TCP SYN Protection: An Evaluation

Pascal Anelli, Fanilo Harivelo
LIM
Université de la Réunion
La Réunion, FRANCE
e-mail: pascal.anelli@univ-reunion.fr,
fanilo.harivelo@univ-reunion.fr

Richard Lorion
LE2P
Université de la Réunion
La Réunion, FRANCE
e-mail: lorion@univ-reunion.fr

***Abstract*—The drop of initial TCP control packets can dramatically penalize flow performance. More the flow is small, more the penalty is important. This paper studies an Active Queue Management (AQM) aiming to protect TCP SYN and SYN-ACK from losses, and evaluates the improvements for short TCP flows and the impacts on long-lived TCP flows. This AQM is an extension based on Random Early Detection (RED). Evaluations are performed in the ns-2 simulator. Results demonstrate the effectiveness of the idea supported in a decrease in the transfer delay of short flows and indiscernible effects on large flows.**

***Index Terms*—TCP; Short-lived flow; AQM; RED; Connection establishment.**

I. INTRODUCTION

Each of us has already noticed that sometimes a web page takes a significant time to display. Asking for refresh permits the page to be loaded. One possible reason for this behavior could be the loss of SYN or SYN-ACK TCP segments as explained in [1]. These TCP segments are exchanged in the three-way handshake procedure used in the connection establishment phase [2]. Congestion is the main cause of these losses. Network reacts by dropping packets in a router's queue. TCP relies on Round Trip Time (RTT) values to tackle packet losses. As no RTT estimate is available at connection opening, the retransmission timeout (RTO) is set by default to 3 seconds [3]. Recently, the IETF working group tcpm (*TCP Maintenance and Minor Extensions*) has proposed decreasing the default RTO value to 1 second [4]. The rationale behind this change is that the RTT of more than 97.5% of the connections observed in a large scale analysis were less than 1 second [5]. However, retransmission rates within the three-way handshake are measured roughly at 2%. This shows that a solution to avoid packet loss in the connection establishment phase will benefit a non-negligible set of connections. While setting the initial RTO value to 1 second provides interesting results, its deployment requires end host modification. Furthermore, one second is inappropriate for brief exchanges. We offer here another option. Rather than dealing with retransmission concerns, i.e., after the loss, we act upstream. Our idea consists in preventing initial packet losses, avoiding in this way, the RTO triggering.

Loss issues at the connection establishment phase are not limited to TCP flow. All connection-oriented services and protocols, such as DCCP and SCTP, are affected. However, short-lived TCP flows are the most impacted. Those are flows made up of few packets. The applications generate short flows in order to ensure an interactive feature. As presented in [6] and [7], Internet traffic is mostly populated by short-lived TCP flows, mainly generated by Web applications. Theses flows suffer more from the initial drops than long-lived flows. Indeed, the connection establishment phase is a process that takes a significant time compared to the duration of the connection. The RTO expiration and corresponding backoff time, due to SYN or SYN-ACK packet loss, add a delay that is significant for short-lived flows. The RTO penalty is perceptible at the service level as the main performance metric is the latency. On the other hand, when a long-lived flow experiences an initial drop, it is equivalent to a shift in the starting of data transfer. The performance metric for this kind of flow is the goodput.

In this paper, we aim to evaluate the benefits of protecting the packets used in connection establishment. The evaluation is made in the context of TCP. The idea is to protect the TCP segments with the SYN flag set (referred to as SYN segment or SYN packet in the following) from losses. The term protection means keeping the SYN segment even if the queue is full. This action is performed on the router's queue. SYN packets are never dropped whenever data packets are present in the queue. Queue management is done in a push-out fashion. If the queue is full on a SYN packet arrival, the last enqueued data packet is pushed out of the queue and dropped. The motivation to act at the router level is that the congestion and the choice of, which packet must be lost, are made at the router level. Furthermore, as the problem affects all connection-oriented services, the solution at this network level deals with this problem globally. With the proposed solution, only data packets are intended to be lost. These packets will be recovered in a better fashion than TCP SYN segment, i.e., either by fast retransmit [8] or by RTO adjusted relatively to the RTT estimations. No additional change other than SYN packet protection is done on routers. Provided service offered by network remains best effort.

