HAL
archives-ouvertes.fr

An Investigation on Software-Defined Networks' Reactive Routing against BitTorrent

Damián Vicino, Chung-Horng Lung, Gabriel Wainer, Olivier Dalle

▶ **To cite this version:**

Damián Vicino, Chung-Horng Lung, Gabriel Wainer, Olivier Dalle. An Investigation on Software-Defined Networks' Reactive Routing against BitTorrent. IET Networks, IET, 2015, 4 (5), pp.249-254. <10.1049/iet-net.2014.0105>. <hal-01274250v2>

HAL Id: hal-01274250
https://hal.inria.fr/hal-01274250v2

Submitted on 15 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An Investigation on Software-Defined Networks' Reactive Routing against BitTorrent

Damián Vicino[a,b], Chung-Horng Lung[a], Gabriel Wainer[a], and Olivier Dalle[b]

[a]Dept. of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada
[b]Laboratoire d'Informatique, Signaux et Systmes de Sophia-Antipolis (I3S) - UMR7271 - UNS CNRS, 2000 route des Lucioles - Les Algorithmes - bt. Euclide B, 06900, Sophia Antipolis, PACA, France

November 12, 2014

Abstract

Technologies in Software-Defined Networks (SDNs) introduce programmatic ways to reorganize the network logical topology. A possible practical usage of SDNs is Reactive Routing, where the logical topology is continuously evolving based on traffic statistics and policies. Usually, the SDNs controllers are considered transparent to the higher layers. It is expected that changes in logical topology may not affect applications. Our goal is to study the impact of logical topology changes on BitTorrent, a popular peer-to-peer protocol in practice. In this paper, we focus on BitTorrent and the experimental results show that BitTorrent may produce the opposite effect to the one expected. We have run 32 BitTorrent clients in an emulated SDN ring topology and changed the virtual topology periodically by removing one link at the time from the ring. The experiments produced lower propagation when logical topology changed periodically than when it was static for BitTorrent traffic. For comparison, we recreated the same experiments using HTTP. For HTTP, we obtained slower propagation when logical topology changed than when it was static. We discuss the results and conclude that high layer protocols need to be carefully studied, and in some cases adapted, before being deployed in SDNs.
Keywords. Software-Defined Networks; BitTorrent; Reactive Routing

1 Introduction

Software-Defined Networks (alternatively called Software-Driven Networks) provide management flexibility and automation introducing programmability mechanisms to the network components. Using these mechanisms, network opera-

tors or software tools can quickly react to cover specific requirements applying changes to the logical topology when needed.

The mean exploited to provide this programmability mechanisms is the clear separation of the Data Plane, involved in the forwarding of packages, from the Control Plane, involved in routing decisions. Most of the time this separation splits the Network Layer [15] in two, but sometimes other layers are also involved at the time of decision making.

When the reactions are automatic, it is called Reactive Routing. On Reactive Routing the logical topology is continuously evolving based on traffic statistics such as workload or jitter which can be collected by the switches and external information provided by services or users.

Our motivation is to study the impact of SDNs' Reactive Routing ideas on different network traffic scenarios and develop algorithms to predict the impact of the decisions taken before applying those algorithms. One use of this study would be to know if a stabilization time needs to be considered before deciding a new change to be made or avoided. The effect of one change to the network may affect subsequent monitoring and decision making, and may loop for a long time though different logical topologies without getting benefit at all.

Our goal is investigate scenarios and conduct experiments to understand the interaction between higher level protocols and changes in the logical topologies adopting SDN controllers.

As a first step to separate scenarios we classify the traffic in the network by its participants. We describe three categories: Fully Internal, Fully External and Half Internal depending on the location of the participants of the communication being deployed. The participants can all be inside of a local SDN, outside of a SDN, or half inside and half outside.

The scope of this paper is limited to one particular protocol, the BitTorrent protocol [3], in a Fully Internal situation which has showed some interesting unsuspected behavior. In our experiments, we emulated a ring topology and we periodically changed the logical topology. The results obtained from experiments show higher propagation rate than when the same logical topology rested static for the whole experiment.

We recreated similar experiments for HTTP. For HTTP, we obtained the opposite results than those using BitTorrent.

In the following sections we explain how SDN and BitTorrent works. We describe our interaction classification. We explain our experiments and the workbench we used to run the experiments. We show the results obtained and discuss them. And we conclude that high level protocols need to be studied carefully before using them in a Reactive Routing network to avoid bouncing effects. This work is an extension of the one presented presented in SDN-NGAS-2014 [20].

