multiplicatively using a decrease ratio β.

On the other hand, one can measure network conditions, estimate the available bandwidth or even flow contention, and obtain some knowledge about the network. However, measurements are taken at time-instances which may not necessarily represent current network dynamics, or may not correspond to the overall conditions; consequently, protocols may not manage to accurately estimate the load and predict its duration, resulting in either wrong estimations or wrong recovery strategies. Furthermore, some generic questions cannot really be addressed with certainty: How frequently should we measure the network? How far can we trust our measurements? How responsive should the recovery strategy be? How shall we associate the instantaneous measurements of congestion with the network load over some sufficiently long but also sufficiently recent time period? That said, the network may not be a black box but it is certainly not better than *grey*, involving occasionally a considerable risk.

One can go beyond the blind algorithms or the high risk of estimations and actually ask the network[2] for help. Of course, precision comes at some cost. Besides the practical difficulty of layer collaboration and the issue of convincing people to add functionality (and invest money) to their network, the issue of recovery strategies remains. That is, even when the network is really a *green box* (which practically is very difficult), changes may be so rapid and unpredictable that our costly and painful effort to obtain some information may go wasted.

Beyond our attempt to provide a categorized description of different congestion control strategies, we also attempt to introduce a manner to characterize them comparatively. Hence, we discuss an evaluation framework, which we use to highlight the advantages of different approaches, to exploit the way they handle the tradeoffs of network parameters, and to understand their impact on the network itself or the network applications. In order for us to explore the complexity of the issue, we discuss the nature of congestion itself along with other open issues. Finally, we discuss the open issues that arise mainly from the diversity of applications, the heterogeneity of internetworks, and the interrelation of network parameters.

More specifically, we introduce the control models for congestion and define congestion-related terms, goals and metrics in section 2. In section 3 we discuss end-to-end congestion avoidance and control algorithms that belong in the first category, including the blind AIMD and AIMD-FC algorithms, equation-based congestion control. In section 4 we detail measurement-based avoidance. We present the explicit calculation of the fair-share along with network-assisted schemes in the 5th section. Having discussed the major approaches, we highlight in section 6 some issues which are yet to be solved.

[2]An approach to calculate the fair-share without any network support was published recently in [2]

2. GOALS, METRICS AND MODELS

In the context of our work, we define the following measurement units:

The *window* is a mechanism in the transport layer which limits the number of packets put into the network and the rate describes packets per second or the bits per second. The window or rate can be dynamically adjusted as the total load on the system changes.

A *cycle* is the phase between two seriates feedbacks of 1 (indicating congestion). Hence, a cycle consists of one decrease step triggered by congestion and a number of additive increase steps. A step describes a single window adjustment in response to a single feedback (either 0 or 1).

The system is in an *equilibrium state*, when resource usage of all flows in a bottleneck is balanced. AIMD based congestion control algorithms have guaranteed convergence to equilibrium (due to multiplicative load decrease). In congestion avoidance algorithms this is not always guaranteed.

A non-TCP network protocol is called *TCP-friendly* when it yields the same throughput as TCP and its flows are working fairly together with TCP flows. TCP-friendly protocols are generally optimized for multimedia applications.

Goals we set in the evaluation process of a congestion avoidance/control algorithm are:

- To achieve high bandwidth utilization.
- To converge to fairness faster.
- To minimize the length of oscillations.
- To maintain high responsiveness.
- To coexist fairly and be compatible with traditional widely-used (AIMD based) protocols.

Although the sources discover their fair-share early on, the dynamics of real systems in practice prohibit a straightforward adjustment, but instead, they call for continuous oscillations as a means of discovering the available bandwidth.

Our metrics for the system performance are as follows:

Efficiency Efficiency is the average flows throughput per step (or per RTT), when the system is in equilibrium.

Fairness Fairness characterizes the fair distribution of resources between flows in a shared bottleneck link. A well-known metric is [6]:

$$F(x) = \frac{\sum(x_i)^2}{n\sum(x_i^2)} \quad (1)$$

This index is bounded between 0 and 1.

Convergence speed Convergence speed describes time passed till the equilibrium state.

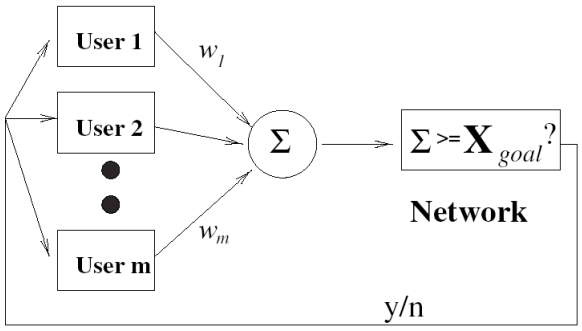

Figure 1: Synchronous control system model of m users sharing a network.

