Fast and Fair Handling of Multimedia CAPTCHA Flows

http://dx.doi.org/10.3991/ijim.v9i4.4644

A.N. Tsioliaridou[1], C. Zhang[2] and C.K. Liaskos[3]
[1]Democritus University of Thrace, Xanthi, Greece. [2]Florida International University, USA.
[3]Aristotle University, Thessaloniki, Greece.

***Abstract*—Multimedia CAPTCHAs play a crucial role in bot filtering policies for VoIP applications. Network flows carrying CAPTCHA content have a very small lifespan and require high transmission quality, making for special handling at transport layer. The present study introduces an analysis-derived, rapidly converging control mechanism for CAPTCHA flows. Instead of relying on generic heuristics, the novel scheme enables the flows to estimate their deviation from the equilibrium and adapt to it in a single step. Simulations demonstrate the high efficiency and convergence speed of the scheme, highlighting its unique fitness for the appointed task.**

***Index Terms*—Network congestion control, Fairness, Convergence speed, CAPTCHA application.**

I. INTRODUCTION

Voice over IP services are envisioned as the future of telephony and online communications. However, thinning the borders between computer networks and telephony has its risks. Modern VoIP networks have to account for "bots", i.e. pieces of software that perform automated calls for purposes ranging from advertisement to denial of service [1]. Call filtering policies are employed in retaliation. An integral part of these policies are the CAPTCHAs, a form of reverse Turing test which is administered as a means of proving that the caller is indeed human. CAPTCHAs generally come in multimedia form and contain simple puzzles to be solved, involving text, video and sound recognition in the process [2]. The proper management of CAPTCHA flows at transport layer is the focus of the present study.

Performing CAPTCHA tests on users decreases their quality of experience. This situation can be aggravated further under presence of network issues. Audio/video problems on the received content could result into test failure, leading to caller blacklisting, blocking or banning, as per the enforced policy. Long delays in reception are not acceptable either, since the tests are administered with time restrictions. Using an error control-capable transport protocol (TCP variant) is in general mandatory. Thus, the problem can be reduced to proper bandwidth allocation for the CAPTCHA-conveying network flows.

However, a unique attribute of a CAPTCHA flow is its small size. The actual media content typically varies from a few KBytes to a few MBytes. Thus, a desirable attribute of a proper flow management scheme is the fast convergence to equilibrium. The coexistence with flows with larger lifespans (e.g. FTP, streaming) could pose an issue [3]. A commonly used approach is then to employ separate router buffers for the short-lived flows [4]. The same outcome can also be accomplished by placing the flows into virtual dedicated channels [5]. Nonetheless, router buffer allocations, or dedicated bandwidth, are a scarce and valuable resource. The corresponding CAPTCHA flows will eventually saturate the available buffers, causing congestion. The present work offers a novel way of CAPTCHA flow management at transport layer. The study is motivated by the results of [6], which shows that the delay and jitter per flow are minimized when all flows are assigned their fair share of the bandwidth. A mechanism is presented that allows each flow independently to estimate its deviation from its fair-share. More particularly, each flow becomes aware of whether it has operated beyond, below or around its fair-share, which allows them to determine the next congestion control strategy for fast convergence to fairness. We then introduce the necessary window adjustment rules. According to the Fair-Share rule each flow can estimate and rapidly operate at its fair-share. Simulation results confirm that, in the presence of the Fair-Share rule, the system converges in one congestion cycle (one epoch) to equilibrium.

II. ANALYSIS

We assume the typical bottleneck topology of Fig. 1. A number $i = 1 \ldots N$ of flows compete for the link and adjust their congestion windows, $w_i(t)$, following the Additive Increase-Multiplicative Decrease rule [7]. The system model is characterized by synchronous flow notification for congestion events, which is a classic assumption as well [8]. We define the instantaneous Throughput (T) of the i^{th} flow at time t as:

$$T_i(t) = \frac{w_i(t)}{R(t)} = \frac{w_i(t)}{R_o + q_d(t)} \quad (1)$$

where R(t) is the time-variant round trip time, comprising the queuing delay $q_d(t)$ and a constant factor R_o representing physical attributes:

$$R_o = 2\left(d_{src} + d_b + d_{\text{sink}}\right) \quad (2)$$

The flow windows undergo the following repeating procedure. Initially, all flows are in the additive increase stage. At the time the bandwidth is exhausted and the buffer has overflowed (cliff point) the network signals the senders to decrease their data rate.

