Evaluation of Dynamic Circuit Switching to Reduce Congestion in Tor

Timothy Girry Kale Satoshi Ohzahata Celimuge Wu and 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 a widespread overlay anonymous network that is used by millions of users today. Tor focuses on improving the privacy of its users by routing their traffics through a series of onion routers that are located around the world. Since Tor operations mainly depend on volunteers who donated onion routers in the network, there are variances of capacities amongst different onion routers, which degrade the performances of Tor. In an effort to reduce the congestion problems in Tor, first we implemented control metrics to detect congestion on the entry OR. Second, we performed dynamic estimation of circuit congestion to select the best path in Tor. Third, we addressed the unfair distribution of bandwidth between the bulk and light traffics in Tor. We distributed the light and bulk traffics in Tor by having almost the same throughputs compared to default Tor multiplexing circuit approach. We use RTTs, circuit congestion and OR capacities as the main metrics to select the path. From our experimental analysis, we showed that our circuit switching method is effective to reduce the congestion problems in Tor.**

Keywords—Onion Routing (OR); Onion Proxies (OP); Tor Anonymity; Circuit Congestion; Latency; Throughput; Capacity

I. INTRODUCTION

Tor has just started and releases in 2002 [1] and fully published its core design document in 2004. Tor provides the privacy service for its users by sending their traffics through a series of onion routers (ORs). Tor enhances the anonymity of its users by encrypting all the transfer cells with multiple layers of encryptions technique. However, there are problems in Tor, which relating to unfair distribution of traffics that degrades the performance of Tor users.

Roger and Steven [2] discussed the problems in Tor and effort to solving it. The main problem is congestion control in Tor does not work well. Tor fails to feed back information to the sender when cells are dropped. To address these problems, Tang and Goldberg [3] proposed a method to improve the traffics between the bulk and light circuits. They implemented their method by calculating the exponentially weighted moving average (EWMA) of cells and measured the recent activity of a circuit. They gave higher priority to circuits that have lower EWMA value to have their cells sent. The problem with their approach is bulk and light circuits are still using the same TCP connection, which can lead to increasing queue length in Tor and TCP buffers when cells are dropped. In addition, there is competition between the bulk and light circuits of having their data transferred [2, 3]. The other common problem is Tor does not have enough capacities to handle all its users [2].

Our motivation comes from the significant high circuit congestion and variable delays in Tor, which degrades the quality of end-to-end communication. In our previous work [4], we introduce the circuit switching method to reduce the congestion in Tor by quickly detecting the congestion in the entry OR. However, there are limitation to our previous work, which we do not address the mechanism of dynamic selection and measuring of lower congested path in Tor. The circuit congestion mechanism of combining the routing state of all OR nodes and links between the client onion proxies (OPs) to the web server, are also not addressed.

To address the previous problems in Tor and the limitation to our previous work, in this paper, we presented the dynamic circuit selection method to switch the active bulk traffics to less congested circuit. We dynamically select the alternative circuit by combining the congestion state of all the selected ORs and links between the sources and destination.

We successfully performed our experimental analysis and offered the following contributions.

- The implementation of dynamic selection and switching of active bulk traffic improves the throughputs and RTTs of cells for all light and bulk users. We observed that by sending the probe cells to measuring the RTTs and isolating the circuit congestion, to dynamically select the best path in Tor are promising.
- Allowing a single congested OR to create another TCP connection and connecting the bulk circuit to higher OR capacity, increases the overall network capacities and reduces the circuit congestion.
- Our modified control algorithm is implemented in the entry OR to monitor the buffer occupancy and, our control scheme can quickly detect the congestion along the circuit.

In the next Section, we discuss the Tor overview and Section III explains our proposed method. Section IV explains our experimental setup and Section V shows our measurement approaches. Section VI discusses our experimental results and Section VII holds our conclusion.

II. TOR OVERVIEW

Tor was first built in the US Naval laboratories and it is classified as the second generation of onion routing [5]. Tor's main goal is to prevent hackers from discovering the physical location of its users by hiding their IP addresses and contents access from the web servers [6].

Fig. 1 shows the overview of the Tor network. Users around the world runs their ORs based on voluntarily basis, and Tor directory authority server is responsible for keeping the list of all the ORs. Tor uses mixed network techniques to develop its security and privacy communication [6]. A mix network consists of multiple ORs that provide anonymity to Tor users. The ORs accept fixed length 512 bytes cells from the Tor client and performs the cryptographic transformation on the cells before forwarding them to the next OR.

Tor client runs an onion proxy (OP) to connect and send the cells through the Tor network. The client (OP) picks a first entry OR from the Tor directory lists, and makes a TCP connection as well as the transport layer security (TLS) on that connection. Tor was designed to multiplex circuits to single TCP connection to enhance its security services. At the circuit level is where Tor de-multiplex and multiplexing many circuits to single outgoing TCP connection.

