Vol.2016-GN-99 No.14
Vol.2016-SPT-18 No.14
2016/5/13

Emulation Evaluations of Tor Schedulers with Active Circuit Switching

Timothy Girry Kale Satoshi Ohzahata Celimuge Wu Toshihiko Kato

Graduate School of Information Systems, The University of Electro-Communications

1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan

E-mail: tgk@net.is.uec.ac.jp, {ohzahata, clmg, kato}@is.uec.ac.jp

Abstract: The Tor network is an open network that helps its users defending against adversaries who performed traffic analysis and threatens the personal privacy and confidential security. Tor protects its users by preventing the sites they visit from revealing their physical location. This trust of services raises the number of Tor users daily, which makes Tor become one of the widely used privacy networks today. However, as the number of Tor user increases, the performance of Tor degrades badly due to Tor's scheduler does not fairly distribute the traffics in Tor. In this paper, we highlight the problems in Tor performance due to scheduling design of Tor, which degrades the web and bulk traffic performances that multiplexes on a single TCP connection. To solve these problems, this paper emphasis on applying the simple method of active circuit switching on Tor application level, in order to distribute the bulk and web traffics through different TCP connections when congestion is detected in Tor. We evaluate our modified switching with the other default schedulers in Tor, such as vanilla and priority EWMA scheduler through a wide deployment network in shadow emulator. The experimental results show that our switching technique considerably offers stable improvement of traffic performance for all the web and bulk traffics.

Keywords: Tor, Vanilla, EWMA, Circuit Switching, Anonymity, Onion Router (OR), Onion Proxy (OP).

1. Introduction

The Tor network [1] is the most widely used anonymous network today, which comprises of over ten thousands onion routers (ORs) around the world [2]. Tor serves millions of users privacy on the Internet by encrypting traffics through multi-hop ORs, which are communicating using pairwise TCP connections [3]. However, despite the increase adaptation of Tor, it is well known that Tor does not fairly distribute the bulk and web traffics through widespread ORs [4, 5, 6]. This problem contributed to degrading the overall network performances for the majority of Tor web users who heavily depend on Tor anonymity services, and further degrading the security of Tor. There are wide ranges of studies [4-12] that recognized these as problems and confirmed that the limitation of Tor's bandwidth resources, relay selection and Tor scheduling [11] are the primary sources for the unfairness of traffic distribution in Tor [12].

To solve the previous problems, Tang and Goldberg [5] implemented their method of priority exponentially weighted moving average (EWMA) schedule in Tor, which calculating the EWMA of cells, and measures the recent activity of a circuit. They applied this scheduler to give higher priority to circuits that have lower EWMA value, which classified as web interactive traffics, to have their cells transferred first over the bulk traffic. Their [5] algorithm keeps track of a number of cells schedules for each circuit, and after a configurable time-period of "halflife" reached, the cell counts are exponentially decayed. This approach looks promising however, Jansen and Hopper [6] tests the EWMA priority scheduler in a similar setup, which runs in a shadow emulator environment, and they concluded that the priority EWMA scheduler does not applicable under all network conditions, since it is not clear that performance will always improve. Therefore, more experimentation and improvement is needed to confirm the effectiveness of EWMA priority scheduler though it is now used as default scheduler in Tor [5, 6].

In the first implementation of Tor, the OR was designed to use the round-robin TCP read/write vanilla scheduler [6], which has been realized in [12, 13] that this scheduler also contributed to the unfairness of traffic distribution for web and bulk clients. The bulk traffics tend to have higher priority over the web traffics, which discourage the wide adaptation of the Tor network. This gives us the reason to focus on implementing an alternative approach to improve the current scheduler of Tor architecture.

Our primary goal is to improve the traffic performance of the entire web and bulk Tor users, by implementing a simple method of active circuit switching schedule on Tor application level [14]. In our previous work [14], we did not address the dynamic distribution of traffic on Tor and we did not compare the modified switching algorithm with other schedulers in Tor.

