

An Empirical Study of the Effectiveness of Fallback on the ECN

Linshi Zou[†], Chun-Xiang Chen[†]

[†]Faculty of Management and Information Systems,
Prefectural University of Hiroshima,
Hiroshima 734-8558, Japan
Email: {q922011rw, chen}@ed.pu-hiroshima.ac.jp

Kenichi Nagaoka[‡]

[‡]Dept. of Electronics and Information Engineering,
National Institute of Technology, Ishikawa College,
Ishikawa-ken, 929-0392 Japan
Email: nagaoka@ishikawa-nct.ac.jp

Abstract—ECN (Explicit Congestion Notification), as an effective approach for TCP congestion control, has been proposed for many years. However, the current network environment cannot provide particularly good support for ECN, so its deployment is much lower than expected. In order to bring the advantages of ECN into full play, a mechanism called Fallback is introduced to solve the problem that connection cannot be established after enabling ECN by default setting. In this paper, we evaluate the effectiveness of Fallback by using the data set of web domains as the test servers provided by Alexa. For the same test server, we first negotiate a connection with Not-ECN-Capable, and then negotiate a connection with ECN-Capable, if the sender does not receive the response to ECN-Capable negotiation from the receiver by the end of timeout, it will enter the Fallback state, and switch to negotiating a connection with Not-ECN-Capable. By extracting the header fields of the TCP/IP packets, we have found that the rates of Fallback are a bit different among the domains, the effectiveness of Fallback is shown on the low side, and the value of initial RTO (Retransmission Timeout) has a strong impact on Fallback.

Keywords – Congestion Control, Explicit Congestion Notification, Fallback, Effectiveness.

I. INTRODUCTION

Based on the AIMD (Additive-Increase/Multiplicative-Decrease) principle, many congestion control algorithms have been proposed, such as TCP Tahoe, TCP Reno, TCP NewReno [1], CUBIC [2], which are based on packet loss, and TCP Vegas [3], which is based on delay, and BBR [4], which is based on delay and the bandwidth of the bottleneck link. However, packet loss will cause the waste of network resources and deteriorate the performance of network service, and algorithm based on delay is less aggressive than that based on packet loss, so fairness will be affected. To solve these problems, another approach called ECN has been proposed.

ECN is an extension of the function in the network intermediate node. Combining ECN with AQM (Active Queueing Management) [5], the sender can be notified of the impending congestion in the network by marking packets instead of discarding them, then the sender can actively slow down the transmission rate of TCP to improve the network performance.

Although ECN has been proposed as a very effective congestion control approach for many years, its application status is not as expected. The reason is that ECN works best only

when all routers along the path between sender and receiver can support ECN. However, there are still many older routers which are incapable of identifying the IP header supporting ECN. Besides, [6] found the presence of a nontrivial number of paths that mangle the ECN field in IP header. Also, The establishment of TCP connections supporting ECN might be blocked in some security gateway or firewalls along the path [7]. Therefore, if ECN is enabled as default setting at the sender, it is likely that some connections cannot be established, which is not what we want. To solve this problem, a mechanism called Fallback was proposed [8]. The main principle of this mechanism is that the sender negotiates a connection with ECN-Capable first and if it does not receive the response to ECN-Capable negotiation from the receiver by the end of timeout, it will enter the Fallback state, and switch to negotiating a connection with Not-ECN-Capable like the normal way.

[9] analyzed the connectivity of connections negotiated with ECN, but did not take Fallback into account. In this paper, we evaluate the effectiveness of Fallback as well as how the initial RTO will affect the behavior of Fallback.

The rest of this paper is organized as follows. Section II describes data set and methodology of measurement. Section III presents the analytical results. Concluding and outlook are given in Section IV.

II. DATA SET AND METHODOLOGY OF MEASUREMENT

A. Data set

Different target web servers may cause big deviation in the results, so should be carefully chosen. We used the web server entries provided by Alexa [10]. Alexa collects more than 1000GB of information on the Internet every day around the world to count website visits. Therefore, the domain entries provided by Alexa can well meet the data requirements of our research.

We downloaded the data released by Alexa on June 12 2019, which included one million domains of web servers. After removing the domains which have same IP address, there were 508,122 domain entries left. We took the top 100,000 domains in unique domain entries and all these 508,122 domains as the test servers in June and December, respectively. If a web server has IPv4 and IPv6 addresses, we performed two tests.