After Tor client connect to entry OR, the Tor client then instructs the entry OR to connect to the middle OR, and the TCP connection is established between them.

978-1-4673-9348-5/15/$31.00 ©2015 IEEE

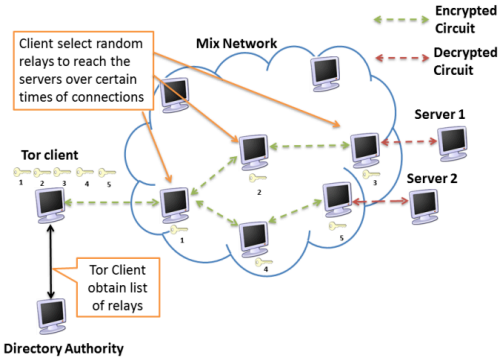


Fig. 1 Overview of the Tor network.

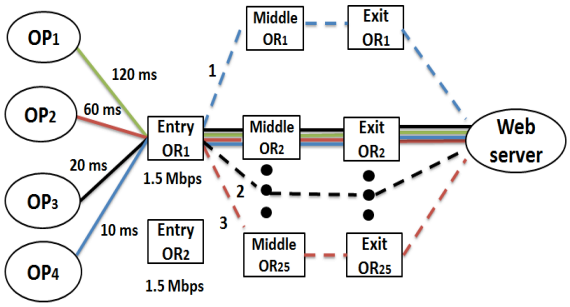


Fig. 2 Overview of dynamic selection and switching of bulk circuit traffics in the live Tor network.

The client OP then instructs the middle OR to contact the exit OR, and the TCP connection is also made between the middle and the exit OR. The exit OR can de-multiplex many circuits from the middle OR and opens the TCP connection to the web server. When the connection is successful, the exit OR reports a one-byte message back to the Tor client [6]. Note that the circuit identification (CircID) for each connection is unique between OP to OR and, OR to OR.

III. Proposed Method

In this paper, we proposed the dynamic selection of best circuit in Tor and, switching of active bulk circuit that contributes to bottleneck to different circuit. This approach addresses the increasing congestion on OR and, gives more space to light circuit traffics to have better throughputs.

A. Overview of the Proposed Method

In this Section, we explain the overview of our proposed method to solve the congestion problems in Tor.

Fig. 2 shows the overview approach of dynamic selection and switching of bulk circuit traffics in the live Tor network. Each Tor client dynamically selects the middle and exit ORs, which are obtained from the Tor directory server during bootstrap via authority "DirPort". The middle ORs relay traffics within Tor network and the exit ORs have the exit policies to relay traffics outside of Tor. We performed modification in the entry OR_1 and entry OR_2 and, connected them to the middle and exit ORs in the live Tor network. We proposed and designed the methodology for maintaining the TCP connection between the ORs, to stay within the adequate utilization of bandwidth usage by all incoming circuits.

Fig. 3 shows the design of circuit switching method in the entry OR, we used a separate kernel-mode TCP connection for each outgoing circuit from Tor, after circuit is switched to different TCP connection. We considered this approach of TCP specification for efficient event delivery packets between OP-to-entry OR, and OR-to-OR. In addition, the circuit switching method uses the default TCP over TLS protocols that are already deployed in Tor. Therefore, implementing our method in the Tor network is simple and does not require altering the whole Tor architecture. Each client OP in our design continues to maintain a single TCP connection with each of their selected entry OR. The circuit switching proposed modification is local only to the entry OR. This means that not all middle and exit ORs along the circuit needs to upgrade to benefit from dynamic circuit switching.

B. Dynamic Selection of Circuit in the live Tor Network

We implemented a method to allow the client OPs to quickly detect the congestion and dynamically switch the active traffics to alternative circuit, which is less congested. All the Tor clients can preemptively built circuits and measure the congestion state of all circuits. The client OP can construct many circuits at the beginning, and while each client uses one circuit to transfer their data, they can also continue to measure the other circuits to obtain the end-to-end RTTs (from OP to exit OR). The end-to-end RTTs give the estimation of circuit congestion T_c. The measurement is done on the stream level by sending the probe BEGIN cell and receiving the CONNECTED cell without causing any overhead on the circuit. The round-trip time (RTT) measurements were saved in the sets of list k $\{RTT_1, RTT_2, RTT_3 \ldots RTT_k\}$, from which the Tor client can isolate the circuit congestion T_c. There are total of 5 probe cells sends on each circuit after every 10 seconds interval. The Tor clients select the lowest RTT and consider that as RTT_{min}, and highest RTT as RTT_{max}. Circuit that is less congested is always selected to relay the active bulk traffics.