In this paper, we improve the active switching algorithm to dynamically switch the bulk circuit traffic that causes the congestion in the network to different TCP connection. We used the buffer occupancy information that is local to the guard entry ORs to detect congestion, and dynamically switching the circuit with higher EWMA values to a new path that has higher capacity, in order to reduce the bottleneck in the previous TCP connection that is highly congested. The contributions of this paper are stated below.

- We evaluated the active circuit switching schedule algorithm and compared with the un-prioritized vanilla Tor and prioritize EWMA scheduler in the shadow emulator (simulation environment of Tor wide deployment that depicts the real live Tor network) [6].
- The designed of modified circuit switching [14] adopts the current design of Tor EWMA algorithm, however, we performed a small modification of EWMA algorithm to

monitor per-connection for a number of cells occupies the output buffer occupancy in the Tor application level.

- The proposed active switching method considerably more effective, since the configured control metrics of maximum and minimum thresholds, helps to quickly detect the congestion in Tor and actively distribute the traffics compared to priority EWMA and non-priority vanilla schedulers

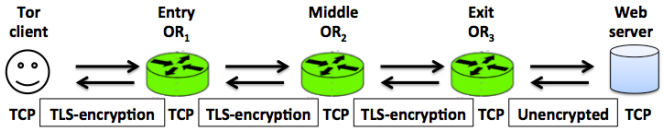

Figure 1. The Tor network architecture.

2. Background of the Tor Network

The Tor network is a second generation of the onion routing overlay network [1]. The Tor users first built it in the US Naval laboratories with a goal of preventing hackers from discovering a physical location and content access.

Figure 1 shows the architecture of the Tor network. The Tor client locally runs software, which called an onion proxy (OP). The OP presents a SOCKS proxy interface to local applications, such as the web browsers. When an application makes a TCP connection to the OP, the OP splits the TCP segment into 512 bytes of fixed-size cells, and forwards them over the Tor network.

To construct a circuit, the client (OP) picks a first entry OR_1 and makes a TCP connection as well as the transport layer security (TLS) encryption on that connection. The client OP then instructs the entry OR to connect to a middle OR_2. After that, the OP instructs the middle OR_2 to contact the exit OR_3 and the TCP connection is made between the middle OR_2 and the exit OR_3. The last exit OR_3 performs decryption of circuits from the middle OR and opens the unencrypted TCP connection to the web server. After the connection to the web server is made successfully, the exit OR_3 reports a one-byte status message back to the client application proxy [1]. Note that by default, Tor uses three hops circuit on the Tor network. For the communication protocol over Tor, all the ORs are communicating using pairwise TCP connections. Tor was designed to multiplex multiple circuits over a single TCP connection to enhance its anonymity services. The multiplexing of circuits to single TCP connection can carry traffic for multiple services or streams that a user may be accessing[1]. When the data arrives at the OR, it is immediately processed as it arrives in connection input buffers [1] and each cell is either encrypted or decrypted depending on its direction through the circuit. The cell is then switched onto the circuit corresponding to the next hop and placed into the circuit's first-in-first-out (FIFO) queue [4]. Cells wait in the circuit queues until the circuit scheduler selects them for writing. Note that the reading and writing of data operations are based on round-robin manner across input and output sockets, with a scheduler that controls the execution in Tor.

3. The Circuit Schedulers in Tor

3.1 Vanilla Tor (Non-Priority) Scheduler

The vanilla Tor scheduler represents the primary settings in the Tor application and it is the unmodified algorithm that uses the round-robin circuit scheduler [15, 16]. Vanilla scheduling is responsible for controlling the execution of reading and writing operations of cells in a round-robin manner across input and output sockets when there is space available [15]. Note that the writing to sockets is scheduled alongside reading from sockets, and so is performed in a round-robin manner when data is available. The Tor buffers operate with first-in-first-out (FIFO) order and queue on the circuit until the circuit scheduler selects them for writing. Furthermore, taking a data from the output buffer and dispatch to the network is happening in a single thread scheduling for reading, processing and dispatching (writing) to a socket. This means all work is done serially for vanilla Tor scheduler [16].