Considering that the test location may have an impact on the results, we tested the data in June in three different places: PUH (Prefectural University of Hiroshima, Japan), NIT,IC (National College of Technology, Ishikawa College; Ishikawa, Japan), Chongqing China. And the data in December was tested in PUH and NIT,IC.

B. Methodology of Measurement

Two kinds of hosts are used in this work (CentOS 6.10 with the customized Linux Kernel 4.9.80 and Fedora Core 30 supporting ECN and Fallback) to perform the testing.

First we test the situation when Fallback is disabled, the procedures are as following (The procedures of IPv6 are same to IPv4). Note that, to collect TCP/IP packets, `tcpdump` must be started and terminated before and after the testing.

- 1) Disable Fallback by setting the kernel parameter `net.ipv4.tcp_ecn_fallback=0`.
- 2) Disable ECN by setting `net.ipv4.tcp_ecn=0`.
- 3) Pick up next entry (as server A) from the entries of the data set; Negotiate an http connection (TCP port 80) with server A by setting flag SYN (SYN only). If the negotiation succeeds, issue a request to load top page from server A; Close the connection.
- 4) Enable ECN by setting `net.ipv4.tcp_ecn=1`.
- 5) Negotiate an http connection (TCP port 80) with server A by setting flag SYN (SYN, ECN, CWR). If the negotiation succeeds, issue a request to load top page from server A; Close the connection.
- 6) Repeat the above procedures until all web servers in the entries are visited.

Second we test the situation when Fallback is enabled, most of the procedures are the same as above. Here we only show the different procedures 1 and 5.

New1. Enable Fallback by setting the kernel parameter `net.ipv4.tcp_ecn_fallback=1`.

New5. Negotiate an http connection (TCP port 80) with server A by setting flag SYN (SYN, ECN, CWR). If the negotiation succeeds, issue a request to load top page from server A, then close the connection; If the sender does not receive the SYN-ACK packet from the receiver by the end of timeout, it goes to the Fallback state, and does the procedures 2 and 3 (but test the same domain) again, then finish the test to this web server.

III. ANALYSIS AND RESULTS

we extracted the key information we need from the header fields of TCP/IP packets by the tool we developed. In this section, we present our analytical results.

A. Fallback rates among different domains

We use the data in June with the initial RTO set to 1s, and get the Fallback rates of IPv4 web servers and IPv6 web servers in all test domains as shown in the Figures 1 and 2, respectively (the environment of IPv6 network was unavailable in Chongqing China, so there is no IPv6 results in

Figure 2). Let N_{noecn} be the number of the test servers, which reply to the sender with Non-ECN-setup SYN-ACK packet (i.e. Not ECN-Capable) after receiving the ECN-setup SYN packet, and N_{fb} be the number of the senders, which enter the Fallback state. So the Fallback rate can be calculated by $\beta_{fb} = N_{fb}/N_{noecn}$. Note that as long as the sender receives the response from the receiver (whether it is ECN-setup SYN-ACK or Non-ECN-setup SYN-ACK packet), it does not enter the Fallback state.