The main contribution of this work is the demonstration that the proposed scheme is able to significantly improve the performance of flows by the protection of the segments exchanged in the initial phase. The proposition does not involve a complex identification scheme or per-flow state management. Improvements should be obtained without penalizing long

 ISBN: 978-1-61208-183-0

flows. We propose an implementation of the idea to prove its effectiveness. We then study the integration of the proposed scheme with an existing Active Queue Management (AQM) mechanism, namely, Random Early Detection (RED). This extension of RED is developed and analyzed.

The next section presents the related work that deal with connection establishment and packet losses due to congestion. The proposed idea is described in Section III. In Section IV, we evaluate the performance experienced by short and long TCP flows with the implementation of the proposed AQM. We conclude by our findings.

II. Related work

In the literature, there are overall three types of approaches that address the problem of losses:

1) Preventive actions to reduce the loss rate. Indeed, Besides wasted bandwidth, the retransmission of lost packets introduces extra delay. AQM aims to deal with this congestion issue.
2) Marking packets rather than dropping them; and
3) Responsive actions to improve the retransmission procedure. Solutions based on this approach consider modification of TCP settings.

As mentioned previously, our motivation to act at the router level is that the congestion occurs at this level. For more than a decade, the research community has developed Active Queue Management (AQM) in order to prevent packets being dropped and to maintain high throughput and low delay. In [9], it is recommended to deploy RED [10]. RED monitors the queue length and adopts a packet drop policy based on probabilities, which increase with the level of congestion. However, RED fails to improve the performance of short flows and to provide fairness with unresponsive flows. Choke [11] has been proposed as a solution for this problem. It approximates max-min fairness for the flows that pass through a congested route. It draws a packet at random from the FIFO buffer when a packet arrives and compares it with the arriving packet. If they both belong to the same flow, they are both dropped. Choke can also be considered as a solution for the short-lived flows by considering that it corrects unfairness problems between short and long-lived flows. However, Choke doesn?t prevent the SYN lost when the queue is full. Another way in the router context is the use of DiffServ architecture [12]. In [6], a proposition relies on DiffServ to protect retransmissions and the first packets against loss. After loss detection, the segments are sent with higher priority. This solution is inappropriate for best effort network.

These previous works handle the problem of packet losses, but they do not specifically focus on connection packets. In [13], the authors recognize this problem of lost packets belonging to the connection establishment phase and their simulations show how the response time can be significantly increased by just avoiding the loss of the SYN packet. They show that setting Explicit Congestion Notification (ECN) bits [14] in IP header of TCP control packets while leaving the treatment of the initial TCP SYN packet unchanged, can significantly improve system performance. But, as authors mentioned, this method has a limited scope due to poor usage of ECN on servers and in routers.

Another way to improve the performance in case of lost SYN packets is based on the modification of TCP settings in the operating system, such as defining a smaller value [4] to the initial retransmission value. In [1], the authors investigate the possibility of setting the initial retransmission time to a value smaller than 3 seconds. However, this will then apply to every TCP connection and possibly introduce unnecessary retransmissions and could even cause TCP to fail in certain cases of extreme delay. So they implement an application layer tool to keep a copy of sent packets belonging to the connection opening phase. In case the corresponding acknowledgement does not arrive within a given and a configurable time, the packet is retransmitted. The designed application can be used only for specific ports, such as 80 and not for all TCP connections as opposed to the approach of RTO decrease.

III. Description

Our solution to initial drops in the connection establishment phase is to protect SYN packets within the network. As those losses appear in congestion situations, the proposition takes place on routers. Indeed, a congested router drops packets when its queue fills up (or is about to be filled).