Consider a flow i that competes for the bw_b bandwidth. After a successful delivery of a packet, it receives the relevant ACK from the receiver. Consider two ACKs that it receives between the system knee and cliff points [8], one at time t_1 and another at time t_2. The throughput of the i^{th} flow at time t_1 and t_2 are $T_i(t_1)$ and $T_i(t_2)$ respectively, and $\Delta T_i(t_2)$ is:

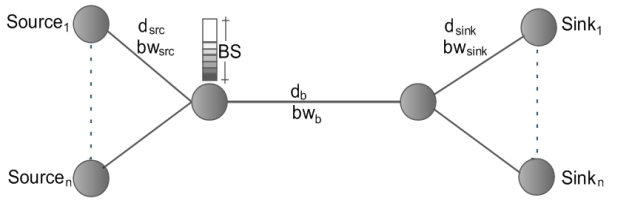


Figure 1. A typical model of n users sharing a link. $bw_{(*)}$ and $d_{(*)}$ denote the bandwidth and delay respectively. BS is the buffer size at the router.

$$\Delta T_i(t_2) = T_i(t_2) - T_i(t_1) = \frac{w_i(t_1) + \Delta w_i(t_2)}{R(t_2)} - \frac{w_i(t_1)}{R(t_1)} = \frac{\Delta w_i(t_2) R(t_1) - w_i(t_1)\Delta d_q}{[R_o + d_q(t_2)][R_o + d_q(t_1)]} \quad (3)$$

where Δq_d is defined as:

$$\Delta d_q(t_2) = d_q(t_2) - d_q(t_1) = \frac{\Delta\left(\sum_{k=1}^{n} w_k(t_2)\right)}{bw_b} \quad (4)$$

Consequently from (3) and (4):

$$\Delta T_i(t_2) = \frac{\Delta w_i(t_2) R(t_1) - w_i(t_1)\dfrac{\Delta\left(\sum_{k=1}^{n} w_k(t_2)\right)}{bw}}{[R_o + d_q(t_2)][R_o + d_q(t_1)]} = \frac{\Delta w_i(t_2) R(t_1) bw - w_i(t_1)\Delta\left(\sum_{k=1}^{n} w_k(t_2)\right)}{[R_o + d_q(t_2)][R_o + d_q(t_1)]bw} \quad (5)$$

Remark 1. If network resources were allocated equally among competing flows, they would all experience the same increase of their congestion window, and equal to $\Delta w_i(t_2)$ during the time period from t_1 to t_2, defined as:

$$\Delta\left(\sum_{k=1}^{n} w_k(t_2)\right) = n\Delta w_i(t_2) \quad (6)$$

Due to this remark, equation (5) becomes:

$$\Delta T_i(t_2) = \frac{\Delta w_i(t_2) R(t_1) bw - w_i(t_1)\cdot n\cdot \Delta w_i(t_2)}{[R_o + d_q(t_2)][R_o + d_q(t_1)]bw} = \frac{\Delta w_i(t_2)}{[R_o + d_q(t_2)]}\left[1 - \frac{n\cdot w_i(t_1)}{R(t_1)\cdot bw}\right] \Leftrightarrow \quad (7)$$

$$\Delta T_i(t_2) = \frac{\Delta w_i(t_2)}{[R_o + d_q(t_2)]}\left[1 - \frac{n\cdot w_i(t_1)}{\sum_{k=1}^{n} w_k(t_1)}\right] \quad (8)$$

Note that the term $\Delta w_i(t_2) \big/ [R_o + d_q(t_2)]$ in (8) is always positive, while the term $1 - n \cdot w_i(t_1) \big/ \sum_{k=1}^{n} w_k(t_1)$ might be either positive or negative.