We applied the approach of Wang et al. [7] to estimate the total circuit congestion from obtained sets of RTTs. When the selected nodes are congested, the amount of congestion time is added to the RTT of probe cell before it can reaches back to the client. Equation (1) shows the calculation for the total circuit congestion (T_c) to determine the less congested path and, dynamically switch the bulk traffic. μ is the small constant added to the measurements as the reactions to transient congestion [7].

$$T_c = RTT_{\max} - RTT_{\min} + \mu \qquad (1)$$

The circuit congestion T_c is varying for each circuit depending on the extent of distribution nodes in the Tor network. T_c also takes into account the congestion in the Tor OR, since the probe cell is passing through all the ORs as it traversed through Tor.

C. Circuit Traffic Classification in the Entry OR

To easily classify the circuit traffic in the entry OR, we applied the techniques, which we used in our previous work [4], to identify the bulk and light traffics in Tor. We classified the circuit traffics as bulk or light by using the exponentially weighted moving average algorithm (EWMA) [3] shown in Eq. (2). The EWMA calculates the moving average number of cells sent on each circuit and, adds greater weight to recent values over time. We estimated the EWMA periodic samples of instantaneous buffer occupancy B_t in the Tor output when the algorithm runs on the entry ORs.

$$B_t = \gamma * B_{t-1} + (1-\gamma) * B_{sample} \qquad (0 < \gamma < 1) \qquad (2)$$

B_{sample} is the current 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

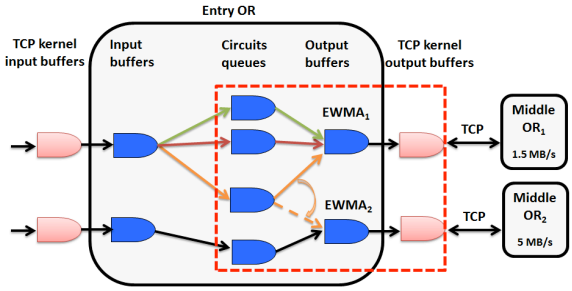


Fig. 3 Design of circuit switching method in the entry OR.

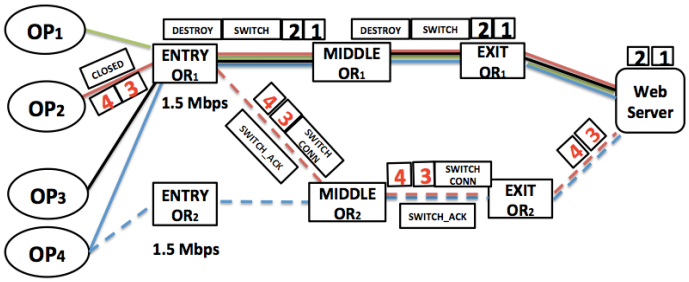


Fig. 4 Design procedure of dynamic circuit switching. The solid-line indicates the initial circuit that all OP built. The dash-line indicates the alternative switched circuit that transfers the active bulk traffics.

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 from all incoming circuits.

Fig. 3 shows the traffics from circuit queues multiplexed to output buffers in Tor. Each output buffer are monitored and calculated by the EWMA algorithm. The box dash-line indicates the area where dynamic switching of active bulk traffics occurs. The dashed-line arrow, which switched to another output buffer, indicates the switching of bulk traffic when the previous output buffer is congested. The switched bulk traffic then transferred to the middle OR_2, which has the 5 MB/s bandwidth. This approach connected the bulk traffic to higher middle OR capacity to lessen the congestion in the entry OR. Hence, the OR capacities and throughputs for all circuit traffics are improved.

The maximum threshold values configured on each output buffer in Tor is β = 0.5 (16 KB), and minimum is α = 0.1 (3.2 KB). These thresholds are fractional of the total allocated kernel memory on each output buffer, which is 32 KB. If the current buffer occupancy B_t is within the acceptable range of minimum and maximum ($\alpha * B_{t-1} \leq B_t \leq \beta * B_{t-1}$) thresholds, then no switching occurs. In this case, all the circuit traffics are not congested. However, if the current Tor output buffer B_t rises above the maximum threshold ($B_t > \beta * B_{t-1}$), the congestion occurs and the bulk circuit is dynamically switch to different alternative circuit (see Fig. 3). This approach allows the circuit with lower EWMA value (light interactive circuits) to have better throughputs.

D. Design Procedure of Dynamic Circuit Switching

This Section explains in details the procedures for dynamic switching of active bulk circuit begin from the point in time when the congestion occurs.