2 Background

This section describes the basic background related to SDN and BitTorrent. In addition, the section presents some recent proposed approaches in the literature that are related to SDN and peer-to-peer communications, including BitTorrent.

Project	Southbound Protocols	Northbound APIs	Lang.
NOX/POX[4]	OpenFlow 1.0	undefined	C++ Python
RYU[19]	OpenFlow 1.0 OpenFlow 1.3	undefined	Python
Floodlight[17]	OpenFlow 1.0	Java RESTful	Java
OpenDaylight[5]	OpenFlow 1.0 OpenFlow 1.3	OSGI RESTful	Java

Table 1: Comparison of Open Source Controller projects.

2.1 Software-Defined Networks

The Open Networking Foundation defines a SDN as "the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices"[6]. This definition states an objective, but it does not tell about the means to achieve it. SDNs are based on a layered architecture with three levels. The lower layer, known as the Infrastructure Layer, is where the switches are. The middle layer, known as the Control Layer, is where the controllers and services are. Finally, the top layer, known as the Application Layer, is where developers can implement software based on the controllers' APIs.

Since the architecture is layered, it needs to define the communications between the connected layers to provide services. In the case of communications between the Infrastructure Layer and the Control Layer, there is as many protocols as switches' manufacturers.

To increase compatibility between devices from different manufacturers, ease the research and simplify the controllers implementation, the Open Networking Foundation's OpenFlow project defined a set of standards adopted by several switches' manufacturers. OpenFlow proposes a set of protocols to be used to communicate between OpenFlow-enabled switches and controllers and it also defines the controller southbound API. The protocols are classified as part of the WIRE protocol (also called OpenFlow) and classified as part of the configuration and management protocol (OF-CONFIG). WIRE is used to define match-action rules in the internal forwarding tables. OF-CONFIG protocol is used to configure multiple OpenFlow datapaths sharing the same physical or virtual platform remotely.

In SDNs, the controllers [13] are the most important component, controllers are defined as the software components taking care of: managing the network state, working the high level model of the network, exporting a high level API to the Application Layer, maintaining the connections and notifications between Data Plane agents and the controller itself, and discovering services and basic routing functionalities usually implemented as plugins. Usually each Controller developer implements a different API which makes impossible to port applications between different controllers easily. In Table 1, we compare some characteristics of the most popular open source controllers currently available.

In the interaction between the Control Layer and the Application Layer,

the situation is pretty much the same. Each controller developer implements a different API, usually accessed by RESTful HTTP calls. There is no attempt to standardize the API in general, but there is one specific use case where the API was standardized. The case is the interaction with cloud OpenStack technologies, in this case OpenStack's Neutron project has to interact with the Controllers API with is fixed by Neutron. Some controllers decided then to provide multiple APIs, including this one. Two examples are RYU and OpenDaylight which both export OpenStack API.

We will not discuss the Application Layer details, any application developed in any languages using the northbound of the controller is a valid application. Some examples of common applications implemented are firewalls, orchestrators, operator's consoles and visualization tools.

2.2 BitTorrent

Most peer-to-peer content distribution protocols (e.g.,as eDonkey-2000) use a schema where users subscribe to obtain a piece of the content. The peer providing the content keeps track of requests in priority queues, and after waiting for a while the request gets in the front of the queue and the piece is transferred to the user requesting the piece. A completely different approach is taken in BitTorrent [3]. Its goal is to maximize the global aggregation of throughput for an specific content, the result can be orders of magnitude faster than previous protocols [16]. In Figure 1 we show some experimental results of the propagation of a file to 32 nodes comparing the use of eDonkey-2000 protocols (based in queue suscription) against a BitTorrent alternative [21] in PlanetLab [2] (a global research network) which were obtained in previous work.