Two types of approaches are possible: scheduling and active queue management. In scheduling, router's buffer is partitioned into separate queues. Each queue holds the packets of one flow or a category of flow. A scheduling mechanism determines which packet to serve next; it is used primarily to manage the allocation of bandwidth (and provide fair sharing) among flows but it can also apply to traffic protection or isolation. This is an interesting option for the isolation of SYN packets from the other traffic. However, algorithmic complexity and scaling issues of scheduling make its deployment on Internet routers difficult.

On the other hand, active queue management, which is concerned with managing the length of packet queues by dropping packets when necessary or appropriate, has a simpler design. A single queue contains all the packets. The deployment of RED, that falls within this class, on Internet routers is highly recommended. RED possesses interesting and useful features; such as its ability to avoid global synchronization, its ability to keep buffer occupancies small and ensure low delays, and its lack of bias against bursty traffic. Our proposition is, then, compared to AQM mechanisms. As the comparison holds on the effectiveness of the SYN packet protection, RED is extended with this additional feature. This new variant of RED will be referred to as REDFavor hereinafter.

With REDFavor, the router serves as a shield for SYN packets against losses. A congestion episode manifests itself by the filling up of the queue. Any new arriving packet is discarded. In normal operation, the router performs this dropping with no regard to the packet type. REDFavor reacts in a different manner if the new packet is a SYN packet. The router makes sure that no SYN packet is rejected if at least

 ISBN: 978-1-61208-183-0

Algorithm 1 REDFavor algorithm

```
1: function enqueue(p)
2: # A new packet p arrives
3: if the SYN flag is set on p then
4:    # p must be protected
5:    if the queue is full or p is an early drop packet then
6:       if only protected packets in the queue then
7:          p is drop
8:          return
9:       else
10:         # Push out
11:         the last standard packet is dropped
12:      end if
13:   end if
14:   p is enqueued in front of all standard packets
15: else
16:   # p is a standard packet
17:   Fall back to RED
18: end if
```

one standard packet is present in the queue. A standard packet is dequeued and dropped in a push-out fashion, as presented in Algorithm 1, to release space for the SYN packet. This latter is, then, enqueued. However, if all packets in the queue are SYN packets, the arriving SYN packet will be dropped as there is no possibility of making room for it.

Thus, although SYN packets are protected during congestion periods at the expense of standard packets, they can still encounter losses. That happens when the queue contains only SYN packets. To lower this potential risk, SYN packets accumulation must be avoided. One response to this point consists in limiting their waiting time in the queue as much as possible. Then, a new enqueued SYN packet is positioned in front of all standard packets and at the tail of already enqueued SYN packets. Thus, it is prioritized in transmission over standard packets.

The exposed protection mechanism can be considered as an isolation or separation of SYN packets from standard ones. This separation relies on SYN flag identification. This flag acts as a priority bit that triggers special and privileged treatment for corresponding packets. A transport layer signalisation is handed over to network-level entities to solve a transport layer issue. Such cooperation can be seen as a cross-layer approach. This operation does not involve complex scheme or per-flow state management. A simple check on the SYN flag suffices; this ensures the scalability of deployment on real networks.

Nevertheless, some questions may arise with the use of isolation and prioritization of SYN packets. Indeed, both types of packets are competing for transmission. One possible issue might be the starvation of standard packets in bandwidth sharing. The problem is not relevant in non-congestion periods. Intuitively, in case of congestion, starvation should not happen as the number and size of SYN packets are relatively small (40 bytes) compared to standard packets. Another potential problem relates to the impact of protection on RED operations and properties. In fact, SYN packets are not checked against RED filters on their arrival. They can be seen as unresponsive and may raise or reinforce congestion. However, the same assertion about the number and size of SYN packets still holds. We think that the effects on RED performance are negligible or minor. These assumptions are validated by simulation results in the evaluation section.

We do not claim that the combination of the proposition and RED is the best one nor gives the best performance. However, this choice highly facilitates analysis and evaluation of the presented solution.

IV. Evaluation

This section presents the performance simulation results of REDFavor using ns-2 simulator. We look at 2 points:

- Latency of short-lived flows, that expresses the improvements brought by SYN packets protection,
- The counterpart of the observed improvements on long-lived flows.