Fig. 4 shows the design for dynamic switching approach we applied to client OP_2 (as the bulk circuit traffic) after congestion is detected at the entry OR_1. Firstly, we showed the designed procedure of dynamic circuit switching when uploading a file to the web server. Secondly, we explained the procedure of downloading a file from the web server.

For uploading and downloading procedures, the design communication between all the OPs to exit ORs, and communication from the exit ORs to the web server is using the TCP protocol over TLS protocol. The web server accepts the TLS session resumptions [8] to resume connections after switching the circuit traffics, and continues receiving the data from different exit OR_2. We applied two control cells that Gopal and Heninger [9] defined to manage the connection transfer, which are SWITCH and SWITCHED_CONN cells. The SWITCH cell has the payload that contains a flag to inform the middle and exit ORs on the previous congested circuit that, all further cells will be switch to the new circuit. We designed the SWITCH_ACK cell, which the middle and exit ORs on the new alternative circuit can sends back to inform the entry OR that the connection is ready to be used. Note that the SWITCHED_CONN and SWITCH_ACK cells have no payload.

1) Designed procedures when uploading a file to the web server: We performed two circuit switching approaches for active bulk circuit. 1. Switching on a single entry OR and 2, switching to different entry OR[1].

a) Dynamic circuit switching of traffics on a single entry OR: First, we allow the entire client OPs (OP_1, OP_2, OP_3 and OP_4) circuits to multiplexed on a single entry OR_1 as shown in Fig. 4. Before entry OR_1 switches the circuit to the alternative circuit, the entry OR_1 monitors the incoming circuit traffics and checks if there is no congestion. Our algorithm runs on the entry OR_1 and keeps track of the buffer usage. When the entry OR_1 detects the bulk circuit is overflowing the output buffer beyond the configured maximum threshold β, the congestion occurs along the circuit. The entry OR_1 then sends the SWITCH cell to the middle OR_1 and exit OR_1 (initial circuit created) with the DESTROY cell [10], after cells with sequence number 1 and 2 are transferred. Note that when the entry OR_1 switched the circuit to alternative circuit, the old circuit is discarded after the entry OR_1 sends the DESTROY cell. In addition, when the exit OR_1 receive the SWITCH and DESTROY control cells, the exit OR_1 must first sends the cells with sequence number 1 and 2 to the web server, before it tears down any associated edge for the corresponding circuit identification (CircID). We take into consideration that when switching the active bulk circuit traffic to new alternative circuit, the transferring cells with sequence number 3 and 4 cannot be easily decrypted along the new switched circuit. This is because the encryption keys are different for each OR in the new circuit compared to the previous old circuit. To solve this problem, when the entry OR_1 tears down the circuit to middle OR_1 and exit OR_1, the entry OR_1 sends a single byte CLOSED cell[2] to inform the client OP_2 that the appropriate CircID, which connected to middle OR_1 is closed. This will prompt the client OP_2 to use another alternative circuit and resends the cells[3] with sequence number 3 and 4 (with right CircID header) to the web server. The entry OR_1 then adapted the switching of bulk traffic to the alternative circuit that was preemptively built by the client OP. The client OP_2 can resume the session, which includes sending the "Stream

[1] The switching of circuit to different entry OR occurs if the middle and exit ORs on the alternative circuit are highly congested.

[2] CLOSE cell indicates a failure on the circuit. This cell prompted the client to use another new alternative circuit.

[3] The client OP_2 encrypted the resending cells via the alternative circuit with the shared keys of entry OR_1, middle OR_2 and exit OR_2.

Identifier" and "relay begin cell". When the exit OR_2 in the new circuit receives the information of relay begin cell, it sends the cells with sequence number 3 and 4 to the same web server IP address and port number. The same information encoding in the relay begin cell ensures that the cells sent by the exit OR_1 and exit OR_2 are successfully delivered, and processed by the web server.

b) Dynamic circuit switching of traffics to different entry OR: When the entry OR_1 detects the OP_4 circuit flooded the output buffer, the entry OR_1 sends a SWITCH_CONN control cell to the middle OR_2 and exit OR_2. If the entry OR_1 does not receive any SWITCH_ACK in reply from the middle OR_2 and exit OR_2 or, remain unattached for SocksTimeout, (default 2 minute) at the entry OR_1 [11], the entry OR_1 discards the control cell and sends an "error message" back to the client OP_4 as appropriate to close the SOCKS connection. The client OP_4 will immediately switch its bulk traffic to another new circuit via entry OR_2 (see Fig. 4). Note that the new circuit was dynamically selected and connected to the web server. When the circuit connected to entry OR_2 is ready to be used, Tor (client OP_4) attaches the request's stream to the circuit and sends a BEGIN, BEGIN_DIR and RESOLVE cells as appropriate to the entry OR_2, middle OR_2 and exit OR_2. Note that when switching the circuit traffic via different entry OR_2, the cryptographic arrangement should not affects the flow of traffic, because the client OP_4 already has all the exchange keys to encrypt and decrypte the cells.