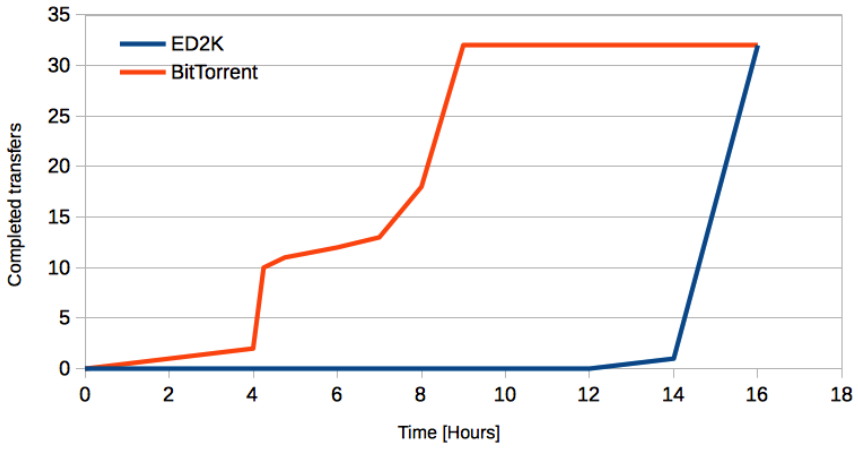

Figure 1: Comparison of propagation of a 300MB in PlanetLab using aMule (eDonkey-2000) and hMule (BitTorrent).

In BitTorrent, once the peer joins a group of peers sharing interest in the same content, it collaborates applying the following strategies:

- Piece selection (strategies deciding next piece to request)
  - Strict priority: Once a subpiece of a piece has been obtained, transferring the remaining sub-pieces of that piece has priority over anything else.
  - Rarest first: When starting the transfer of any piece, but the first piece downloaded, the least available piece is requested.
  - Random first piece: The first piece of the transfer is selected randomly.
  - End game mode: Once all the sub-pieces that a peer doesn't have are actively being requested, it sends requests for all sub-pieces to all peers.
- Choking (these are strategies that a peer temporarily refuses transfer to some peers, although transferring from those peers is not blocked).
  - Pareto efficiency: BitTorrent tries to maximize the reciprocate of upload connections.
  - BitTorrent choking algorithm: BitTorrent unchokes a fixed amount of peers to try to saturate upload capacity, decision is based on the download rate.
  - Optimistic unchoking: Every three cycles, a single peer is unchoked to check if there is a better option than keeps going with those peers already unchoked.
  - Anti-snubbing: When not data have been received from a peer for more than a minute, anti-snubbing assumes the peer doing the transferring decided to choke and as reciprocation it chokes.
  - Upload only: Once the download is complete and there are no download rates in which to base decisions, the upload rate is used to maximize availability.

On Piece Selection strategies, the input for decision making is restricted to the content and the known peers, there is no metrics about current traffic involved. We assume these strategies should not be influenced by changes in the underlying network.

On Choking Algorithms strategies, decisions are based on recent transfer connectivity metrics. A slight variation in the metrics can force to choke a peer and disconnect it for several minutes. For this reason, we believe that BitTorrent can be misled by the underlay topology changes using SDNs Reactive Routing approach and fails to select the best connections available.

Several implementations of open source and proprietary BitTorrent clients exists, in some cases as standalone applications and in others enbeded in an application for handling specific tasks as downloading updates or sharing users submitions. Some popular standalone clients are: BitTorrent, Vuze, rTorrent.

For BitTorrent experiments in this paper, we decided to use Libtorrent-Rasterbar [14]. LibTorrent-Rasterbar not only implements the BitTorrent protocol, it also implements μTP protocol and a set of related tools needed to build a complete client in a few lines of code. Some extra features provided are: session management, plugins management, DHT access, metadata exchange and

persistence of the content. The build from source code offers the option to generate Python and Ruby bindings and it also allows to externally define extensions to the protocol through its API.

An extra difficulty that BitTorrent adds is the detection of the traffic it generates by the underlaying network components. Finding a fingerprinting algorithm to recognize the flow is extremely hard when BitTorrent traffic obfuscation is enabled (usually it is because of Internet Service Providers' filtering out of the peer-to-peer traffic [18]). However, since enabling protocol obfuscation requires both parties in the communication to agree to do so, having control of at least one end of the communication can be used to enforce clear traffic between both participants.

2.3 Related work on SDN for Peer-to-Peer Communications

SDN is still an emerging technology and the pace of SDN technology advancement is evolving fast. Both [10] and [11] describe state-of-the art SDN research and the challenges of SDN. Both survey papers present a comprehensive discussion on SDN and mention some potential applications of SDN, e.g., data-center networks, network-as-a-service, cloud computing, network virtualization, and wireless services. However, neither of those two latest survey papers explicitly addresses SDN for peer-to-peer (P2P) communications. Furthermore, a large number of articles have been published recently on the subject of SDN and a great amount of efforts have been put into SDN. Nevertheless, only a few research articles have been reported explicitly on SDN and P2P.