We compare the performance of the SYN protection with RED and Choke. The simulation is designed to demonstrate the improvements in latency of the proposed SYN protection in a single bottleneck scenario. We adopt the model of web traffic developed in [15]. In this model, a pool of clients request web objects from a pool of servers. Pools are interconnected by a pair of routers and a bottleneck link. This link has a bandwidth of 10 Mbits/s as shown by Figure 1.

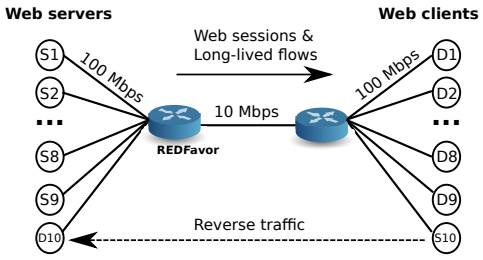

Fig. 1. Simulation model

In detail, we used TCP flows with RTT varying from 5 ms to 100 ms. A client makes a web session. A web session is composed by a sequence of web pages. A page is constituted by several objects. Each object is downloaded through a TCP connection. Then, each object will result in a new flow. The parameters used to obtain the simulated web traffic are the average of the number of pages per session, the average of the number of objects per page, the inter-session time, the inter-page time and the inter-object time, as seen in Table I. Exp(x) means the exponential distribution with mean x. Settings for the object size and the number of objects per page in Table I, are similar to those used by [16]. Consequently, we consider a Pareto object size denoted as P(1,1.2,12) where 1 is the minimum possible object size (in packets), 1.2 is the shape parameter, and 12 is the mean object size (in packets). The Pareto distribution shows high variability. It represents an accurate model of flow size distributions as

 ISBN: 978-1-61208-183-0

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Number of pages per session	Exp(240)
Number of objects per page	Exp(3)
Intersession (s)	Exp(0.5)
Interpage	Exp(5)
Interobject	Exp(0.1)
Object size	P(1, 1.2, 12)

empirically observed on the Internet. The settings for the remaining parameters applied to the web traffic model lead to usage of around 70% of the bottleneck link capacity. The web traffic load is generated by 135 web sessions taking place on each of the 9 web servers. Besides, each web server sends a long-lived TCP to one web client. A flow is generated in the reverse direction to mitigate potential synchronization between flows. REDFavor is applied only on the congested link. The simulation model is illustrated in Figure 1. Data are collected after a 100 second warm-up period. The simulation duration is set to 500 seconds.

A. Web traffic

This subsection evaluates the efficiency of REDFavor to improve the performance of short lived TCP flows. As mentioned earlier, the short flows are the most affected by the loss of initial TCP control packets, in terms of latency. The efficiency of SYN packet protection can be appreciated by a decrease in transfer delay.

Figure 2 shows the cumulative distribution of request completion time. The request completion time of a flow is the time interval starting when the first packet leaves that server and ending when the last packet is received by the corresponding client. RED experiences fewer sessions that terminate their requests within 3 seconds. A noticeable peak appears in 3 seconds with RED. This corresponds to the occurrence of initial RTO. The same observations are reported by [17]. REDFavor eliminates these earlier timeouts. It behaves and leads to the same results as Choke.

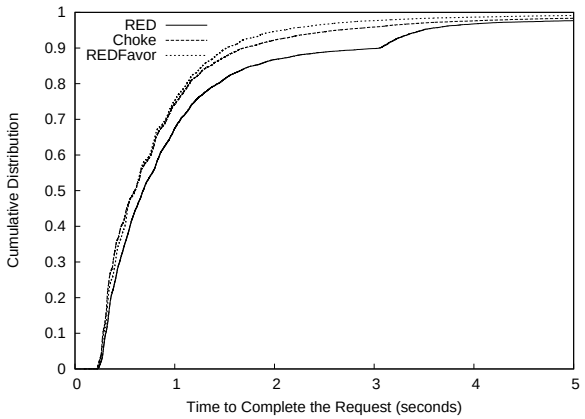

Fig. 2. CDF of completion times

Figure 3 presents the mean completion time as a function of the flow size. REDFavor ensures a lower transfer delay for both short flows and long flows. It performs like Choke with short flows and falls back to RED behaviour on large flows. These results prove that short flows benefit substantially from SYN packets protection offered by REDFavor while long flows are not penalized further than they would with RED.