2) Designed procedures when downloading a file from the web server: In this Section, we explain the inward direction of circuit switching when downloading a file from the web server. The scenario is the same as the previous Section, where all the clients are multiplexed to single OR and the bulk circuit is switched after congestion is detected. Since the control metrics to detect the circuit congestion are running on the entry OR, when the bottleneck occurs due to cells queue in the Tor output buffer or sending TCP buffer towards the client OPs. The entry OR is responsible to send an "error message" after the remaining cells in the Tor output buffer are sent to appropriate bulk client. The entry OR sends an "error message" with corresponding CircID to notify the bulk client OP to close the SOCKS connection. This prompted the client OP that owns the bulk traffic to use another circuit, and continue on with the download from web server. The process of Tor clients to attach the stream data to new circuit and continue on with the download involves sending a BEGIN, BEGIN_DIR and RESOLVE pending cells as appropriate to download on the alternative circuit. The new exit OR contacts the web server after receiving the "relay resolve cell" to continue the download, based on the request from bulk client OP.

IV. Experimental Setup

Fig. 2 shows our experimental setup. We analyzed the circuit congestion and RTTs of four Tor clients when dynamically switching their bulk circuits to 25 middle ORs, and 25 exit ORs in the live Tor network. The setup entry OR_1 and entry OR_2 switches the bulk circuit one-by-one from the current obtained list of running ORs in the live Tor network. We use the Netem emulator tool to enforce the delay effects on the TCP traffics to client OPs. The delays between OP_1 to OR_1, OP_2 to OR_1, OP_3 to OR_1, and OP_3 to OR_1 are 120 ms, 60 ms, 20 ms and 10 ms, respectively. These settings enable us to configure the client OP circuits as bulk and light traffics. The entire four client OPs and two entry ORs were running on the commercial 1 Gbps network in our University. The OPs and ORs were running on Ubuntu OS 14.04.2 LTS, CPU 2.60 GHz 32 bit with 4 GB of RAM. We run our setup ORs in the Tor network for more than a month to gain stability in performance. We worked with Tor stable source code 0.2.4.19 for all the OPs and entry ORs at the time of experiment. We configured the bandwidth as 1.5 Mbps on the entry ORs so that it can hold enough pool of transferred bytes from the Tor clients.

The reason we perform this setup is for us to easily control and observe the effectiveness of our circuit-switching algorithm that runs on the entry OR. The entry OR_1 and entry OR_2 monitors the four clients traffics that multiplexes to single TCP connection, *before switching* and *after switching* the bulk circuit to different TCP connection. Since we modified and upgraded our entry ORs, the upgraded EWMA algorithm [4] can detect the bulk circuit that overflows the output buffers in Tor. In addition, setting up our entry ORs allows us to easily measure any improvements of throughputs after bulk circuit is switched.

Note that, other experiment setup topology would affect the outcome results of our circuit switching method if the entry OR is not upgraded. This is because, our method of sending the control messages to switch the active bulk circuit to new alternative circuit and, our modified EWMA algorithm to detect congestion [4] in Tor are newly invented approaches. To further examine how other experimental setup could impact the final results of circuit switching method is our future work.

V. Measurement Approaches

We measured the capacity of ORs when transferring a 12.3 MB file by uploading and downloading from the web server (https://www.dropbox.com/home). We observed the changes of OR capacities by taking the difference of capacities "*after switching*" and, "*before switching*" the bulk circuit, C_{after} - C_{before}. Note that we referred to OR capacity as the number of bytes an OR can transfer per second on all circuits going through it. In addition, before the circuit is switched means, all the light and bulk circuits are multiplexed to single TCP connection, and after circuit is switched means the bulk circuit traffic is successfully switched to different TCP connection.

To gain insights on how the circuit switching affects the OR capacity and throughputs between the bulk and light traffics, we present a standard analytical model that uses to express the OR capacity changes in Eq. (3) [12]. The $throughput_i$ is the average throughput measured on selected ORs. *Normalized* C_t is the estimated normalized capacity by the directory authorities for each OR at time *t*.

$$C_i^t = C_i^{t-1} * \frac{throughput_i}{Normalized\ C_t} \tag{3}$$

We measured the changes of capacity C_i^t (increase or decrease) at time *t* for selected OR, when all Tor clients are transferring traffics. We executed the Tor bandwidth script "nodemonitor.py" to measure the throughputs that OR can send per second for all circuits that multiplexed to single TCP connection (before switching) and, after bulk circuit switched to different TCP connection. When the OR capacity increases, the throughputs for both light and bulk traffics are also improved. We also measured the changes of circuit congestion and RTTs when sending the probe cells "after and before" the circuit traffics are switched to confirm the improvements.