P2P communications, such as BitTorrent traffic, are popular in todays Internet. Hence, it is crucial to investigate these two topics at the same time; namely, how to support P2P with the emerging SDN architecture and the effect of SDN on P2P performance. One of the key benefits of SDN is that SDN enables application-level control/programming of network.

Authors in [9] and [8] argue that SDN is well positioned to support P2P communications among others, as SDN enables access to the network topology information which can support the application-level decision making. The authors also argue that the Application Layer Traffic Optimization (ALTO) [1] provides a standards-based solution that can be used by SDNs to support applications, including P2P communications. The proposed ALTO SDN can allow the BitTorrent client to queries a tracker proxy which uses ALTO SDN information to maintain a list of peers that are more cost-effective for P2P communications. However, as the authors describe, P2P swarms are highly dynamic and can select a number of replicas. As a result, SDN may not offer significant advantage compared to the existing application-level peer selection strategies. Further, actual experiments were presented in the report.

SDN eXchange (SDX) was proposed to support interdomain routing [7]. The authors advocated an innovative architecture to explore how to combine different policies from multiple ASes at an Internet eXchange Point (IXP). One of the benefits of SDX is its potential to facilitate application-specific peering [7]. However, the paper does not address if the approach is appropriate for a number of P2P communications, such as BitTorrent traffic. The paper does not present any experiments based on the SDX architecture. Lin, et al. [12] also

present interdomain routing for SDN peering. But the paper does not scale up to the popular P2P communications that have been used in todays Internet.

3 Reactive Routing stability on Software-Defined Networks

As we explained in previous sections, SDNs' technologies introduce programmatic ways to reorganize the network logical topology and a possible practical field of use of SDNs is Reactive Routing. On Reactive Routing the logical topology is continuously evolving based on traffic statistics such as workload or jitter, which can be collected by the switches.

When looking at how BitTorrent and Reactive Routing interact, one can assume there is no interaction because SDNs layered architectures modify the logical topology in the lower layers (L2/L3) while BitTorrent decisions happen at the Application Layer. Although, layer encapsulation should make these changes in the lower layers transparent to the Application Layer, the reality is far from that in this case. BitTorrent choking strategies are very sensitive to the changes in the underlaying topology, since BitTorrent keeps constantly probing for better paths to saturate every possible channel. The most notorious cases are the Pareto Efficiency and BitTorrent choking algorithms. On the other hand, Reactive Routing is sensitive to upper layers traffic, which should be the main point of Reactive Routing.

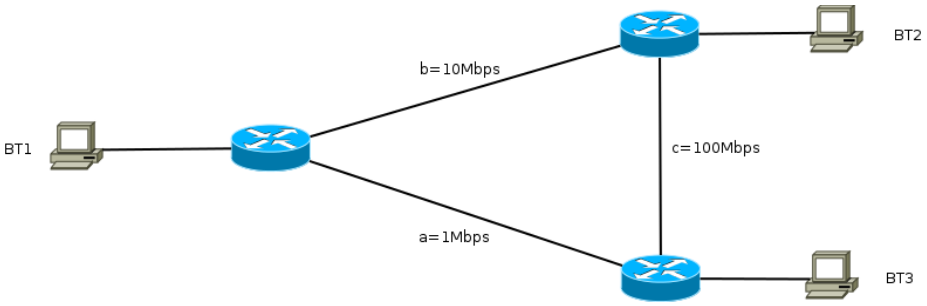

Figure 2: Example topology

Let's hypothesize a possible result of applying Reactive Routing with no information about the protocols used in the higher layers using the example topology presented in Figure 2. For simplicity we assume that all hosts (BT1, BT2, BT3) run BitTorrent and they all are participating in the same swarm, only 2 connections are maintained between peers, paths are symmetric and the policy used by our Reactive Routing application assigns always the oldest flow alive to the best Bandwidth link.

A possible evolution of the traffic and network may start with an initial state, before any transfer out of BT1 starts, where the paths are already defined in the flow table to use link b between BT1-BT2 and link a between BT1-BT3. After selecting the piece, the client in BT1 decides to unchoke BT3. The limit for transfer between these 2 peers is 1Mbps, soon after unchoking BT3, it also unchokes BT2. When the unchoke operation happens, the new transfer for BT1-