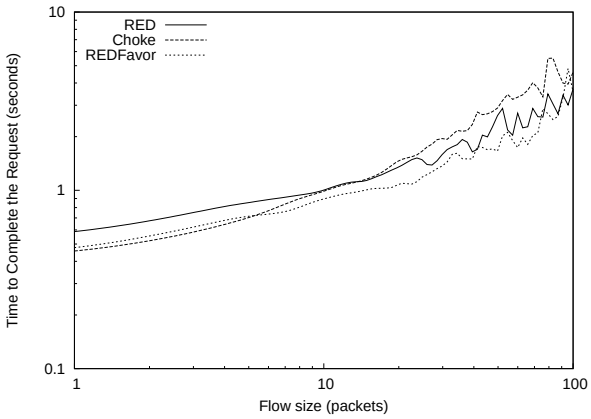

Fig. 3. Mean completion time

Drop protection is evaluated in Figure 4. This figure shows the distribution of dropped packet numbers on the congested link. We note that for flow sizes less than 10 packets, REDFavor has nearly the same behaviour as RED, then, it follows Choke. Quantitative results in Table II show that not a single SYN packet is dropped with REDFavor. Lesser standard packets are even lost compared to the two other schemes. So, protection is obtained at the expense of the loss of some standard packets, i.e., those with a higher packet number. This criticism should be moderate as the drop rate decreases. The obtained results show that REDFavor achieves the initial goal of SYN packet protection.

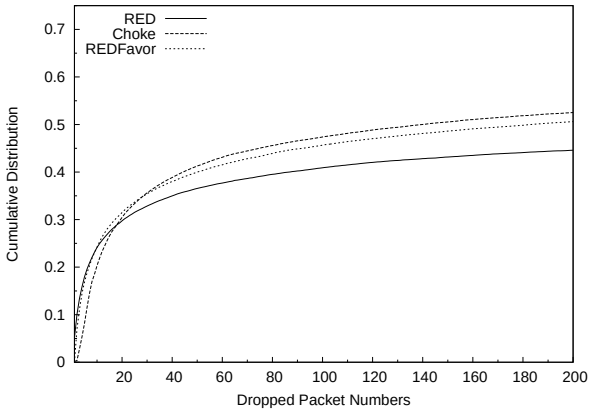

Fig. 4. Distributions of dropped packet numbers on the congested link.

B. Long-lived TCP flow

This subsection studies the impact of the proposition on long-lived flows. Let us remember that the throughput is the

TABLE II
DROPPED PACKETS COUNT

	Choke	RED	REDFavor
Dropped packets count	55566	36758	33150
Drop rate	0.082	0.143	0.073
Dropped SYN packets count	196	3391	0

TABLE III
BANDWIDTH USAGE

	Choke	RED	REDFavor
Bandwidth usage (%)	83.34	96.87	96.71

metric that matters for this type of traffic.

Figure 5 shows the normalized rate obtained by each of the 9 long-lived flows between a couple of web client and web server. With REDFavor, all large flows get the same throughput as in RED. This efficient use of bandwidth is confirmed by the quantitative results presented in Table III. We can note that Choke's improvements for short flows are obtained by a decrease in bandwidth share for long flows.

The impacts of SYN packets protection on long-lived flows are negligible. As stated previously, the "unresponsive" character of SYN packets (and short flows), has no impact on overall performance. Indeed, due to their small size, their participation in congestion occurrence is largely limited.