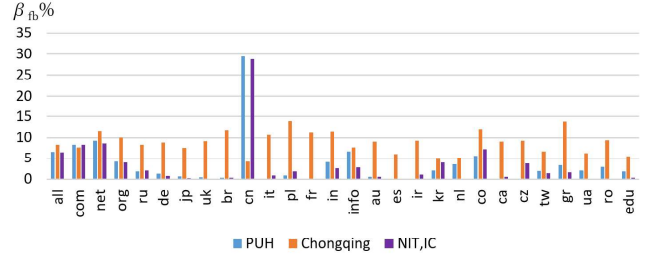


Fig. 1. Fallback rates, IPv4

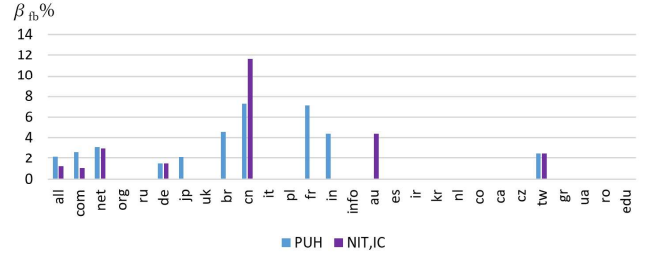


Fig. 2. Fallback rates, IPv6

Figure 1 shows that the Fallback rates among different domains are a bit different. The results tested in PUH are almost consistent with that in NIT,IC, and it can be seen that the Fallback rate in domain `cn` is abnormally high (close to 30%), which is twice higher than that in domain `net` with the second highest rate of Fallback. And it is strange that in the results tested in China, the rate of Fallback in domain `cn` is the lowest, while the rates of Fallback in other domains are much higher than the results tested in Japan. Since the domains we tested are exactly same, we suspect that the possible reason for this phenomenon may be due to the fact that some data packets going to cross-border access need to be inspected in gateways in China, so the RTT (Round-Trip-Time) is increased, leading to partial senders cannot receive SYN-ACK packet from the receiver by the end of timeout after sending ECN-setup SYN packet, so the rate of Fallback is increased. On the other hand, since most of the `cn` domain servers are distributed in China geographically, the data packets from the domestic connections do not need to go through such gateways, so there is no extra checking time, which makes Fallback rate low.

At the same time, we can also see from Figure 2 that IPv6 and IPv4 have the same features, but the Fallback rate of IPv6 is generally smaller than IPv4.

B. The impact of Fallback on connectivity

We use the data in December with the initial RTO set to 1s, and get the relationship between Fallback and connectivity

of IPv4 web servers and IPv6 web servers in all test domains as shown in the Figures 3 and 4, respectively. And the results tested in NIT,IC are consistent with the results in PUH, in order to save space, we only show the results tested in PUH.

Let T_{no} be the number of the web servers, which reply to the negotiation for establishing connection with SYN-ACK packet and N_{11} be the number of the web servers, which reply ECN-setup SYN-ACK packet if they are also ECN-Capable otherwise reply Non-ECN-setup SYN-ACK packet after receiving ECN-setup SYN packet from the sender. So the connectivity can be calculate by $\beta_{11} = N_{11}/T_{no}$. It can be seen from Figure 3 that before Fallback is enabled, all domains except `cn` have a very high connectivity (close to 100%), and IPv6 is higher than IPv4. When Fallback is enabled, the connectivity of these domains has been improved, even though the rate of increase is very low. As regards domain `cn`, it has lower connectivity than other domains both in IPv4 and IPv6. In contrast, the connectivity has been greatly improved while Fallback is enabled, and IPv6 has been improved more than IPv4, but the increased value is still lower than the value in other domains when ECN is disabled.

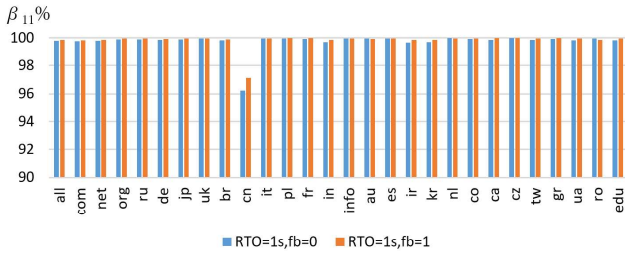


Fig. 3. Connectivity of ECN-Capable servers, IPv4, PUH

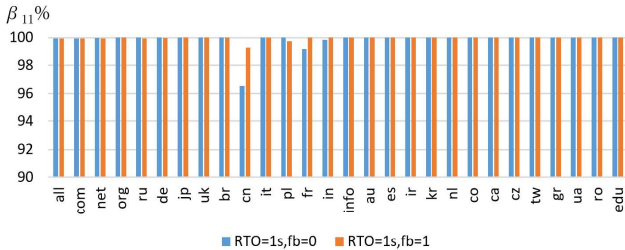


Fig. 4. Connectivity of ECN-Capable servers, IPv6, PUH

C. The impact of Fallback on the total time spent on establishing a connection

In order to find out the impact of Fallback on the total time spent on establishing a connection (we call it establishing time in the following), we first filtered the statistical data in December which the initial RTO is set to 1s. For convenience, we give some definitions of the symbols as follows.

ETT1 the establishing time from sending ECN-setup SYN packet to receiving the response, when Fallback is disabled.

ETT2 the establishing time from sending ECN-setup SYN packet to receiving the response, when Fallback is enabled.

N_0 the number of connections that receive the response from the receiver after sending ECN-setup SYN packet, when Fallback is disabled (i.e. ETT1 has a value).

N_{00} the number of connections on which ETT1 has a value bigger than the initial RTO (1000ms), when Fallback is disabled.