Smoothness Smoothness is reflected by the magnitude of the oscillations during multiplicative decrease. It depends on the oscillations size.

Responsiveness Responsiveness is measured by the number of steps (or RTTs) to reach an equilibrium (i.e. to equate the windows in order to be in a fair state).

A synchronous-feedback control system model is shown in Fig. 1. In congestion control, the load change is the response in one occurred event. This occurred event is a binary feedback. The synchronous model is characterized by a synchronous generation of responses, in congruity with [6]. The system response is 1 when bandwidth is available and 0 when bandwidth is exhausted. The instruction to the system entities (sources) is to increase or decrease their data rate, respectively. Note that in real networks, the responsive behavior of the system is not administered by any centralized authority with additional knowledge of the network dynamics-it is simply a packet drop due to congestion that naturally happens when bandwidth is exceeded.

The system has m users (flows) and the instantaneous throughput for the i_{th} flow is W_i. The system's goal is to operate at an optimal point X_{goal}. Note that this point is not necessarily bandwidth B, since throughput might decrease before we reach B. We assume that responses are synchronous and consequently the duration of RTTs is common for all flows. Hence, the sources respond uniformly by decreasing their windows in response to a 0 signal; they increase their windows by one in response to a signal of 1 (in case of traditional AIMD).

The limitations of the system are derived from the dynamics of packet networks:

- Bandwidth B is limited.
- Each flow is not aware of the throughput rates (window sizes) of other flows.
- Each flow is not aware of the number of competitors in the channel.
- No flow is aware of the size of bandwidth B.

Although the synchronous model is widely adopted, it is associated with a number of assumptions and/or simplifications, which may not really hold in real networks. We note that some drawbacks of the simplified synchronous model are somewhat canceled experimentally, due to the wide use of long ftp flows. Gorinsky in [12] shows that the choice of the model has a direct impact on the results and extends further the model of Chiu and Jain to include different RTTs and consequently asynchronous feedback. MIMD (Multiplicative Increase Multiplicative Decrease), which is not stable in Chiu-Jain model, does converge to fair-states under the more realistic assumption of proportional negative feedback. A new Gorinsky's approach is [11].

In a very recent work [22] Lahanas and Tsaoussidis described an asynchronous-feedback model, which corresponds to the diverse round-trip times (RTTs) of competing flows within the same communication channel. In this system, authors show that congestion epoch, which equals[3] to the RTT of a flow times the number of additive increases, is a common knowledge among all competing flows. Based on this knowledge they proposed a new algorithm that increases the consumption rate proportionally to the RTT of the flow using a mechanism which adjusts the sizes of the windows of the competing flows at the end of each congestion epoch. Long RTT flows are relatively favored because their window is increased faster than in traditional AIMD scheme. The system reaches a window-based equilibrium. This mechanism, named τ-AIMD, has an extra adjustive component τ to the additive increase formula of AIMD.

In paper [35] authors compare AIMD and MAIMD (Multiplicative additive increase and multiplicative decrease). They show that the convergence speeds to fair states of AIMD and MAIMD are close to each other. Furthermore, MAIMD has some advantages. For example, its speed to use available network bandwidth can be much faster than AIMD. Authors have made their investigations using a more realistic asynchronous system model (users can have different round-trip times).

A different approach, based on game theory, is [17].

3. THE BOX IS BLACK: BLIND CONGESTION CONTROL

The Additive Increase Multiplicative Decrease (AIMD) algorithm is used to implement TCP window adjustments; based on the analysis of Chiu and Jain the algorithm achieves stability and converges to fairness in situations where the demand (of competing flows) exceeds the channel's bandwidth [6].

The congestion control in the traditional TCP, is based on the basic idea of AIMD. In TCP-Tahoe and TCP-Reno the additive increase phase is adopted exactly as in AIMD, when the protocols are in the Congestion Avoidance phase. In case of a packet drop, instead of the multiplicative decrease a more conservative tactic is used in TCP-Tahoe. The conges-

[3]This is not an exact equality because the RTT is variable [14] and increases with the load of the system. However, the definition gives an intuition of the relation between congestion epoch and the number of additive increases

tion window resets and the protocol enters again the slow-start phase. On the other hand, in TCP-Reno, when the sender receives 3 DACKS, a multiplicative decrease is used in both window and slow-start threshold. In such case, the protocol remains in the Congestion Avoidance phase. When the retransmission timeout expires, TCP-Reno enters the slow-start phase like in TCP-Tahoe.