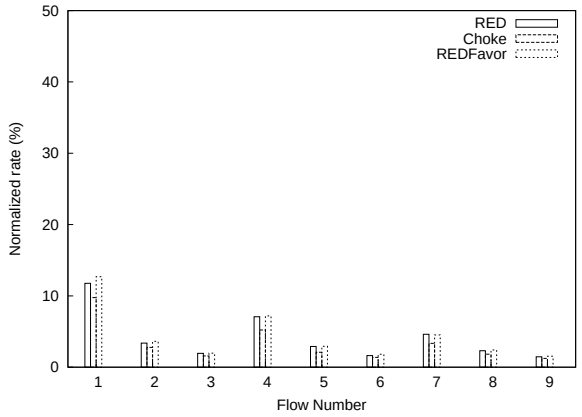

Fig. 5. Bandwidth by the long flows

V. CONCLUSION

This paper proposed a scheme consisting in protecting SYN packets and by the way, leverages the penalty of short flows. The proposition is relatively simple as it is implemented as an AQM scheme on a router's queue. Evaluations validate the idea while showing that noted improvements are not resulting in substantial impact on long-lived flows. A non-permanent (performed at the flow startup) and targeted action significantly improves short flow performance without a significant decrease in the throughput of large flows. Benefits are higher than costs.

The simplicity of the proposition constitutes its main advantage. Operations are solely based on the content of the packet's SYN flag. Deployment of the solution is transparent to end hosts as it involves routers (specifically, queue management), only. As REDFavor works independently, its deployment can be done in incremental manner, i.e., only on routers with heavily loaded links. When the congestion is not present, our AQM has no effect.

In operational aspect, a special attention should be paid to security concerns as the proposition relies on SYN packets identification. For example, it is vulnerable to SYN flood attacks. However, solutions to those security issues, such as firewalling, packet filtering or Intrusion Detection and Prevention Systems, exist and are fully functional. These solutions mitigate those security threats.

ACKNOWLEDGMENT

The authors would like to thank Emmanuel Lochin for earlier discussions on the subject.

REFERENCES

[1] D. Damjanovic, P. Gschwandtner, and M. Welzl, "Why is this web page coming up so slow? investigating the loss of SYN packet (work in progress)," in *IFIP NETWORKING*, Aachen, Germany, May 2009.
[2] J. Postel, "Transmission Control Protocol," RFC 793, Internet Engineering Task Force, September 1981.
[3] R. Braden, "Requirements for internet hosts," RFC 1122, Internet Engineering Task Force, October 1989.
[4] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," RFC 6298, Internet Engineering Task Force, June 2011.
[5] J. Chu, "Tuning TCP parameters for the 21st century," IETF 75 - Stockholm, Sweden, July 2009. [Online]. Available: http://www.ietf.org/proceedings/75/slides/tcpm-1.pdf
[6] M. Mellia, I. Stoica, and H. Zhang, "TCP-aware packet marking in networks with diffserv support," *Elsevier Computer Networks*, vol. 42, no. 1, pp. 81–100, 2003.
[7] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir, "Atlas internet observatory 2009 annual report," in *47th NANOG*, 2009.
[8] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, Internet Engineering Task Force, September 2009.
[9] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Internet Engineering Task Force, April 1998.
[10] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.
[11] R. Pan, B. Prabhakar, and K. Psounis, "Choke: A stateless aqm scheme for approximating fair bandwidth allocation," in *IEEE INFOCOM*. IEEE, 2000.
[12] S. Blake, D. Blak, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," RFC 2475, Internet Engineering Task Force, December 1998.
[13] A. Kuzmanovic, "The power of explicit congestion notification," in *ACM SIGCOMM*, Philadelphia, August 2005, pp. 61–72.
[14] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, Internet Engineering Task Force, September 2001.
[15] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *ACM SIGCOMM*, Vancouver, British Columbia, September 1999.
[16] I. Rai, E. Biersack, and G. Urvoy-Keller, "Size-based scheduling to improve the performance of short TCP flows," *IEEE Network*, vol. 19, no. 1, pp. 12–17, January 2005.
[17] D. Ciullo, M. Mellia, and M. Meo, "Two schemes to reduce latency in short lived TCP flows," *IEEE Communications Letters*, vol. 13, no. 10, October 2009.

 ISBN: 978-1-61208-183-0