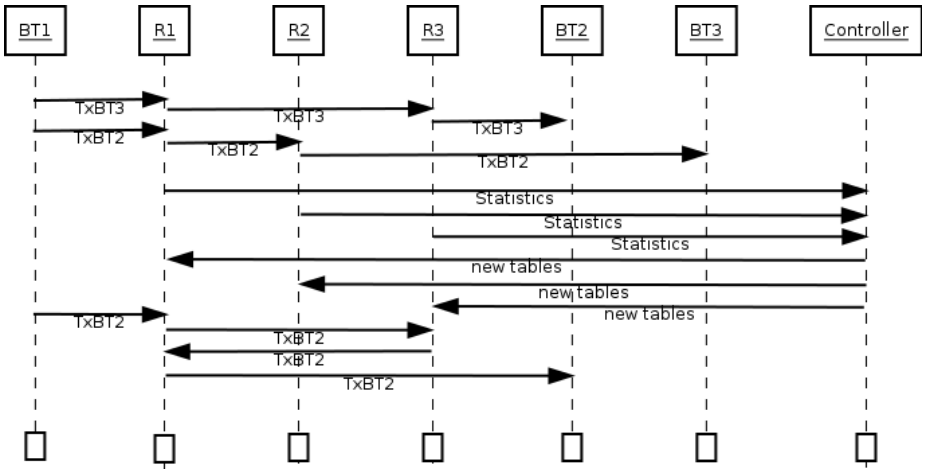

Figure 3: Sequence diagram of the example. In the first transfer to BT2 the path b is used, but after controller updates the flow tables, the path ac is used reducing the throughput.

BT2 flow is using 10Mbps, then the client decides that BT2 is a better option, so in the next round it will choke BT3.

In this example, based on the policy, the controller will move the flows so the first flow uses links *b*-*c* and the second flow uses *a*-*c*. After changing the flows, the connection BT1-BT2 is limited to 1Mbps and the one between BT1-BT3 is limited to 10Mbps. The decision was already taken to choke BT2, therefore they will keep using the lowest possible bandwidth between peers when it is the worst possible action. If the Reactive Routing algorithm chooses again before system stabilizes, it will again choose the wrong one and it will never be able to obtain the max possible bandwidth.

Figure 3 shows a sequence diagram of the example. In this case, the Reactive Routing and BitTorrent interaction leads to worse results than keeping static routing and let BitTorrent find its ways alone.

Another possible source of misled behavior in BitTorrent that may appear when networks have many switches in the path between a pair of nodes, is that the updates of tables to change a path creates micro-cuts of service between the two nodes making the anti-snubbing strategy to trigger when it is not necessary. The triggering of anti-snubbing will effectively close the connection for a long time between two nodes significantly degrading the propagation rate.

As the examples suggest, at the time of using Reactive Routing in BitTorrent traffic carrying networks can have a negative impact in the system. What we do not know from this simplified analysis is how frequently they appear, and if the impact is negligible in practice. In following sections we will discuss some experimental results in emulated networks with real BitTorrent clients.

4 Classification of traffic in a SDN

To organize our experiments, we propose a classification of the possible interactions between nodes and the SDNs in three categories: Fully Internal, Half

Internal and Fully External as shown in Figure 4.

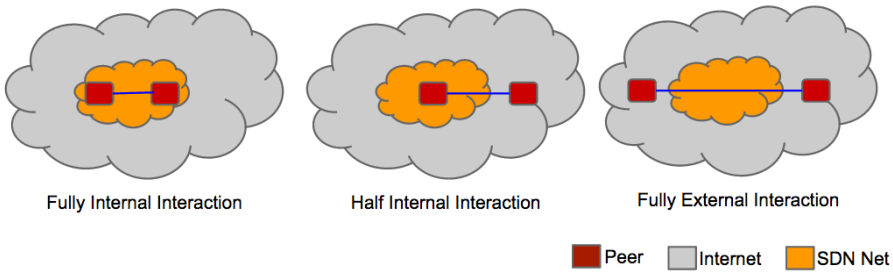

Figure 4: Classification of traffic in SDN.

We consider an interaction to be Fully Internal when both ends of the communication are inside the same SDN domain. An example of this kind of interaction is when the OS upgrade package is distributed to all hosts in the network, or as internal distribution service in a Content Delivery Network location.

We consider an interaction Half Internal (or Half External) when one of the ends of the communication is inside the SDN domain and the other is not. An example of this kind is when using BitTorrent clients internally to distribute content to end users outside the local network as the Amazon S3 service does.

We consider an interaction Fully External when neither of the ends of the communication are inside the SDN domain. This is the case of traversing traffic. An example of this kind is when using SDNs to manage an Internet Service Provider's network. Usually ISPs do not have BitTorrent clients running in their networks, but there may be several connections from users in other networks traversing the local network.