3.2 EWMA (Priority) Scheduler

The exponentially weighted moving average (EWMA) scheduler was recently introduced into Tor [5] and is currently used by default as an update version of Tor algorithm. The EWMA scheduler records the number of cells it schedules for each circuit and, exponentially decaying cell counts over time (configured "halflife" parameter). The EWMA priority scheduler writes one cell at a time chosen from the circuit with the lowest cells count. The cell counter of the connection to which that circuit belongs cannot affect other per-connection EWMA [5], i.e., per-connection EWMA is enabled and configured independently of its circuit counterpart. Every time when cells are ready to transfer to connection's output buffer, the EWMA algorithm calculates the decayed cell count value for each circuit based on the EWMA equation in [5]. After the calculation and OR pick the circuit with the smallest cell count value, the cell count value is updated correspondingly. The detail of priority EWMA equation and algorithm are further explained in [5].

3.3 Modified Circuit Switching Scheduler

We implemented the active circuit switching algorithm in [14], which focuses on detecting the congestion in the network at the guard entry OR, and switching of an active bulk circuit with higher EWMA value to different TCP connection. In this paper, we improved the circuit-switching algorithm to dynamically switch the bulk traffic to different TCP connection, in order to reduce the network congestion for all clients. Unlike the existing techniques of vanilla and priority EWMA schedulers that previously explained, our simple active switching techniques use the buffer occupancy information that is local to the guard entry OR.

In addition, this dynamic switching technique is essential and effective to solve the problems of Tor's performance with respect to increasing queuing delays in Tor and TCP kernel buffers, increasing memory usage and node's inability to receive more data (socket un-writable) [17]. These problems are contributed to increasing the latency on the Tor network.

The next subsection further explains the control metrics applied to our modified switching algorithm to reduce the queuing cells in Tor and TCP kernel buffers.

3.3.1 Modified Circuit Switching Control Metrics

Since we focused on the buffer occupancy information that is local to guard entry OR, we applied a statistical algorithm for scheduling traffic on circuit connecting to Tor output buffer using the EWMA algorithm.

Equation 1 shows our modification on the EWMA algorithm to count the number of cells entering the output buffer in Tor. The

[1] To easily identify different circuits, the OR can assign different circuit ID to each circuit.

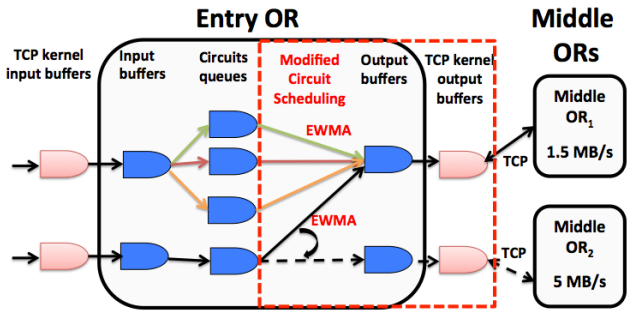

Figure 2. The designed of modified circuit scheduling in the guard entry OR.

periodic sample of instantaneous output buffer occupancy is represented as B for each incoming circuit in Tor.

$$B_t = \gamma * B_{t-1} + (1-\gamma) * B_{sample} \qquad (0 < \gamma < 1) \qquad (1)$$

The B_{sample} is the currently collected number of sample cells sent from the circuit queues and occupied the output buffer in Tor. The EWMA parameter γ is the fractional weight of the previous buffer estimated in the EWMA estimator. We used it to estimate the current buffer occupancy and determine the depth of kernel memory usage in the allocated output buffers in Tor. The larger γ has a greater influence on the number of cells occupying the output buffer by the incoming circuits. Whenever the circuit's cell counter is incremented, the cell counter of the outgoing connection to which that circuit belongs is also incremented. Our control scheme is based on this incremented value to identify which circuit contributed to overflowing the output buffer.