3.1 AIMD-FC

A recent improvement to AIMD, Additive Increase Multiplicative Decrease with Fast Convergence (AIMD-FC) was proposed in [20]. AIMD-FC impacts positively both efficiency and fairness. It is not based on a new algorithm, but rather on an optimization of AIMD during the convergence procedure that enables the algorithm to converge faster and achieve higher efficiency. AIMD-FC increases the bandwidth utilization of AIMD from 3/4 to 5/6.

Authors, highlighted the following four observations which were the basis of this work:

1. During the additive increase phase, equal amount of system resources is being allocated to the flows. This amount ('k') is a public or common knowledge (i.e. it is known to every flow in the system).
2. AIMD affects both the initial windows and the amount of system resources (k), that has been fairly allocated, during the multiplicative decrease phase. Note that the manipulation of the initial (and unknown) windows is the real target for achieving fairness.
3. The distance between the bandwidth limit (see figure x) line and the efficiency line when the system is in equilibrium depends only on the multiplicative decrease factor [6].
4. Two algorithms may need the same number of cycles to converge to fairness: for example, two variants of AIMD with different additive increase rate but the same multiplicative decrease ratio. The number of steps determines the relative efficiency of the algorithm to converge to fairness.

Practically, fairness is achieved in AIMD-FC by releasing (through multiplicative adjustments of the windows) the (unknown to other flows) initial resources of the flows, because during the additive increase phase the flows increase their resource consumption uniformly. So, it is becoming apparent that the distinctive difference of AIMD and AIMD-FC is centered on the portion of the congestion window that is affected by multiplicative decrease (this portion is called decrease window).

Furthermore, there are some open issues related to AIMD-FC:

- The efficiency boundaries of AIMD have not yet been exploited.
- Further modifications can be made in order for AIMD-FC to favor responsiveness or smoothness.

The same authors improved in [20] furthermore the efficiency, smoothness and fairness of AIMD-FC and proposed a new algorithm named AIMD-FC+.

3.2 Binomial Mechanisms

Bansal and Balakrishnan presented in [4] a new class of nonlinear congestion control algorithms named Binomial Congestion Control Algorithms. Those algorithms are called binomial because their control is based on the involvement of two additional algebraic terms with different exponents.

While an AIMD control algorithm may be expressed as:

$$Increase : W_{t+R} \leftarrow W_t + a; a > 0 \quad (2)$$

$$Decrease : W_{t+\delta t} \leftarrow (1-\beta)W_t; 0 < \beta < 1 \quad (3)$$

authors generalized the AIMD rules in the following way:

$$Increase : W_{t+R} \leftarrow W_t + \frac{a}{W_t^k}; a > 0 \quad (4)$$

$$Decrease : W_{t+\delta t} \leftarrow W_t - \beta W_t^l; 0 < \beta < 1 \quad (5)$$

where I refers to the increase in window as a result of the receipt of one window of acks within a single RTT, D refers to the decrease in window upon detection of congestion by the sender, Wt the window size at time t, R the flow's RTT, and a, b, k, l are constants.

For example, for k=0, l=1 we get AIMD. Authors proposed, in the (k,l) space, two AIMD variations (which are also TCP-compatible). IIAD (with k=1 and l=0) and SQRT (with k=1/2 and l=1/2) algorithms. The first is called Inverse Increase Additive Decrease (IIAD) because its increase rule is in inverse proportion to the current window. The second is called SQRT because both its increase is inversely proportional and decrease proportional to the square-root of the current window.

Authors in [4] concluded the following:

- A binomial algorithm is TCP-compatible if and only if $k + l = 1$ and $l \leqslant 1$ for suitable a and b.
- Those algorithms may compete unfairly across a drop-tail gateway. Use of a RED active queue management scheme at the bottleneck link alleviates this unfairness problems.
- AIMD is aggressive in probing for available bandwidth for given a and b and is the most efficient and best suited binomial algorithm for applications that can tolerate large bandwidth adjustments.
- Due to the fact that binomial algorithms are nonlinear, for a given two flows with initial windows x1 and x2, the smaller of the two values increases more than the larger one. This leads to a fairer allocation of resources.

- The parameter k represents the aggressiveness of probing and the variable l represents the conservativeness of response to congestion. So, there is a trade-off between k and l in order for a binomial protocol to achieve a certain amount of throughput (at some given loss rate).

3.3 SIMD

Another TCP-friendly nonlinear congestion control algorithm is SIMD [16]. SIMD is the first congestion control algorithm which utilizes history information.