N_1 the number of connections which dose not receive the response after sending ECN-setup SYN packet by the end of timeout, then enter the Fallback state, but receive the response from the receiver after sending Non-ECN-setup SYN packet, when Fallback is enabled.

Note that the ETT2 of the connections in N_1 includes two periods. One is the initial RTO, because these connections entered the Fallback state. The other is the time interval from sending Non-ECN-setup SYN packet to receiving Non-ECN-setup SYN-ACK packet.

N_{01} the number of connections that both in N_0 and N_1 (i.e. domain is same).

N_{001} the number of connections that both in N_{00} and N_1 .

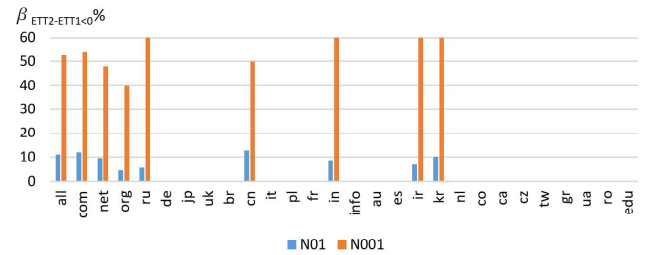


Fig. 5. The impact of Fallback on establishing time, IPv4, IPv6, PUH

In Figure 5, the blue results indicate the rate of the connections with $ETT2-ETT1 < 0$ in N_{01} (noted as $\beta_{ETT2-ETT1 < 0}$), and the orange results indicate the rate of the connections with $ETT2-ETT1 < 0$ in N_{001} . We can see from the blue results, as long as the sender entered the Fallback state, the establishing time (ETT2), has a very high probability (around 90%) bigger than the time from negotiating a connection in the normal way (i.e. Not ECN-Capable) to receiving the SYN-ACK packet (ETT1), so it can be concluded that Fallback will increase establishing time.

Interestingly, it can be seen from orange results that in N_{001} the rate of establishing time reduced after entering Fallback state ($ETT2-ETT1 < 0$) is abnormally high. In order to find the reason for this phenomenon, we chose the domains `com`, `net`, and `cn`, which have more connections in N_{001} and the number is 261, 23, and 82, respectively. We extracted the original TCP/IP packets related to the 366 (261 + 23 + 82) connections, and analyzed them one by one. We found that when Fallback is disabled, these connections did not receive SYN-ACK packet by the end of timeout, they all repeated sending ECN-setup negotiations for establishing connection, with the average number of negotiations is 2.23. We confirmed that, in the host used for testing, the maximum number of

negotiations for establishing a connection to the same server is 3, and the backoff interval of RTO is exponential, i.e. 1s, 2s and 4s in order. This internal setting let the ETT1 be bigger than the ETT2 in some connections.

D. The impact of the initial RTO value on Fallback effectiveness

We use the data in December with the initial RTO set to 1s and 3s, and get the relationship of the initial RTO value on Fallback effectiveness in Figures 6 and 7 as following.

Figure 6 shows that for all the domains, the rates of Fallback when the initial RTO=1s is almost higher than that when the initial RTO=3s. This result is consistent with the theory. There are some connections with ECN-Capable negotiation, in which the receiver replied SYN-ACK packet, but sender did not receive it by the end of timeout, so it entered the Fallback state. Setting the initial RTO = 3s will give SYN-ACK packet more time to return to the sender, so the Fallback rate will be reduced.

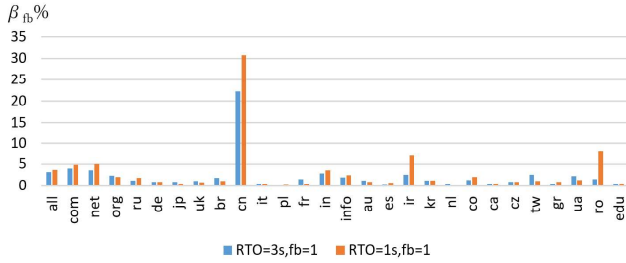


Fig. 6. The impact of the initial RTO value on Fallback rates

At the same time, the difference between this two results can be considered as the rate of connections with RTT greater than 1s. The Appendix of RFC 2988 shows that more than 97.5% of the connections observed in a large scale analysis were less than 1 second [11]. But from our statistical results, we can see that domains *cn*, *ir* and *ro* have 8.5%, 4.8%, and 6.7% of connections are greater than 1s. Therefore, changing the initial RTO value from 3s to 1s proposed in RFC 2988 may have a negative impact on the access to these domains.