3.3.2 How Modified Circuit Switching Scheduler Works

Figure 2 shows the modified circuit scheduling that runs in the guard entry OR, which estimated the EWMA periodic samples of instantaneous output buffer occupancy for each circuit. If congestion occurs, i.e., if the output buffer queues rise above the limit, then the incoming bulk traffic with higher EWMA value dynamically switch to a different output buffer in Tor and TCP socket as illustrated by the dashed arrow line. The dashed arrow line indicates the newly switched circuit to route the bulk traffic. The solid arrow line indicates the initial circuit route. The red dash-square shows the focused area of control metrics for the output buffer occupancies in Tor and TCP socket. This method is a backward compatible with the existing Tor network, which the TCP header is secured to avoid the disclosure of per-connection data transfer through the old and newly switched circuit of each TCP connection. After the active bulk traffic switched to a new circuit that was preemptively built by the Tor client, the traffic is routed to a higher bandwidth ORs with an aim of improving the traffics flows. Note that any packet drops on the newly TCP connection will not affect the flow of traffic in the previous old TCP connection. This approach lowers the queuing packets in the congested output buffer in Tor and TCP connection.

The detail procedures involving the active circuit switching of bulk traffic, and the assignment of transferred cells to the newly circuit are explained in our previous work [14].

3.3.3 Buffer Thresholds

We defined the two threshold values to control the output buffer occupancy in Tor to ensure both bulk and the web traffics have a fair share of network resources. The two thresholds are α and β, which indicates the minimum and maximum buffer thresholds respectively. Data occupying the output buffer below α is considered as the buffer is under full and above β is considered as buffer overfull. If the current buffer occupancy is within the acceptable range between the minimum and maximum threshold, no switching of bulk traffic occurs. However, if the Tor output buffer rises above the maximum threshold β, switching of bulk traffics with higher EWMA value occurs. This approach allows the circuit traffics with lower EWMA value (the web traffics) to have better performance through the previous congested TCP connection. We configured the $\alpha = 0.1$ and $\beta = 0.5$. These threshold values have smaller circuit switching time of less than 1 second. Note that these threshold parameters are fractional of the total allocated memory for each output buffer size in Tor, which is 32 KB.

4. Experiment Setup

4.1 Wide Network Deployment in the Shadow Emulator

We setup the modified circuit switching scheduling with the priority EWMA and the vanilla schedulers in the shadow emulator [6]. The shadow emulator is more efficient and scalable discrete event simulator, that can run a multiple threads and processes during execution of parallelizing simulation workloads with high accuracy [5, 6]. We configured a scalable and efficient design topology in the shadow emulator to easily run the priority EWMA scheduler, non-priority vanilla scheduler and modified switching algorithm.

4.2 Shadow Limitation

The shadow emulator can simulate the similar scenario of the entire Tor network deployment, however, there are limitations to this emulator in terms of providing accurate results of specific kernel features or kernel parameter settings. And since our proposed active circuit switching method focuses on dynamic routing changes in the Internet, the shadow emulator is limited and does not precisely model these behaviors. These problems also hinders our full evaluation in the shadow emulator, while priority EWMA and vanilla schedulers had already been implemented and tested in the shadow emulator [5, 6, 15].

To overcome the limitation we faced, we configured the traffic generator that currently exists in the shadow simulator repository [18], by building traffic generator (TGen) as an external tool and skip building both the full simulator and the plug-in as part of TGen. This configuration enables our modified circuit switching algorithm to runs well with shadow emulator environment and specifies the structure of the network topology, and network properties such as processing packets, circuit queues, transfer bytes and downloading times. Furthermore, this newly configuration of shadow TGen approach enable our proposed method to directly configured in the guard entry ORs, which are located in the shadow emulator and connecting to the middle and exit ORs outside of shadow emulator, i.e., over the Internet and the results are collected (mirrored) in the shadow simulation log files.