The proposed classification takes in account only one connection between two participants. In practice, BitTorrent clients usually have about 300 open connections for each swarm it is participating. Hence, hybrid scenarios where a client has connections with Fully External, Half Internal and Fully Internal interactions are expected in practice.

5 Experiments

Our first experiment setup is composed of an emulated network of 32 switches connected as a ring where each switch is connected to a single host running a BT client. Every BT client knows the addresses of all other peers and every peer is trying to obtain the same content. The emulated network is isolated and not interaction to external networks is allowed. The bandwidth in each host NIC is limited and the links bandwidth is big enough to avoid saturation of the links. We need the links to avoid saturation in this stage to compare results against the experiments in real networks showed in Figure 1 which never saturated and use them as validation of the emulated network.

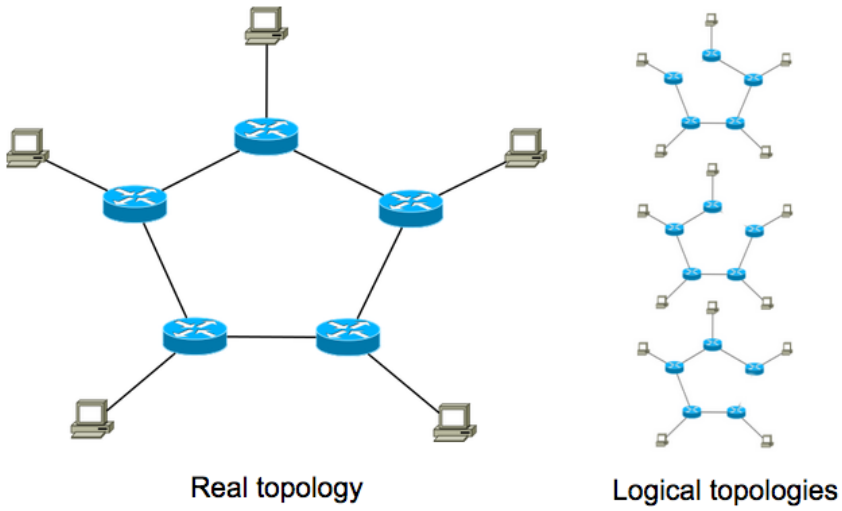

Figure 5: Real and logical topologies of the experiment.

Before we start the emulation, one link is taken down to obtain a loop-less logical topology. One peer is chosen to start providing the content and as soon as this peer enters the swarm all others start requesting pieces. The controller periodically changes the logical topology choosing a new link to remove and putting the previously removed back into place. We chose what link to take down in a Round Robin fashion as showed in Figure 5.

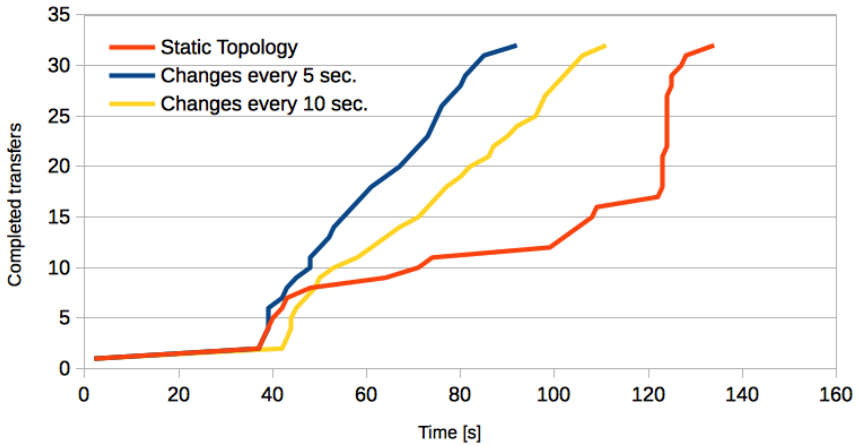

Figure 6: Comparison of propagation of 10MB of content using BitTorrent to 31 nodes using different frequency of topology changes in emulated SDN.