The control rules of SIMD is defined as:

$$Increase : W_{t+R} \leftarrow W_t + a\sqrt{W_t - W_0}; a > 0 \quad (6)$$

$$Decrease : W_{t+\delta t} \leftarrow W_t - \beta W_t; 0 < \beta < 1 \quad (7)$$

where w_0 is the window size after the last decrease and wt is the continues approximation of the window size at time t (in RTTs) elapsed since the window started to increase. Authors show that:

$$w(t) = w_0 + a^2 \frac{t^2}{4} \quad (8)$$

It uses multiplicative decrease like AIMD. But SIMD uses a different increase rule from those used by AIMD and binomial algorithms. A rule which is based on history information.

If two SIMD flows are competing, the flow with the smaller window size is more aggressive due to the nonlinear nature of SIMD. This results in better convergence behavior. Authors show also that SIMD converges faster than memoryless AIMD and binomial controls. Authors evaluated in [16] the TCP-friendliness of SIMD and showed that SIMD can maintain smoothness in steady state.

3.4 Equation-based Congestion Control

3.4.1 GAIMD

General AIMD Congestion Control (GAIMD) is a parameterized TCP-friendly protocol which generalizes AIMD congestion control by parameterizing the additive increase value a and multiplicative decrease ratio b. Authors of [36] extended the throughput equation for standard TCP, proposed in [26], to include parameters α, β:

$$T_{a,b}(p, RTT, T_0, b) = \frac{1}{RTT\sqrt{\frac{2b(1-\beta)}{\alpha(1+\beta)}p} + T_0 min(1, 3\sqrt{\frac{(1-\beta^2)b}{2\alpha}p})p(1+32p^2)} \quad (9)$$

where p is the loss rate; T_0 is the retransmission timeout value; b is the number of packets acknowledged by each ACK. The overall throughput of TCP-Friendly (a, b) protocols is bounded by the average throughput of standard TCP (a=1, b=0.5), which means that equation (11), which is derived from (10) (see [36]) could provide a rough guide to achieve friendliness.

$$T_{a,b}(p, RTT, T_0, b) = T_{1,0.5}(p, RTT, T_0, b) \quad (10)$$

Authors of [36] derive from (1) and (2) a simple relationship for a and b:

$$a = \frac{4(1-b^2)}{3} \quad (11)$$

Based on experiments, they proposed a b=7/8 as the appropriate value for the reduced the window (i.e. Less rapidly than TCP does). For b=7/8, (3) gives an increase value a=0.31.

3.4.2 TFRC

TFRC is also a measurement-based protocol and will be discussed in section 4.

3.4.3 RAP

Rate Adaptation Protocol (RAP) [28] is a rate-based transport protocol friendly to TCP. It employs additive increase and multiplicative decrease algorithm. It decouples network congestion control from application level reliability. However, smoothness is not a design criterion for RAP, compared to TFRC.

3.4.4 Ideally Scalable Congestion Control (ISCC)

Another rate-based congestion control is Ideally Scalable Congestion Control (ISCC) in [23]. ISCC method is based on the idea of "Ideal Scalability". A scheme is defined to have ideal scalability, if S_n is constant for all flows. Where parameter S_n specifies how fast packet loss increases when more flows share a common link and directly relates to the scheme's ability to support a large number of flows.

4. THE BOX IS GREY: MEASUREMENT-BASED CONGESTION CONTROL

Standard TCP relies on packet losses as a congestion indication, or a signal from overloaded links. However, packet loss is not a sufficient indication of congestion, in its own right, for a number of reasons:

1. Packet loss can be caused by random bit corruption when bandwidth is still available.
2. Acknowledgement-based loss detection at the sender side can be affected by the cross-traffic on the reverse path.
3. Packet loss, as a binary feedback, cannot indicate the level of contention before the occurrence of congestion.

Therefore, an efficient window adjustment tactic should adapt to various network conditions, which cannot be signaled simply by packet drops. Several measurement-based transport protocols are based on more precise information on network conditions.

4.1 TCP-VEGAS

A well-designed, measurement-based congestion avoidance mechanism is TCP Vegas [5]. Vegas defines BaseRTT to be the minimum of all measured RTTs, and ExpectedRate to be the ratio of the congestion window to BaseRTT. The sender measures the ActualRate based on the sample RTTs. If the difference between the ExpectedRate and ActualRate is below a lower bound, the congestion window increases linearly during the next RTT; if the difference exceeds an upper bound, TCP Vegas decreases the congestion window linearly during the next RTT. According to [5], Vegas achieves better transmission rates than Reno and Tahoe. However, [13] shows that Vegas can not guarantee fairness. Plus, cannot distinguish nature of error. From the research perspective of the present work it is important to consider that the authors of Vegas demonstrated effectively that measurement-based window adjustment is a viable mechanism.