4.3 Shadow Configuration

Since shadow can run the entire Tor network on a single Linux machine, we setup the network topology in the Ubuntu 14.04 LTS, 64-bit, 16GB RAM, Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz

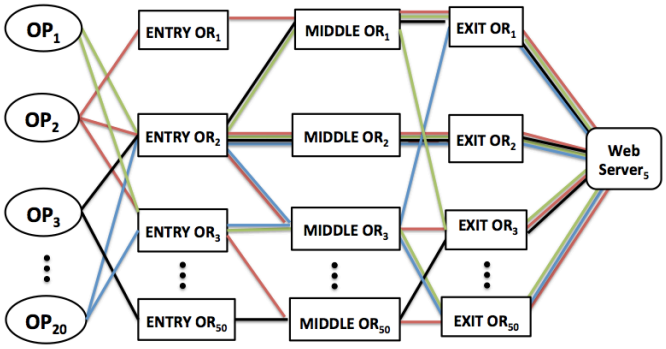

Figure 3. Topology setup in the shadow emulator.

to analyzed the processing packets, circuit queues, transfer bytes and downloading times. We used the scallion plugin [6, 19] that contains XML scripts to assist us in generating the topologies and running Tor experiments in shadow to collect results.

Figure 3 shows the configured topologies setup in the shadow emulator. We configured 5 HTTP servers, 150 ORs (50 entry ORs, 50 middle ORs and 50 exit ORs) and 20 clients (10 web clients and 10 bulk clients) for our experiments. To analyze the effects of various network loads on each scheduler running in the shadow emulator, we configured each bulk client to repeatedly download a 5 MB file from a randomly selected HTTP server without pausing, while each web client download a 320 KB file from a randomly HTTP servers but pauses after each download completed, before downloading the next file. In this way, the bulk users are actively using the network and the responsiveness of bulk traffics are faster because of continuous traffics. In the experiments, Tor relays are configured with bandwidth parameters according to the Tor network consensus bandwidth. All the ORs, clients, and web servers have 10 MB connections. We configured the EWMA scheduler with the "halflife" parameter of 66.

In addition, for modified switching scheduler, we directly insert our modified algorithm in the shadow guard entry OR as previously mentioned. Since the modified circuit algorithm can dynamically switch the bulk traffic outside of shadow emulator, we setup other ORs in our testbed so that the bulk traffic can pass through other middle and exit ORs after switching to a new circuit. This approach makes it possible to reconnect to the shadow HTTP server that the old circuit was previously used.

4.4 Measurements

In the experiment measurements and analysis, we focus on the time to first byte that received by the web and bulk clients, and the time when the last byte arrived for downloading of 320 KB (web client traffic) and 5 MB files (bulk client traffic). Note that the time to first byte indicates the network responsiveness of the web and bulk traffics that routed in the Tor network.

When testing the modified circuit switching schedule that running on the entry ORs, we measured the time to first byte received and download time of traffics at the beginning when all the circuits are multiplexed to single TCP connection (before switching) and, after the bulk circuit switched to different TCP connection. All the measurements results, including cells processing inside the Tor application level and events of all circuit traffics for each scheduler, are obtained by shadow and outputted into the log files when the experiments are done.

5. Experimental Results

The experimental results show how each scheduler running on the shadow emulator maximizes the web and bulk traffics under variances of network loads.

Figure 4 (a) and (b) shows the client performances for our modified switching scheduling compared to non-priority vanilla and priority EWMA schedulers, for time to first byte received. Figure 4 (a) show the modified switching algorithm reduces the web time to first byte for all web clients by 90% improvement, over vanilla Tor and priority EWMA scheduler with "halflife" parameter 66. In Figure 4 (b), the bulk time to first byte measurements shows our modified switching algorithm has better performance over EWMA and vanilla Tor scheduler. The switching of bulk traffic to the new alternative higher capacity of TCP connection improves the performance for our proposed switching method.

In Figure 4 (c), the EWMA scheduler seems to be effective as modified switching algorithm for less than 40% testing of web download traffic over vanilla Tor. However, 60% of experiment testing shows that modified switching algorithm still improves the overall web download traffics. Interestingly, in Figure 4 (d), the EWMA 66 and vanilla Tor schedulers seem to degrade very deficiently and has unstable performance compared to modified switching algorithm for bulk traffics. The un-priority round-robin vanilla Tor scheduler seems to be most affected in all cases for the web and bulk download times.

The experiment results in Figure 4 (b) and (d) show that our method of actively switching the bulk traffic to different TCP connection gain significant improvement when there is congestion detected in the guard entry OR. Both web and bulk traffic are distributed through different TCP connections to improve all traffics performance, unlike the default round-robin vanilla scheduler that continues to passes all the circuits through single TCP connection despite any congestion occurs in the network. This leads to poor performance for vanilla Tor scheduler in all cases from Figure 4 (a) to (d).