Corollary 2. Equation (8) enables the i^{th} flow to determine whether it operates below, beyond or at its fair-share as follows:

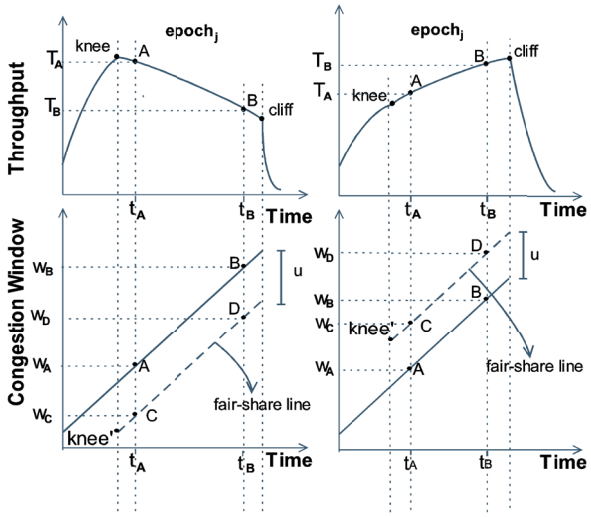


Figure 2. Flow operation above (left) and below (right) its fair-share.

- If $\Delta T(t_2)<0$, the i^{th} flow at time t_1 has reached a rate above its fair-share, since $\sum_{k=1}^{n} w_k(t_1) < n\cdot w_i(t_1)$.
- If $\Delta T(t_2)>0$, the rate of i^{th} flow at time t_1 is below its fair-share, since $\sum_{k=1}^{n} w_k(t_1) > n\cdot w_i(t_1)$.
- If $\Delta T(t_2)=0$, the i^{th} flow has reached its fair-share, since $\sum_{k=1}^{n} w_k(t_1) = n\cdot w_i(t_1)$.

Therefore, equation (8) establishes a criterion, based on which each flow independently can review its transmission rate and deduce whether it operates greedily, fairly or suffering mistreatment.

Theorem 3. In order to operate at its fair-share, the flow should adjust its congestion window, according to equation:

$$w_{fair-share} = \frac{w_B + u}{R_B}\cdot R_o \quad (9)$$

where $u = \left(R_A w_B - R_B w_A\right) \big/ \left(R_B - R_A\right)$.

Proof. We examine the cases defined by Corollary 2. Notations A and B represent two measurement samples during the j^{th} epoch at times t_A and t_B, where $t_A<t_B$ and R_A, $R_B>R_o$.

1st case (operation above the fair-share, left part of Fig. 2). Consider two times, t_A and t_B during the j^{th} epoch, where $t_{knee}<t_A<t_B<t_{cliff}$, that flow f_1 receives feedback (ACKs) from the receiver. Flow f_1 records the congestion window values of the acknowledged packets ACK_A and ACK_B, measures the round-trip time of packets, R_A and R_B, and calculates the corresponding values of throughput, T_A, T_B.

Since flow f_1 has exceeded its fair-share, the throughput values during the period t_{knee}-t_{cliff} are decreasing and the

congestion window values (w), which guarantee flow f_1 operation at its fair-share during the j^{th} epoch, is lower than the current value of operation.

Consider that the optimal congestion window values, that guarantee operation at its fair-share at times t_A and t_B are w_C and w_D respectively. Consequently, the straight line connecting points C and D is the fair-share line. According to (8) it will hold that $T_C=T_D$ and, therefore:

$$\frac{w_C}{R_C} = \frac{w_D}{R_D} \tag{10}$$

Since the additive increase factor is the same in both cases (A_I=1), vector $\overrightarrow{CD}$ is parallel to vector $\overrightarrow{AB}$, and thus it holds that $w_C=w_A+x$ and $w_D=w_B+x$. Consequently, from (10):

$$\frac{w_A - u}{R_C} = \frac{w_B - u}{R_D} \tag{11}$$

Note that due to the different rate of incoming packets at the router during the optimal epoch, the flow f_1 might not have measurement samples for which $R_C=R_A$ and $R_D=R_B$. However the fair-share line of the congestion window does not depend on these values, since the additive increase factor is static (e.g. unary). Hence, from (11) we derive:

$$u = \frac{R_B w_A - R_A w_B}{R_B - R_A} \tag{12}$$

The optimal decrease of the congestion window at the end of the j^{th} epoch, should also guarantee that the system operates beyond the efficiency line. The latest is guaranteed as long as $R_{knee'}=R_o$. So, we seek to satisfy:

$$T_{knee'} = \frac{w_{knee'}}{R_o} = \frac{w_D}{R_B} \tag{13}$$

From equations (12) and (13) we conclude:

$$w_{fair-share} = w_{knee'} = \frac{w_B + u}{R_B} \cdot R_o \tag{14}$$

2nd case (operation below the fair-share, right part of Fig. 2). Since flow f_1 has consumed less resources than its fair-share, the throughput line of the flow is increasing and the optimal congestion window line, the fair-share line, is greater than the measured one. Similarly to before we conclude that:

$$w_{fair-share} = w_{knee'} = \frac{w_B + u}{R_B} \cdot R_o \tag{15}$$

where $u = \frac{R_A w_B - R_B w_A}{R_B - R_A}$.

3rd case (the flow has reached it fair-share). Via (8):

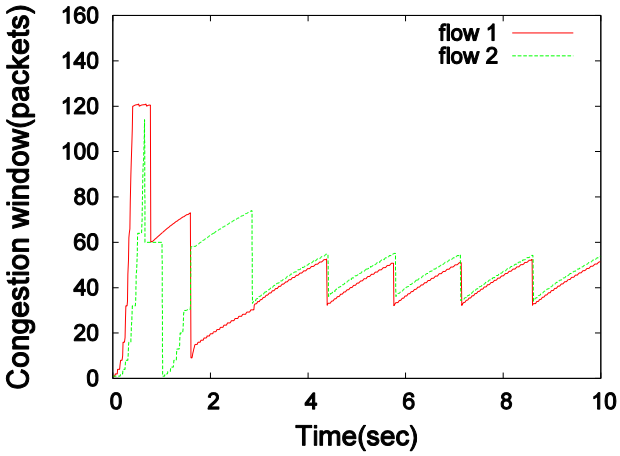
$$T_A = T_B \Leftrightarrow \frac{w_A}{R_A} = \frac{w_B}{R_B} \tag{16}$$

Consequently, from (12) we deduce $u = 0$ and from (9):

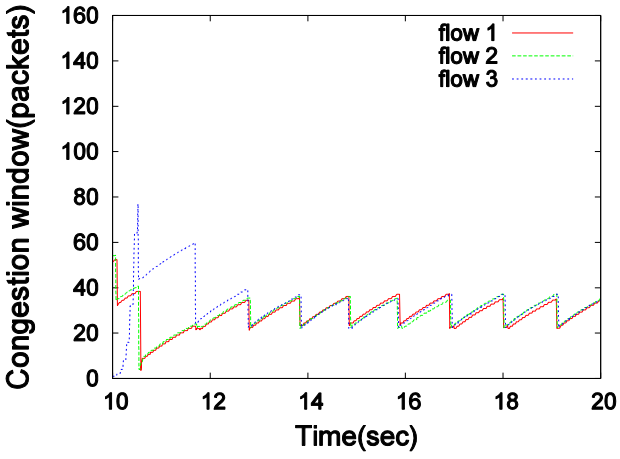
$$w_{fair-share} = \frac{w_B}{R_B} \cdot R_o = w_{knee} \tag{17}$$

which is true, since the flow has operated at its fair-share line during the j^{th} epoch.

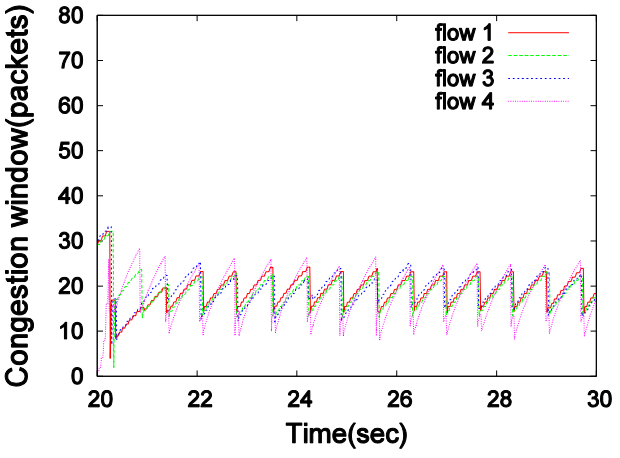
Conclusively, using Corollary 2 and Theorem 3 a flow can instantaneously set its congestion window to the fair-share value, without relying on simple, converging heuristics like the multiplicative decrease of the AIMD scheme.