4.2 TCP-REAL

TCP-Real [37, 34] employs a receiver-oriented and measurement-based congestion control mechanism that significantly improves TCP performance over heterogeneous (wired/wireless) networks and over asymmetric paths. TCP-Real goes beyond the limitation of ack-based binary feedback. It estimates the level of contention and distinguishes the reason of packet losses. TCP-Real relies on:

- Receiver-oriented congestion detection that abrogates the impact of false assessments at the sender due to lost or delayed acknowledgments on a lossy reverse path. The receiver measures the network condition and attaches the results to the ACKs sent back to the sender.
- Measurements based on wave patterns that distinguish the nature of a packet loss (due to congestion or transient wireless errors

A wave [33] consists of a number of fixed-sized data segments sent back-to-back, matching the inherent characteristic of TCP to send packets back-to-back. The receiver computes the data-receiving rate of a wave, which reflects the level of contention at the bottleneck link. If a packet drop is due to a wireless error, the data-receiving rate shall not be affected by the gap of missing packets, since the wave size is published to the receiver. The congestion window is multiplicatively reduced only when a drop is associated with congestion.

4.3 TCP-WESTWOOD

In TCP-Westwood [24], the sender continuously measures the rate of the connection by monitoring the rate of returning ACKs. Upon three duplicate acknowledgments or timeout, the slow start threshold and the congestion window are set in consistent with the effective bandwidth used at the time the packet loss is experienced. No specific mechanism exists to support error classification and the corresponding recovery tactics for wired/wireless networks, albeit the proposed mechanism appears to be effective over symmetric wireless links due to its efficient congestion control.

4.4 TFRC

TFRC is a TCP-Friendly, rate-based congestion control protocol, which intends to compete fairly for bandwidth with TCP flows. The sending data rate is adjusted in response to the level of congestion as it is indicated by the loss rate. This adjustment is "gentle"; that is, its instantaneous throughput has, in general, a much lower variation over time, compared with TCP. The smoothing of the transmission gaps makes TFRC suitable indeed for streaming media, telephony or other applications requiring a smooth sending rate. However, smoothness has its own price: the protocol becomes less responsive to bandwidth availability [38].

Furthermore, TFRC is designed for applications that use fixed sized packets. In case of applications with a variance in their packet size (eq. some audio applications), TRFC's congestion control mechanism cannot be used. A TFRC variant named TFRC-PS (for TFRC-PacketSize) can be used instead. There is no draft available for TFRC-PS yet, but several researchers are still working on these issues.

TFRC introduces the "loss event" instead of the traditional packet loss. A loss event is defined as one or more lost or marked packets from a window of data (a marked packet refers to a congestion indication from Explicit Congestion Notification). TFRC uses a receiver-based mechanism for the calculation of loss event rate. Such a mechanism is suitable for multicast congestion control and also fits in the case of a large server handling many concurrent requests from clients with more memory and the CPU cycles available. The receiver measures the loss event rate and then passes this information to the sender. The sender calculates throughput using a throughput equation that incorporates the loss event rate, round-trip time and packet size.

In summary, TFRC's congestion control mechanism works as follows:

- The receiver measures the loss event rate (based on lost or marked packets from ECN, in a single window) and then feeds this information back, to the sender.
- The sender uses this information, also, to measure the round-trip time (RTT).
- The loss event rate, round-trip time and packet size are used in the throughput calculation function (12). The sender adjusts its sending data rate to match the calculated rate.

$$X = \frac{s}{R \cdot \sqrt{\frac{2 \cdot \beta \cdot p}{3}} + (t\_RTO \cdot (3 \cdot \sqrt{\frac{3 \cdot \beta \cdot p}{8}}) \cdot p \cdot (1 + 32 \cdot p^2))} \quad (12)$$

Where:
X is the transmit rate in bytes/second.
s is the packet size in bytes.
R is the round trip time in seconds.
p is the loss event rate, between 0 and 1.0, of the number

of loss events as a fraction of the number of packets transmitted.
t_RTO is the TCP retransmission timeout value in seconds.
b is the number of packets acknowledged by a single TCP acknowledgement.

It may be useful to associate each information fed back to the sender with a statistical figure, which indicates the possibility of a potential wrong decision. Then, based on this data either an aggressive or a conservative strategy can be chosen.