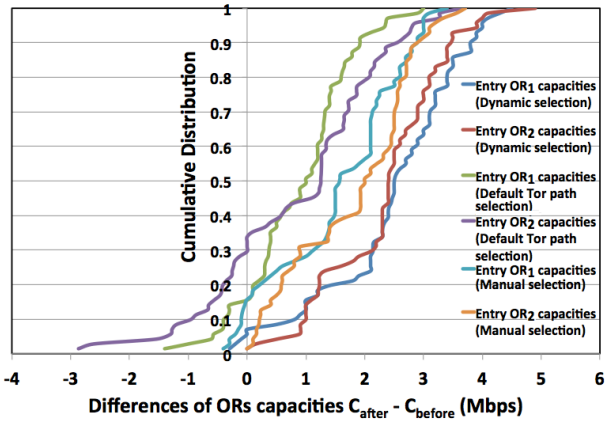


Fig. 5 The differences of OR capacities, C_{after} - C_{before}.

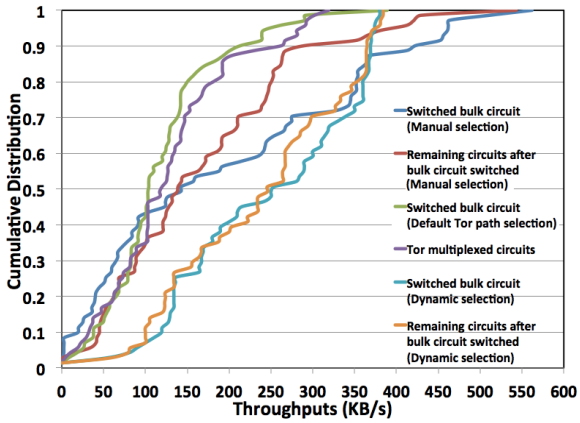


Fig. 6 Comparisons of throughputs when transferring a 12.3 MB file to the web server.

We run two algorithms, *path_finder.py*[4] and *rttprober.py*[5] [13] to obtain the best response time of RTTs during circuit construction and, when client measures the circuit congestion.

VI. Experimental Results

To confirm the effectiveness of our dynamic circuit switching approach, we compare our method with the default Tor approach of multiplexing circuits to single TCP connection. Since we publicly advertised our setup entry ORs in the Internet, other circuit traffics can pass through our entry ORs. We only monitor our own circuit traffics to observed changes in the OR capacities, throughputs, circuit congestion and RTTs.

Fig. 5 shows the cumulative distribution of capacities on the entry OR_1 and entry OR_2 nodes. We compared the results for dynamic selection of circuit to switch the bulk traffics, with the results of default Tor path selection and our previous works [4]. Note that the default Tor path selection is based on the random bandwidth weighted algorithm and our previous work [4] selection method, is based on the manual selection of pre-configured ORs (i.e., we select ORs based on the geographical location and consensus advertised bandwidths).

In Fig. 5, when dynamically selecting and switching the bulk circuit, which shows the results of 50% for entry OR_1 and entry OR_2 capacities are less than 2.5 Mbps and 2.4 Mbps, respectively. Whereas 50% of entries OR_1 and OR_2 capacities when client OPs performed the default Tor path selection, are less than 1 Mbps and 1.2 Mbps, respectively. The results for manual selection of circuit shows 50% of entry OR_1 and entry OR_2 capacities are less than 1.56 Mbps and 2 Mbps. The measurement results shows a positive increase in OR capacities for dynamic selection of less congestion circuit. However, there are also variances of capacities measured at the entry OR_1 and entry OR_2 for all comparisons, since less than 8% of measurements shows no increase in capacities (below zero Mbps) for dynamic circuit selection at entry OR_1. Less than 35% and 15% shows no improvement at entry OR_1 and entry OR_2 when applying the default Tor path selection. For manual selection, 15% shows no increase in capacities after switching the bulk traffic on entry OR_1. The reasons are some of the selected ORs in the live Tor network are not actively running and hibernating.