We implemented this emulated network setup in a computer running Ubuntu Linux Server Edition 12.10-x64 using Mininet 2.1.0, the POX included in the

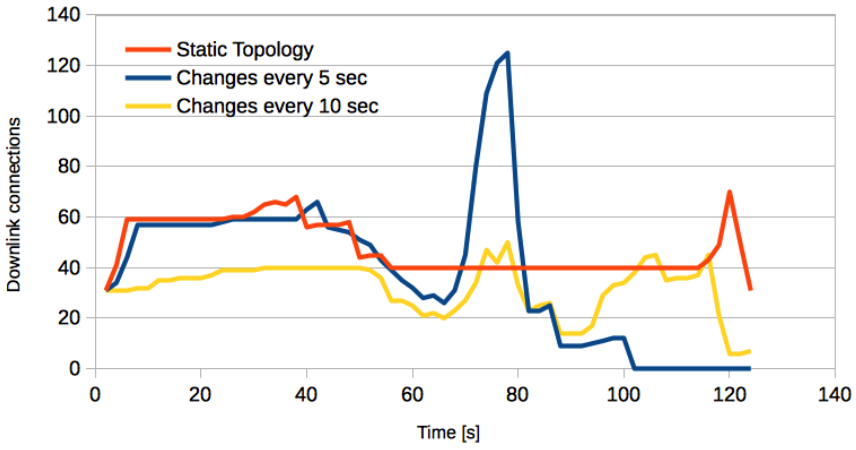

Figure 7: Comparison of simultaneous open connections for download for distribution using BitTorrent of 10MB of content to 31 nodes using different frequency of topology changes in emulated SDN.

same distribution package as the controller, wireshark 1.8.2 for trace captures, and Libtorrent-Rasterbar 0.16.16 using Boost 1.49 as building dependency for the custom BitTorrent clients that we deployed in each emulated node.

We run emulations using period values of 5, 10, 30 and 60 seconds and one run with static configuration as a reference.

In Figure 6 we can see the the static plot curve has some resemblance to the one showed in Figure 1 even when the time is not exactly the same because the content has different size.

Using larger files is limited by hardware resources availability. All nodes run on the same machine, thus the use of larger files will make the emulator swap and ruin the experiments.

In Figure 6 we observe that rotating the logical topology makes BitTorrent more efficient in terms of propagation than keeping the topology static in this particular topology. This was an unexpected result and shows the importance of studying protocols behavior before deploying Reactive Routing solutions in SDNs.

In Figure 7 we show how many download connections are open in the network. We believe the peak of the connections half way the emulation is closely related to the improvement in the propagation.

From this results, even when validated the static topology results, we got reasons to suspect from the emulator, running faster than expected may be an indication of problems. We proceeded then to do the same experiments with more predictable tools.

In this second iteration we used netcat and curl as server and client for HTTP traffic. We put a file in one node and started listening for 31 other nodes using netcat. In each of the other nodes we started downloading the shared file using curl. In this case all nodes transfer at the same time so we needed to lower more the bandwidth limits to keep it reproducible.

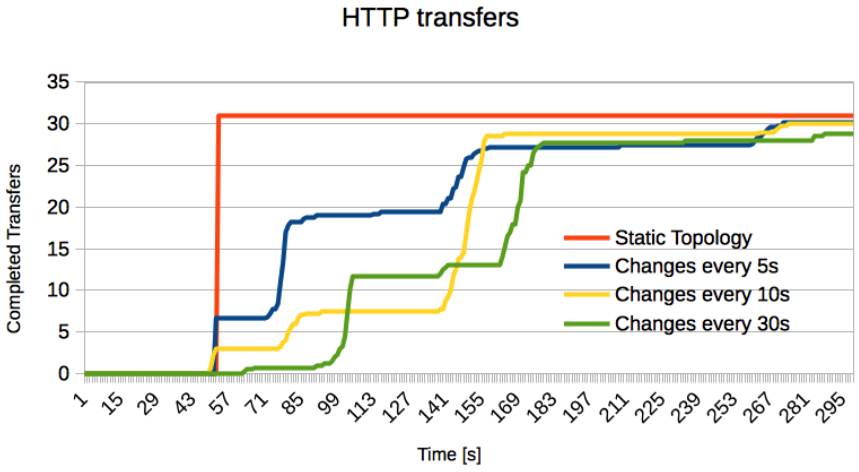

Figure 8: Comparison of propagation of 10MB of content using HTTP to 31 nodes using different frequency of topology changes in emulated SDN.

In Figure 8 we see that when the topology is static, all transfers started together and finished together. But when the topology is changing some nodes took longer to complete. Also in some cases one or two nodes had a broken connection and never complete, both are expected behaviors.

We believe then that the behavior of BitTorrent is properly emulated by the tools we used and there is a concern about the interaction between BitTorrent and the topology change in experiments using SDN. Finishing earlier means that at some point the transfer limits were not respected, which increases saturation in some links.