(a) Windows during 0-10 sec.



(b) Windows during 10-20 sec.



(c) Windows during 20-30 sec.

Figure 3. Flow congestion windows for TCP-FS. All flows, new and existing, converge to their appropriate shares in just one epoch.

III. SIMULATIONS

The convergence benefits of the Fair-share rule are demonstrated using the NS-2 simulator [9]. The simulations assume the standard topology of Fig. 1. A bw_b = 10Mbps, d_b = 5msec bottleneck link emulates the trivial bandwidth dedicated to CAPTCHA flows. All remaining link bandwidths are set to 50Mbps, while the delays vary per examined case. The queue buffer size is set based on the Bandwidth Delay product.

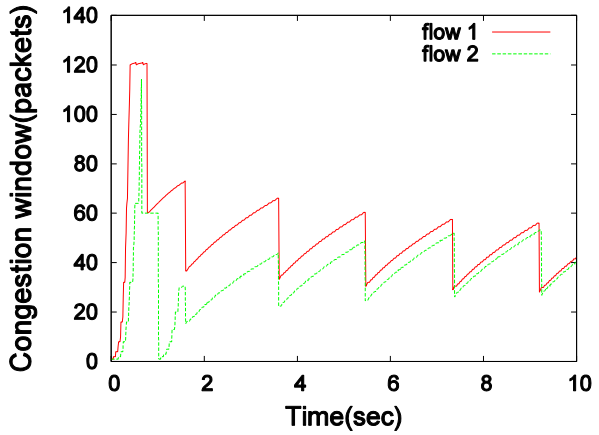


Figure 4. Congestion windows of flows handled by the classic TCP. The flows converge slowly to their fair shares at ~ 10sec. However, a typical CAPTCHA or control flow can have equal or smaller lifespan.

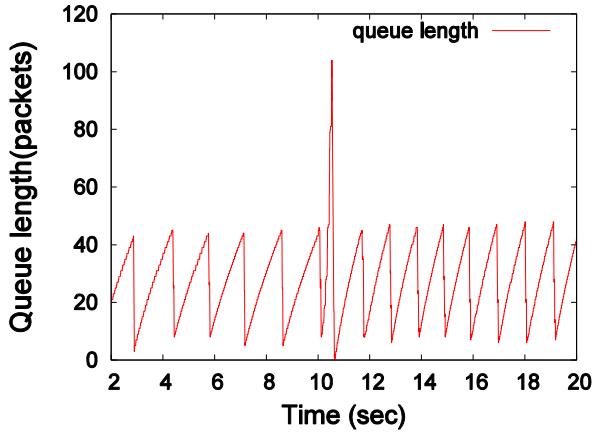


Figure 5. Queue length of TCP-FS. The link remains highly utilized throughout the operation of the novel scheme.

The employed active queue management, i.e. the flow congestion notifier mechanism, is the EGCN algorithm (Explicit Global Notifier), which has been shown to promote fairness with regard to droptail-based approaches [6]. The following simulations considered a varying number of flows and present their time-variant congestion windows. The novel, Fair-share congestion management, denoted as TCP-FS, is compared to the standard TCP which relies on the Additive Increase-Multiplicative Decrease operation (AIMD) [7].

The following contention increase scenario is employed. Initially, two flows with the same round-trip time, 50msec enter the system at 0sec and at 0:1sec respectively (Figure 3a). An additional flow is introduced at 10sec with the same round-trip time (Figure 3b). Finally, at time 20sec, a fourth flow enters the system with different round-trip time, 15msec, to study the impact of different RTTs (Figure 3c).