Figure 4 (e) and (f) shows a CDF of mean cells processed and mean cells in circuit queues respectively. These results are important to analyze how each scheduler attempting to maximize network utilization under different network load, in an attempt to improve traffic performance for all the web and bulk clients. In Figure 4 (e), the results show significant increases in the traffics processed by the modified switching algorithm in 100 minutes run time experiments, at which the point of measurements begins from when the web and bulk clients start downloading files up to times when the downloads are completed. The results show 50% of mean cells processed for modified switching algorithm are less than 119,000 cells for web traffics and, less than 120,000 cells for bulk traffics. Both results of proposed modified switching algorithm are higher than 50% cells processed for priority EWMA 66 and vanilla Tor schedulers. In Figure 4 (f), when compared the results of modified switching algorithm to priority EWMA 66 and vanilla Tor schedulers, the results for mean cells in circuit queues experienced small variances for all three schedulers. The results show that the circuit queues increase in the entry, middle and exit relays for all the web and bulk traffics. Under the heavy load, the EWMA 66 scheduler appears to perform the best for the web traffics with 50% of mean cells in the circuit queues, which are less

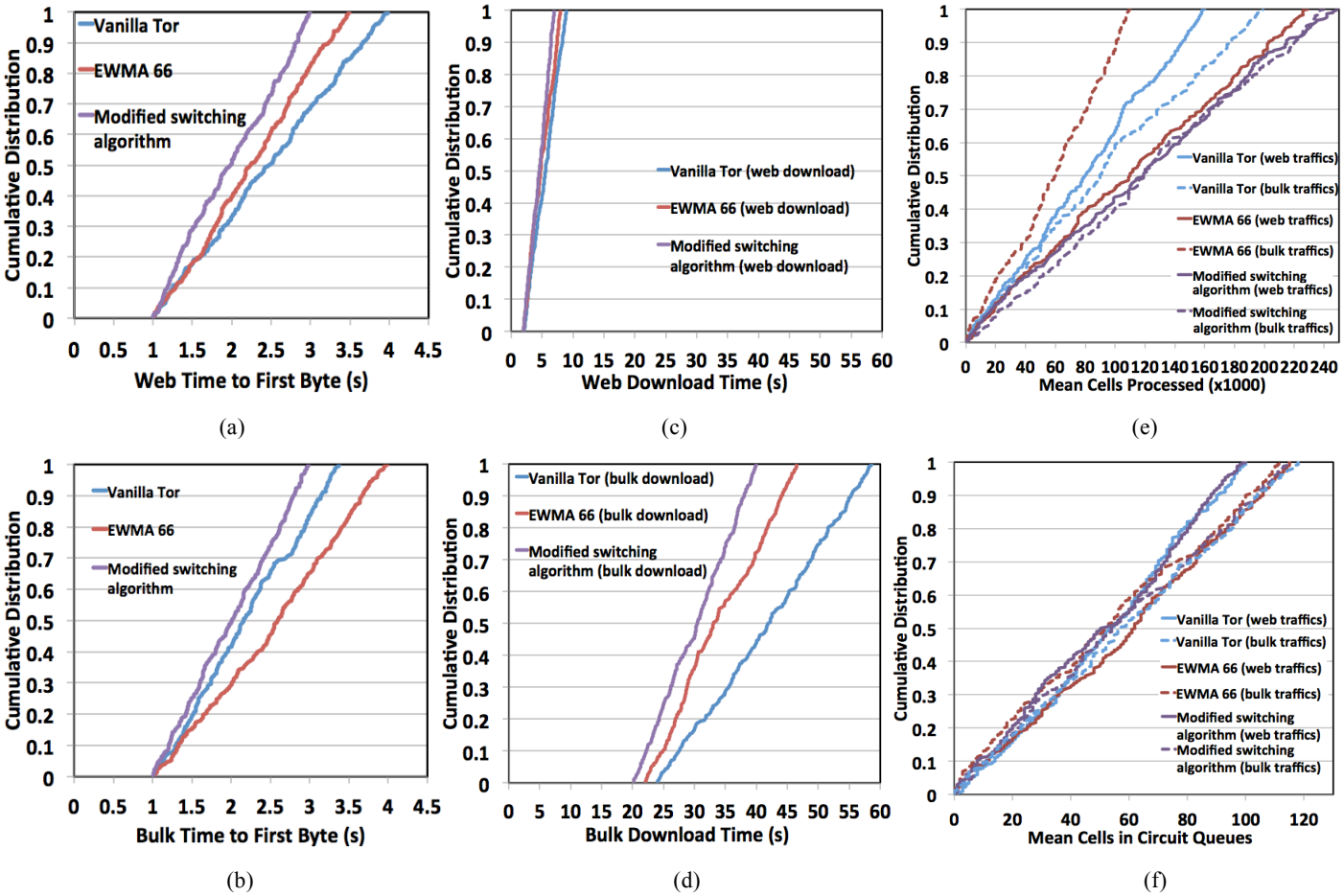

Figure 4. The comparison results of three schedulers running in the setup shadow emulator. (a) The web time to first byte (when the web clients click the web browser and time it took the first byte to arrive at the client side). (b) The bulk time to first byte (when the bulk clients click the web browser and time it took the first byte to arrived at the client side). (c) The web clients download time of 320 KB file and, (d) the bulk clients download time of 5 MB file. (e) CDF of mean cells processed for web clients and bulk clients during downloads and, (f) CDF of mean cells in the circuit queues.

than 62 cells and less than 52 cells are for the bulk traffics. For vanilla Tor scheduler, the results show that 50% of mean cells in the circuit queues are less than 54 cells for the web traffics and, less than 57 cells are for the bulk traffics. And for the modified switching algorithm, 50% of mean cells in the circuit queues are less than 50 cells for the web traffics and, less than 54 cells are for the bulk traffics.

The experiment measurements shows that, when the network loads are light or too heavy, all the results are nearly equal regardless of the selected circuit scheduler. Therefore, in overall observation of the simulation results, we concluded that the modified switching scheduling of bulk traffic with higher EWMA value to different TCP connection appears to be effective regardless of the wide network deployment in shadow or outside of shadow emulator.