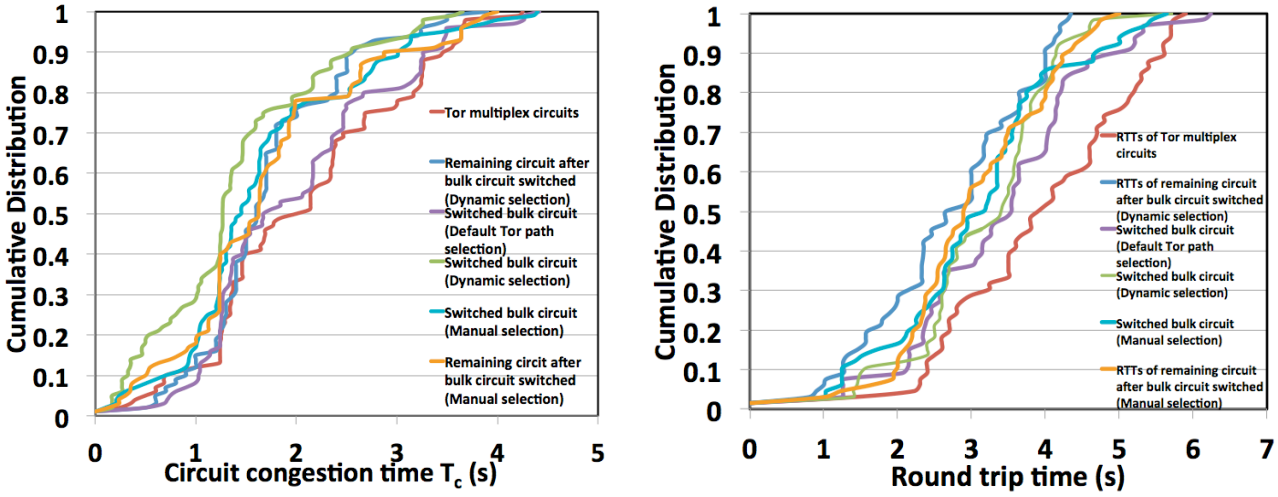
Moreover, the average entry OR capacities when dynamically selecting and switching the bulk traffic, improves the throughputs for all the light and bulk traffics. The average entry OR_1 capacity before bulk traffic switch is 1.6 ± 2.2 Mbps, but after bulk traffic is dynamically switched to alternative circuit, the entry OR_1 capacity rapidly increased to 4.1 ± 0.9 Mbps. On entry OR_2, the average capacities before bulk traffic switch is 2.1 ± 1.2 Mbps, however, after bulk traffic is switched to alternative circuit, the entry OR_2 capacity rapidly increased to 4.2 ± 1.6 Mbps. The measurement results confirmed our assumption that by quickly detecting the congestion at the entry OR and, dynamically switching the bulk traffic to less congested circuit, the OR capacities and throughputs are improved compared to our previous work [4].

Fig. 6 confirms the improvement of throughputs for both light and bulk traffics. We compared the results to default Tor circuit selection, manual circuit selection and Tor multiplexed circuits to single TCP connection. The dynamic circuit selection for less congested circuit path shows better throughput performances. 50% of throughputs are less than 245 KB/s for dynamic path selection and switching of bulk traffics, whereas 50% throughputs of the remaining light traffics after bulk traffic switched are less than 246 KB/s. The throughput results show fair distribution of traffics for both light and bulk users, since 50% throughputs are not variances. When switching the bulk circuit based on manual selection, 50% throughputs are less than 139 KB/s. And for the remaining light circuits after bulk traffic is switched, 50% throughputs are less than 138 KB/s. From the results, we observed that the throughputs are improved for manual selection by 13% (switched bulk circuit) and 6% (remaining light circuits). The worst cases are when multiplexing both light and bulk traffics to single TCP connection and when applying the Tor default path selection. 50% throughputs of the default Tor path selection are less than 103 KB/s, and throughputs for Tor multiplexing circuits to single TCP connection are less than 117 KB/s. Our method of dynamic selection of alternative circuits (based on isolating the congestion state of ORs) shows significant improvements, for all the light and bulk circuit throughputs. Next we measured the circuit congestion T_c to verify the improvement results.

Fig. 7 (a) and (b) shows the improvement of dynamically select the less congested circuit in Tor. We compared the results to default Tor multiplexed circuit approach, manual selection and the default Tor path selection.

[4] The path_finder.py algorithm looks for distribution of nodes from Tor directory servers and allocates the best path for Tor client to generate a circuit.

[5] The rttprober.py sends the probe cell every 10 seconds interval to measure the RTTs and Time-To-First Byte (TTFB), which we used to dynamically select the less congested circuit in Tor.



(a) (b)

Fig. 7 Comparison results of (a) total circuit congestion T_c from the clients to the exit ORs and, (b) round-trip time of probe cells sent between the clients and the exit ORs.