The behavior of the standard TCP is shown in Fig. 4. The flows slowly converge to their fair shares after an interval of approximately 10sec. Notice that the standard lifespan of a CAPTCHA flow can be equal or less, posing an issue of convergence potential overall. The behavior is expected, since the multiplicative decrease of the AIMD strategy is a heuristic that treats the network in a generic way. The same applies to newer variants, such as [10]. Without explicit knowledge of the current deviation from equilibrium, the system is bound to converge in several iterations. For times 10-30 sec, TCP exhibits similar behavior as well.

Concerning the TCP-FS, it is shown that at the end of an epoch, where all co-existing flows measure their Throughput samples, the flows succeed in adjusting their window to their fair-share. This is confirmed by the equally assigned congestion windows of Fig. 3a, 3b and 3c. Even when new flows enter the system (e.g. flow 3 at time 10sec in Fig. 3b), all flows adjust their rates to the new fair-share in just one epoch. Concerning the addition of the fourth flow in Fig. 3c, it is observed that the Fair-share rule correctly assigns a smaller congestion window to it, in order to make up for the higher round-trip time.

Finally, Fig. 5 confirms that the fair allocation of resources is achieved in conjunction with maintaining high levels of link utilization. The system operates between the knee and cliff points, and utilization is affected momentarily (no more than an epoch), which is the time required for all flows to evaluate their new fair-share.

IV. CONCLUSION

A novel, rapidly converging congestion control scheme for the TCP protocol was presented. The new scheme promotes the fair handling of short-lived flows, an attribute of special interest in VoIP/CAPTCHA traffic. Analysis-derived metrics allow the competing flows to estimate their deviation from the equilibrium and converge in a single step. Simulations demonstrated the ability of the scheme to converge fast while maintaining high channel utilization.

REFERENCES

[1] S. Karapantazis and F.-N. Pavlidou, "VoIP: A comprehensive survey on a promising technology," Computer Networks, vol. 53, no. 12, pp.2050–2090, 2009. http://dx.doi.org/10.1016/j.comnet.2009.03.010

[2] E. Pardede, D. Taniar, I. Awan, W. Al-Sudani, A. Gill, C. Li, J. Wang, and F. Liu, "Protection through multimedia CAPTCHAs," in Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia - MoMM '10. ACM Press, 2010, p. 63.

[3] Liang Guo and I. Matta, "The war between mice and elephants," in Proceedings Ninth International Conference on Network Protocols. ICNP 2001. IEEE Comput. Soc, 2001, pp. 180–188.

[4] B. Goode, "Voice Over Internet Protocol (VoIP)," Proceedings of the IEEE, vol. 90, no. 9, 2002. http://dx.doi.org/10.1109/JPROC.2002.802005

[5] B. Etefia and J. Hant, "Delay/overhead measurements for circuitemulation tunnels," in 2011 - MILCOM 2011 Military Communications Conference. IEEE, 2011, pp. 1684–1689. http://dx.doi.org/10.1109/MILCOM.2011.6127552

[6] A. Tsioliaridou and V. Tsaoussidis, "Fast Convergence to Network Fairness," Systems and Software, vol. 83, no. 5, pp. 745–762, 2010. http://dx.doi.org/10.1016/j.jss.2009.11.715

[7] V. P. M. Allman and W. Stevens, "TCP Congestion Control," RFC2581,1999.

[8] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," ComputerNetworks and ISDN Systems, vol. 17, no. 1, pp. 1–14, 1989. http://dx.doi.org/10.1016/0169-7552(89)90019-6

[9] Network simulator, "The Network Simulator 2," 2012. [Online]. Available: http://www.isi.edu/nsnam/ns/

[10] Lin Cai, Xuemm Shen, J. Mark, and Ranping Pan, "A QoS-aware AIMD protocol for time-sensitive applications in wired/wireless networks," in Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2005, pp. 2008–2019.

AUTHORS

A.N. Tsioliaridou is with Democritus University of Thrace, Xanthi, Greece.

C. Zhang is with Florida International University, USA.

C.K. Liaskos is with Aristotle University, Thessaloniki, Greece.

This work was partially funded by the GSRT/CO OPERATION/SPHINX Project (09SYN-72- 419). Submitted 21 April 2015. Published as resubmitted by the authors 20 August 2015.