The configured algorithm thresholds α and β effectively controls the buffer occupancy in Tor application level and maximizes the web and bulk time to first byte received, and downloading times among all different loads in the network.

6. Discussion

We evaluated the improvements of active circuit switching algorithm with the default vanilla Tor and priority EWMA scheduler. We examined the beneficial of modifying the current EWMA scheduler in Tor algorithm by combining the circuit protocol stack (Tor application level) and, TCP protocol in order to improve the overall Tor performance. This performance improvement is essential to achieve better security on the Tor network. This is because by allowing the users to select their preferred guard entry ORs that can detect congestion quickly in the Tor network and dynamically distributing the traffics increases the performance and usability of Tor, and therefore the potential user base and security of the Tor network is greatly enhance. Note that more users joining the Tor network increases the mixing of traffics on the network. Therefore, more mixture of traffics can obscure any adversaries who perform the traffic analysis in trying to identify the sources and destinations of the captured packets.

Furthermore, the performance improvements of modified switching algorithm increases the mean cells processed of transfer times for all the web and bulk traffics on the network. The network utilization is maximized for different network loads,

because any packet drops on one TCP connection does not affect the flow of traffic in another TCP connection. Hence, scheduling the traffics via different TCP connection proved to be effective and dramatically reduces the random variances of queuing cells and traffic flows of all the web and bulk clients as observed in our experiments.

7. Conclusion

In this paper, we presented the modified switching scheduling algorithm to improve the flow of traffics in Tor and compared the results with other schedulers, such as priority EWMA and non-priority round-robin vanilla Tor schedulers in a large-scale discrete event simulator, called the shadow. The plug-in we used to run Tor in the shadow emulator is called scallion. We introduced the threshold technique that controls the output buffer occupancy in the guard entry OR. The results show that our modeling of active circuit switching algorithm proved to be suitable for obtaining favorable results for the entire Tor clients in our experiments.