In Fig. 7 (a), 50% of circuit congestion time T_c when Tor multiplexed circuits to single TCP is less than 2 seconds, whereas 50% of T_c for the remaining light circuit after bulk circuit switched, is less than 1.6 seconds (an improvement of 0.4 seconds). The bulk traffics are switched to less congested circuits that were dynamically selected; therefore, the T_c for all bulk traffics are also improved by 50%, which are less than 1.3 seconds. When compared the results to default Tor path selection for switching the bulk traffic, 50% of T_c is less than 1.7 seconds. For manual selection of circuit to switch the bulk traffic, 50% of T_c is less than 1.45 seconds and the remaining light circuits are less than 1.61 seconds. Our approach of dynamically measuring the circuit congestion by RTTs is effective, since all the Tor clients know the congestion state of new alternative circuits in real-time basis. The reduction of circuit congestion time T_c would also result in faster RTTs, for all the light and bulk circuit traffics.

Fig. 7 (b) shows the distribution of RTTs for the same comparisons. Our dynamic selection of circuit approach shows better improvement of transfer probe cells. 50% of RTTs for the remaining light circuits after bulk traffic switched to different TCP connection are less than 2.6 seconds, whereas 50% of RTTs for bulk traffic when dynamically switched to alternative circuit are less than 3.2 seconds. 50% of RTTs for default Tor path selection are less than 3.4 seconds. Then for manual selection, 50% of RTTs when switching the bulk traffics are less than 3.16 seconds, and 50% of RTTs for the remaining light circuits are less than 2.89 seconds. The RTTs of Tor multiplexing circuits are the worst case, which has 50% of RTTs that are less than 3.8 seconds.

VII. Conclusion

We presented the improvement of dynamic selection and switching of active bulk traffics. The implementation of our control algorithm monitors the buffer occupancy in Tor and changes how Tor selects the alternative circuit. We observed that by sending the probe cells to measure the RTTs and, isolating the circuit congestion to dynamically select the best path in Tor are promising. Moreover, by allowing any single congested OR to switch the bulk traffics to higher capacity OR, we improved the network capacities and the throughputs. Finally, we concluded that by distributing the shared bandwidths on ORs after congestion occurs, achieved fair distribution of throughputs for all light and bulk clients.

Our future work is to evaluate the cases when there are many entry ORs executing our control algorithms to detect congestion in Tor. We need to evaluate the traffic distribution by dynamically switching the bulk circuit traffic, and observe the increase of OR capacities in Tor. In addition, we need to further examine how different experimental setup and topology could impact the final results of the circuit switching method, and do more experimental evaluation to verify our method is fit to implement in the current Tor architecture.

References

[1] R. Dingledine. http://archives.seul.org/or/dev/Sep-2002/msg00019.html. Accessed on June 2011.

[2] R. Dingledine and S. Murdoch, "Performance improvements on Tor or, why Tor is slow and what we're going to do about it," http://www.torproject.org/press/presskit/2009-03-11-performance.pdf, March 2009.

[3] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," Proc. of the 17th ACM conference on Computer and communications security 2010, pages 329-339, 2010.

[4] T. G. Kale, S. Ohzahata, C. Wu, T. Kato. Reducing Congestion in the Tor Network with Circuit Switching, IPSJ, Journal of Information Processing, September 2015.

[5] P. Syverson, M. Reed and D. Goldschlag. Onion routing access configurations. In Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX 2000), IEEE CS Press, Volume 1, pp. 34–40, 2000.

[6] R. Dingledine, N. Mathewson and P. Syverson. Tor: The Second-Generation Onion Router. Proceedings of the 13th USENIX Security Symposium, 2004.

[7] T. Wang, K. Bauer, C. Forero, and I. Goldberg. Congestion aware Path Selection for Tor. In Financial Cryptography and Data Security (FC), 2012.

[8] J. Salowey, H. Zhou, P. Eronen, H. Tschofenig. Stateless TLS Session Resumption. Standards Track, https://tools.ietf.org/html/rfc5077, January 2008.

[9] D. Gopal and N. Heninger. Torchestra: Reducing Interactive Traffic Delays over Tor. In Proc. of the 2012 ACM Workshop on Privacy in the Electronic Society, WPES 2012, ACM, pp. 31–42, 2012.

[10] R. Dingledine and N. Mathewson. Tor Protocol Specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=tor-spec.txt. Accessed on April 2014.

[11] R. Dingledine and N. Mathewson. Tor path specification. http://tor.hermetix.org/svn/trunk/doc/spec/path-spec.txt. Accessed on April 2013.

[12] K. Loesing, M. Perry and A. Gibson. Bandwidth Scanner Specification, 2011.https://gitweb.torproject.org/torflow.git/blob_plain/HEAD:/NetworkScanners/BwAuthority/README.spec.txt.

[13] Atlassian Bitbucket. https://bitbucket.org/ra_/tor-rtt/src. Accessed on April 2015.