5. THE BOX IS GREEN: EXPLICIT CALCULATION OF THE FAIR-SHARE

5.1 Bimodal Mechanism

Bimodal congestion avoidance and control mechanism [2] measures the fair-share of the total bandwidth that should be allocated for each flow, at any point, during the system's execution. If the fair-share were known, then the sources could avoid congestion by adjusting immediately after the fair-share was discovered, to a new state where the bandwidth allocation of each flow is exactly its fair-share. However, bandwidth availability is not only a matter of channel capacity but is also dependent upon the number of participating flows, and the transmitting behavior of the sources. So, fair-share can be measured only in an equilibrium state.

Authors proposed in [2] a bimodal mechanism, which is based on the idea that upwards and downwards adjustments need to operate in association with the system state. Action is determined based on whether the system is in equilibrium (fair-share is known) or not (fair-share is unknown).

When the fair-share is unknown, the algorithm behaves like AIMD, until two congestion cycles have passed, which is sufficient to recalculate the fair-share. The algorithm then sets the bandwidth allocation for flow f to $(1-\epsilon)$ [4] times the calculated fair-share, and shifts to known fair-share mode. So, bimodal congestion control algorithm explicitly calculates the fair share and converges in two congestion cycles to the fair share. In this mode, the algorithm continues to use additive increase and multiplicative decrease, but the multiplicative decrease factor is α instead of β.

This algorithm is distinguished from the class of TCP-friendly algorithms. TCP-friendly algorithms favor smoothness at the cost of fairness. This algorithm calculates fair-share explicitly, and so fairness is not compromised in this approach. Furthermore, since this algorithm restricts its flows to use only their fair share, it can be used in conjunction with any other transport protocols (e.g. standard AIMD) without monopolizing for itself most of the available bandwidth. This is in contrast with the TCP-friendly protocols, which attempt to grab all available bandwidth. Of course, this algorithm will not work well in conjunction with other protocols that aggressively (and unfairly) grab a disproportionate share of the bandwidth for their flows.

There are also some open issues related to the bimodal mechanism:

[4] where ϵ is a small tunable parameter

- The more the gain we have in goodput (i.e. by using a smaller ϵ) the less the free space left for incoming flows when contention increases.
- Investigating the optimal value of ϵ in conjunction with the dynamics of specific environments is a subject of future work.
- The modification of the algorithm for the asynchronous scenario by integrating the RTT into the fair-share calculation. For example, bandwidth allocation of a flow can be increased when the RTT of its packets decreases and be decreased when the RTT increases.
- The integration of such ideas with a receiver-oriented feedback approach (like TCP-Real [37]).

5.2 Network-Assisted Congestion Control

Red Gateways [9] drop packets when congestion is about to happen. RED randomly drops packets, triggering multiplicative decrease in some flows when the length of the queue exceeds a predetermined threshold. The goal is to limit the loss down to one packet. It also reduces persistent queuing delay. RED can function without requiring any change to the current transport level infrastructure.

Ramakrishnan and Floyd in [27] proposed an Explicit Congestion Notification (ECN) to be added to the IP protocol in order to trigger TCP congestion control. Unlike RED, ECN enables routers to probabilistically mark a bit in the IP header, rather than drop the packet, to inform end hosts of pending congestion when the length of the queue exceeds a threshold. End hosts multiplicatively reduce their congestion windows upon receiving packets with ECN bit set, before the router buffer overflows and packet drops are inevitable. A duality is served with ECN: TCP performance can be enhanced by means of avoiding losses of data windows due to limited buffer space at the bottleneck router, and congestive collapse can be avoided.

Recent work [25] presents a critical discussion of the performance expectations with RED. An interesting observation about RED and ECN is that they could, somehow, confine future evolution. Imagine a more sophisticated TCP which distinguishes between congestion and wireless losses. Since RED drops packets in proportion to sending rates, it is unclear how fair RED would be to the sophisticated TCP which just happens not to unnecessarily back off in case of transient wireless losses [19].

6. DISCUSSION

There was recently a significant effort to distinguish congestion-related errors from wireless errors. That effort invested mainly in distinguishing the locus of the error [3, 30, 10], rather than focusing on the dynamics from the combination of both errors. For example, persistent congestion which may be experienced in some router in a wired network may change to a more transient one when wireless errors are introduced at the last mile, where some wireless receivers reside. Furthermore, higher contention can be tolerated under the same circumstances of congestion. In a similar context, in a high-speed network congestion may cause more losses due to the fact that congestion windows may grow to very

high values; however, it will last less and potentially, it will appear much less frequently [1, 15, 7, 15, 29, 18]. The dynamics of combined wired and wireless errors appears to be far more important issue than the ability to geographically locate the error and apply the well-known techniques. Transient congestion combined with higher contention may not call for conservative recovery as this is implemented through the extension of the timeout and the shrinkage of the congestion window.