6 Conclusions

We reviewed the key concepts behind two trendy technologies, SDNs and BitTorrent and we considered the cases where BitTorrent strategies and Reactive Routing may collaborate to produce unexpected results compared to those obtained using static routing.

We described a set of hypothetical scenarios and we proposed a classification for them in three categories based in the kind of interaction the nodes have.

We explained a set of experiments conducted in an emulated ring network using BitTorrent clients and HTTP clients/server.

We compared the static topology results BitTorrent running in an emulated network against those obtained in a previous work in Planet Lab for validation.

When compared the traffic using BitTorrent against analogous experiments using HTTP in the same context and showed that the traffic patterns observed are different. When topology is periodically modified by the SDN controller in the emulated network, BitTorrent traffic pattern is unexpectedly propagating faster than when it had a static topology, while HTTP is slightly lowering propagation.

Finally, we conclude from these experiments that high level protocols, such as BitTorrent, need to be studied carefully before using them in a Reactive Routing network to avoid bouncing effects.

In the long term, we expect to identify the origin of this unexpected results and help controllers to detect this kind of behaviors before applying changes to the topology that may negatively affect the network. We are also planning to extend the experiments to different types of network topologies, e.g., mesh network for further investigation.

References

[1] Alimi, R., Yang, Y., and Penno, R. Application-layer traffic optimization (alto) protocol.

[2] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review 33*, 3 (2003), 3–12.

[3] Cohen, B. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems* (2003), vol. 6, Citeseer, pp. 68–72.

[4] et al., N. Pox/nox (http://www.noxrepo.org/), 2008.

[5] Foundation, L. Open daylight (http://www.opendaylight.org/), 2013.

[6] Foundation, O. N. Openflow (https://www.opennetworking.org/sdn-resources/onf-specifications/openflow), 2013.

[7] Gupta, A., Shahbaz, M., Vanbever, L., Kim, H., Clark, R., Feamster, N., Rexford, J., and Shenker, S. Sdx: A software defined internet exchange.

[8] Gurbani, V. K., Scharf, M., Lakshman, T., Hilt, V., and Marocco, E. Software defined networks (sdn): Leveraging network state for rendezvous services.

[9] Gurbani, V. K., Scharf, M., Lakshman, T., Hilt, V., and Marocco, E. Abstracting network state in software defined networks (sdn) for rendezvous services. In *Communications (ICC), 2012 IEEE International Conference on* (2012), IEEE, pp. 6627–6632.

[10] Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., and Thierry, G. Software-defined networking: Challenges and research opportunities for future internet. *Computer Networks* (2014).

[11] Jammal, M., Singh, T., Shami, A., Asal, R., and Li, Y. Software-defined networking: State of the art and research challenges. *arXiv preprint arXiv:1406.0124* (2014).

[12] Lin, P., Bi, J., Chen, Z., Wang, Y., Hu, H., and Xu, A. We-bridge: West-east bridge for sdn inter-domain network peering.

[13] Nadeau, T. D., and Gray, K. *SDN: Software Defined Networks*. O'Reilly Media, 2013.

[14] Nordberg, A. Rasterbar libtorrent (http://http://www.rasterbar.com/products/libtorrent/). *available on http://www.rasterbar.com/products/libttorrent* (2003).

[15] Peterson, L. L., and Davie, B. S. *Computer networks: a systems approach*. Elsevier, 2007.

[16] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. The bittorrent p2p file-sharing system: Measurements and analysis. *Peer-to-Peer Systems IV* (2005), 205–216.

[17] Project, F. Floodlight (http://www.projectfloodlight.org/), 2011.

[18] Pung, W., and Woodward, A. Can current packet analysis software detect bittorrent activity or extract files from btp and μtp traffic streams? secau Security Research Centre, Edith Cowan University, Perth, Western Australia.

[19] Telegraph, N., and Corporation, T. Ryu (http://osrg.github.io/ryu/), 2013.

[20] Vicino, D., Lung, C.-H., Wainer, G., and Dalle, O. Evaluating the impact of software-defined networks reactive routing on bittorrent performance. *Procedia Computer Science 34* (2014), 668–673.

[21] Vicino, D., Timpanaro, J. P., Chrisment, I., and Righetti, C. Using kad-bittorrent hybrid clients to share contents., 2012.

This paper is a preprint of a paper accepted by IET Networks and is subject to Institution of Engineering and Technology Copyright. When the final version is published, the copy of record will be available at IET Digital Library