We observed that for the network responsiveness of each web and bulk client, the EWMA scheduler appears to perform best for the web traffics but not for the heavy bulk traffics, i.e., the EWMA scheduler performance is unstable under all network condition. For vanilla Tor scheduler, the performance is possibly improved, however, it is still not suitable when there are many bulks and webs traffics multiplexes through a single TCP connection. Hence, the performance for vanilla Tor scheduler is degrading for all our experiments.

To conclude, all the scheduler's results illustrate that performance benefits heavily depend on the network traffic patterns and network congestion state. The overwhelming load of Tor clients on the network can increase congestion and creates additional bottlenecks, which can possibly affect any advance scheduler performance in Tor.

Therefore, more experimentation is needed to perform in the live Tor network to confirm the effectiveness of our proposed modified circuit switching algorithm performance.

7.1 Future work

Having shown the performance benefits of active circuit switching algorithm, a deeper research needs to carry out to understand our method adaptation and its effects on client performance in the live Tor network.

In addition, we need to analyze the security of circuit switching schedule against adversarial attacks on anonymity. We need to study the direct impact of actively switching of bulk traffic to the different path and, what any adversary can learn from the guard entry ORs that performs the switching of active bulk traffics.

Finally, we need to research how the information leaked at the entry ORs can affect the anonymity services of the Tor network.

Reference

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," In Proceedings of the 13th USENIX Security Symposium, USENIX Association, 2004.

[2] Tor metrics portal, available from <https://metrics.torproject.org> (accessed 2016-03).

[3] Who uses Tor? available from <https://www.torproject.org/about/torusers.html.en> (accessed 2016-03).

[4] J. Reardon and I. Goldberg. Improving Tor using a TCP-over-DTLS Tunnel. In Proceedings of the 18th USENIX Security Symposium, pp. 119–133, 2009.

[5] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. In Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 329–339, 2010.

[6] R. Jansen and N. Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In Proceedings of the 19th Network and Distributed System Security Symposium, 2012.

[7] T. W. J. Ngan, R. Dingledine and D. S. Wallach. Building Incentives into Tor. In the Proceedings of Financial Cryptography, 2010.

[8] W. B. Moore, C. Wacek and M. Sherr. Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race With Tortoise. In Proceedings of 2011 Annual Computer Security Applications Conference, 2011.

[9] R. Snader and N. Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In Proceedings of the 16th Network and Distributed Security Symposium, 2008.

[10] T. Wang, K. Bauer, C. Forero and I. Goldberg. Congestion-aware Path Selection for Tor. In Proceedings of Financial Cryptography, 2012.

[11] R. Jansen, N. Hopper and Y. Kim. Recruiting New Tor Relays with BRAIDS. In Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 319–328, 2010.

[12] F. Tschorsch, and B. Scheuermann. Tor is Unfair–and What to Do About It, 2011.

[13] E. Hahne. Round-robin Scheduling for Max-min Fairness in Data Networks. IEEE Journal on Selected Areas in Communications 9, pp. 1024–1039, 1991.

[14] T. G. Kale, S. Ohzahata, C. Wu and T. Kato, Reducing Congestion in the Tor Network with Circuit Switching, Journal of Information Processing, vol.23, no.5, pp. 589-602, September 2015.

[15] R. Jansen, P. Syverson, and N. Hopper, Throttling Tor Bandwidth Parasites. University of Minnesota TR 11-019, 2011.

[16] J. Reardon. Improving Tor using a TCP-over-DTLS Tunnel. Master's thesis, University of Waterloo,Waterloo, ON, September 2008.

[17] T. G. Kale, S. Ohzahata, C. Wu and T. Kato. Analyzing the Drawbacks of Node-Based Delays in Tor, In Proceedings of IEEE CQR 2014, USA, pp. 1-6, May 2014.

[18] Shadow configuration, available from <https://github.com/shadow/shadow/wiki/3-Simulation-Customization> (accessed 2016-01).

[19] Shadow plugin, available from <https://github.com/shadow/shadow-plugin-tor/wiki> (accessed 2016-01).