What mechanisms are appropriate to detect the presence of combined congestion and transient random wireless errors? A recent probing scheme appears to measure network performance but also combines the ability to deal with wireless errors: it freezes the timeout and holds still the congestion window without transmitting any data when wireless errors do not allow the probing mechanism to be completed [32]. Other mechanisms measure contention and decouple wireless errors from others. With what precision can we estimate really network conditions? How can we take into account the risk of wrong estimation in order to avoid false recovery strategies. Or, in another context, what are the situations which may tolerate some risk? And, beyond detection, what strategies correspond to each distinctive new class of errors that are detected? How can we evaluate them? For example, authors in [31] have shown that congestion control determines the aggressive/conservative protocol behavior, which in turn affects energy consumption. This issue calls certainly for further investigation.

On another end, applications themselves generate different traffic patterns and hence are associated with different congestion patterns as well. How can we justify our practice to conduct experiments only with ftp traffic? How much different is congestion due to a large number of flows (larger than the potential capacity of a network) than congestion due to limited number of flows with large windows? That is, what is the contribution and interrelation of each of the timeout and the congestion window mechanisms?

The TCP-related work discussed above raises another important question: where is the right place to add the required functionality? This question does not have a clear answer; error control is not exclusively a management property of the router or the base station, nor is it exclusively assigned to the transport layer. A widely accepted approach, presented by the end-to-end argument [8], states that we can only implement a function at a lower layer, if that layer can perform the complete function. Since the lower level cannot have enough information about the application's requirements, protocol parameters, and device constraints, it cannot implement the whole function of error control; it can only be used to optimize the function of the higher layer.

Clearly, it could be beneficial for the sender to know with precision that congestion is about to happen. However, in the context of heterogeneous wired/wireless networks ECN contribution might be limited: by not receiving an explicit notification the TCP sender will not be able to safely assume that a detected drop was not caused due to congestion. The desired precision of ECN capable TCP senders would be better approached if the level of congestion could also be indicated in order to allow TCP to implement more sophisticated recovery. Precision, however, comes at a cost: the functionality of the routers is more complex, modifications are required to both routers and TCP itself and, finally, the return might be small due to heterogeneity, i.e. Some routers are not ECN-capable.

One can go beyond architectural optimization criterions and worry about potential false strategies due to the locality of router decisions. For example, when a router experiences congestion or progresses towards congestion, it takes action. The action is taken in association with an inherent assumption: that the portion of the network that follows will not change the major traffic characteristics of the flows - something not necessarily true when a wireless network is deployed at the receivers' end. Certainly TCP throughput allows for assuming a monotonic behavior. However, when our assumption is violated, there is no indication that our strategy is still correct.

Departing from the same point, we justified earlier the logic behind measurement-based protocols. There is a significant difference however; the receivers have indeed the capability to measure flow traffic throughout the whole network path, hence they do not fall into wrong assessments due to limited view of the network. Perhaps one can trust end-to-end protocols on that aspect, more than network mechanisms.

By the same token, when many flows compete for a limited bandwidth, a window back-off strategy will not yield any significant gain. For example, a large number of flows over a 10Mbps network would probably operate with single-packet windows that do not permit further shrinkage. Obviously, under such contention of packets, efficiency of the algorithm does not really become an issue. Only a proper timeout adjustment can permit all flows to use the network fairly.

Finally, evaluation of congestion control is an important issue in its own right. Several authors have proposed modifications to existing mechanisms; however, only a few of them have seen even a minor deployment. This fact can be mainly justified based on four distinct observations that we summarize below:

1. The nature of the problem itself; its complexity due to the wide number of parameters that have impact on system's performance does not always allow for effective evaluation.

2. The nature of congestion in the modern Internet is not really known; for example, how can we characterize different levels of congestion? How frequently do they happen? What is an extreme condition of congestion, how long does it last and what percentage of packets is dropped?

3. The locus of congestion is not well-determined; where does Internet congestion occur? For example, congestion might appear in the backbone gateway or a local router. The locus is associated with the number of flows that cross the congested router.

4. The goal of congestion control also changes. In today's network congestive collapse is not a primary concern;

smoothness and responsiveness becomes more important.

7. REFERENCES

[1] I. Akyildiz, G. Morabito, and S. Palazzo. TCP Peach: A New Congestion Control Scheme for Satellite IP Networks. *IEEE/ACM Transactions on Networking*, 9(3):307–321, June 2001.

[2] P. C. Attie, A. Lahanas, and V. Tsaoussidis. Beyond AIMD: Explicit fair-share calculation. In *In Proccedings of ISCC 2003*, June 2003.

[3] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP Performance over Wireless Networks. In *Proceedings of of ACM Mobicom '95*, November 1995.