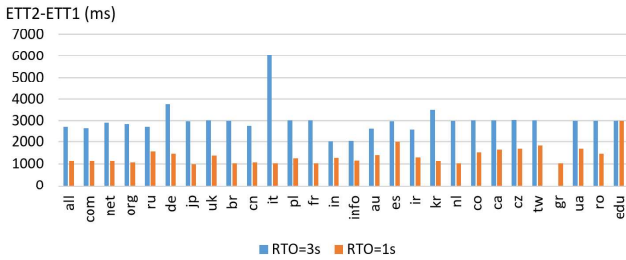


Fig. 7. The impact of the initial RTO value on establishing time when Fallback

Figure 7 illustrates the average value of ETT2-ETT1 to the connections in N_{01} (N_{01} is in section III-C). It shows that the total time using ECN-Capable to negotiate a connection first and switched to using Not-ECN-Capable after entering the Fallback state is averagely one initial RTO value greater than the time using Not-ECN-Capable to negotiate a connection

directly. Similarly, the longer the initial RTO is set, the larger this extra time will be.

IV. CONCLUDING AND OUTLOOK

In this paper, we evaluated the effectiveness of Fallback by using the data set of web domains as the test servers provided by Alexa. For the same test server, we first negotiated a connection with Not-ECN-Capable, and then negotiated a connection with ECN-Capable, if the sender did not receive the response to ECN-Capable negotiation from the receiver by the end of timeout, it would enter the Fallback state, and switched to negotiating a connection with Not-ECN-Capable. By extracting header fields of the TCP/IP packets, (1) We obtained the rates of Fallback among different domains and found that they are a bit different, and the Fallback rate in domain *cn* has opposite features to other domains. (2) We analyzed the impact of Fallback on connectivity and establishing time, and found that connectivity is slightly improved after Fallback is enabled, also as long as the sender entered the Fallback state, the total time spent on establishing a connection increased one initial RTO value averagely. So we considered that the effectiveness of Fallback is on the low side. (3) We explored the impact of the initial RTO value on Fallback effectiveness, and found that the value of initial RTO will strongly affect the rate of Fallback in domain *cn*, *ir*, and *ro*, as there is still a relatively large rate of connections with RTT greater than 1s in these areas.

Due to the existence of Fallback, effective congestion control approach (ECN) can be tried first without loss of the connectivity. Also as it is said in [8], Fallback can reduce the loss resulting from the persistent misuse of ECN. The superiority of using ECN over common congestion control approaches and the correction effect on misused ECN should be considered as the effectiveness of Fallback, this two points are our future works.

REFERENCES

- [1] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 2582, April 1999.
- [2] I. Rhee, L. Xu, S. Ha, et al. "CUBIC for Fast Long-Distance networks," RCF 8312, Feb. 2018.
- [3] L.S. Brakmo and L.L. Peterson, "TCP Vegas: End to END Congestion Avoidance on a global Internet," IEEE J. Sel. Areas Commun., vol.13, no.8, pp.1465-1480, Oct. 1995.
- [4] Mark Claypool, Feng Li, and Jae Chung, "BBR' - An Implementation of Bottleneck Bandwidth and Round-trip Time Congestion Control for ns-3," Technical Report WPI-CS-TR-18-01, Computer Science, Worcester Polytechnic Institute, Jan. 2018.
- [5] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, July 2015.
- [6] S. Bauer, R. Beverly, and A. Berger, "Measuring the State of ECN Readiness in Servers, Clients, and Routers," Proc. 11th ACM SIGCOMM/USENIX Internet Measurement Conference (IMC 2011), pp.171-180, Berlin, DE, Nov. 2011.
- [7] S. Floyd, "Inappropriate TCP Resets Considered Hamful," RFC 3360, Aug. 2002.
- [8] G. Fairhurst and M. Welzl, "The Benefits of using Explicit Congestion Notification (ECN)," RCF 8087, March 2017.
- [9] Chun-Xiang Chen and Kenichi Nagaoka, "Analysis of the State of ECN on the Internet," IEICE vol.E102.D no.5, pp.910-919, May 2019.
- [10] <https://s3.amazonaws.com/alexa-static/top-1m.csv.zip>
- [11] V. Paxson, M. Allman, J. Chu and M. Sargent, "Computing TCP's Retransmission Timer," RFC 6298, June 2011.