[4] D. Bansal and H. Balakrishnan. Binomial Congestion Control Algorithms. In *Proceedings of the IEEE INFOCOM'01*, 2001.

[5] L. Brakmo and L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas of Communications*, October 1995.

[6] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[7] S. Floyd. Highspeed tcp for Large Congestion Windows. *RFC 3649*, December 2003.

[8] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[10] T. Goff, J. Moronski, and D. Phatak. Freeze-TCP: A True end-to-end Enhancement Mechanism for Mobile Environments. In *Proceedings of the INFOCOM, (Israel), 2000*, 2000.

[11] S. Gorinsky. Feedback Modeling in Internet Congestion Control. In *Proceedings of the NEW2AN'04*, 2004.

[12] S. Gorinsky and H. Vin. Extended Analysis of Binary Adjustment algorithms. Technical report, University of Texas, Austin, 2000.

[13] U. Hengartner, J. Bolliger, and T. Cross. TCP Vegas Revisited. In *In Proceedings of IEEE INFOCOM 2000*, March 2000.

[14] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM '88*, August 1988.

[15] C. Jin, D. Wei, and S. Low. Fast TCP: motivation, architecture, algorithms, performance. In *Proceedings of the IEEE Infocom*, March 2004.

[16] S. Jin, L. Guo, I. Matta, and A. Bestavros. TCP-friendly SIMD Congestion Control and Its Convergence Behavior. In *Proceedings of the ICNP'2001*, November 2001.

[17] R. Karp, E. Koutsoupias, C. Papadimitriou, and S. Shenker. Optimization Problems in Congestion Control. In *IEEE Symposium on Foundations of Computer Science*, pages 66–74, November 2000.

[18] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *In the proceedings on ACM Sigcomm 2002*, 2002.

[19] M. Khanna, C. Zhang, and V. Tsaoussidis. Experimental evaluation of RED in Heterogeneous Environments. *The International Journal of Communication Systems IJCS*.

[20] A. Lahanas and V. Tsaoussidis. Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC). In *Proccedings of Networks 2002*, August 2002.

[21] A. Lahanas and V. Tsaoussidis. Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control. *Computer Networks, Elsevier*, 43(2):227–245, October 2003.

[22] A. Lahanas and V. Tsaoussidis. t-AIMD for Asynchronous Receiver Feedback. In *Proceedings of the ISCC 2003*, 2003.

[23] D. Loguinov and H. Radha. End-to-end Rate-Based Congestion Control: Convergence Properties and Scalability Analysis. *IEEE/ACM Transactions on Networking*, 11, August 2003.

[24] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *Proceedings of the MobiCom'01*, July 2001.

[25] M. May, T. Bonald, and J. Bolot. Analytic Evaluation of RED Performance. In *INFOCOM 2000*, August 2000.

[26] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of the ACM SIGCOMM*, 1998.

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

[28] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *INFOCOM (3)*, pages 1337–1345, 1999.

[29] S. Shalunov. Tcp Armonk (tcpar). Technical report, September 2002.

[30] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2-3):301–316, 2002.

[31] V. Tsaoussidis, H. Badr, X. Ge, and K. Pentikousis. Energy / Throughput Tradeoffs of TCP Error Control Strategies. In *Proceedings of the 5th IEEE Symposium on Computers and Communications, ISCC*, 2000.

[32] V. Tsaoussidis and A. Lahanas. Exploiting the Adaptive Properties of a Probing Device for TCP in Heterogeneous Networks. *The Journal of Computer Communications*, 26, February 2003.

[33] V. Tsaoussidis, A. Lahanas, and C. Zhang. The Wave and Probe Communication Mechanisms. *The Journal of Supercomputing*, Kluwer Academic Publishers, June 2001.

[34] V. Tsaoussidis and C. Zhang. Tcp-real: Receiver-oriented congestion control. *Computer Networks Journal (Elsevier)*, 40(4), November 2002.

[35] R. Yang, M. S. Kim, X. Zhang, and S. S. Lam. Two problems of tcp aimd congestion control. Technical Report TR-00-13, Department of Computer Sciences, University of Texas at Austin, June 2000.

[36] Y. Yang and S. Lam. General AIMD Congestion Control. In *Proceedings of the IEEE International Conference on Network Protocols*, November 2000.

[37] C. Zhang and V. Tsaoussidis. TCP-Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks. In *Proceedings of the 11th IEEE/ACM NOSSDAV*, June 2001.

[38] C. Zhang and V. Tsaoussidis. The interrelation of TCP Responsiveness and Smoothness. In *In Proccedings of 7th IEEE ISCC*, July 2002.