Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 34 (2014) 668 – 673

Procedia Computer Science

International Workshop on Software Defined Networks for a New Generation of Applications and Services (SDN-NGAS-2014)

Evaluating the impact of Software-Defined Networks' Reactive Routing on BitTorrent performance.

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

[a]*Dept. of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada*
[b]*Laboratoire I3S UMR CNRS 7271, Sophia Antipolis, PACA, France*

Abstract

Software-Defined Networks' technologies introduce programmatic ways to reorganize the network logical topology. To achieve this, the switches in the network interact with a set of controllers, these controllers can dynamically update the switches configuration based in received events. A possible practical field of use of Software-Defined Networks' is the one called Reactive Routing. On Reactive Routing the logical topology is continuously evolving based on traffic statistics as load or jitter which can be collected by the switches. BitTorrent is a well-known peer-to-peer protocol used on application layer to achieve fast propagation of content. In an effort to find the best set of connections that maximizes the global aggregation of throughput without knowledge from underlying topology, BitTorrent uses choking algorithms that continuously open and close connections to different peers. Software Defined Networks implementing Reactive Routing may be negatively affecting the performances of the system under specific conditions because of it lack of knowledge of BitTorrent strategies. Here, we review the concepts involved in Software-Defined Networks and BitTorrent protocol, we propose a classification of scenarios where these technologies may interact, we discuss hypotheses about possible problems arising from these interaction and we show our preliminary experimental results from study the phenomena.

Selection and peer-review under responsibility of Conference Program Chairs

Keywords: Software-Defined Networks; BitTorrent; Reactive Routing

1. Introduction

Software-Defined Networks (alternatively called Software-Driven Networks) is to provide management flexibility and automation introducing programmability mechanisms to the network components. This mechanisms are necessary to allow the network to react fast and be able to apply changes to the logical topology when needed to cover specific

* Damián Vicino. Tel.: +1-613-520-2600
E-mail address: damian.vicino@carleton.ca

Selection and peer-review under responsibility of Conference Program Chairs
doi:10.1016/j.procs.2014.07.095

requirements. SDN can be used to provide Reactive Routing, on Reactive Routing the logical topology is continuously evolving based on traffic statistics as load or jitter which can be collected by the switches.

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 the time this separation splits the Network Layer[1] in two, but sometimes other layers are also involved too (apparently layer encapsulation is not the most popular and respected concept in Software-Defined Networks' development communities).

BitTorrent is a well-known peer-to-peer protocol used on application layer to achieve fast propagation of content. In an effort to find the best set of connections that maximizes the global aggregation of throughput without knowledge from underlying topology, BitTorrent uses chocking algorithms that continuously open and close connections to different peers.

Our motivation is study the applicability of SDNs' Reactive Routing ideas to different network traffic scenarios. The implementation of Reactive Routing may negatively affect performances of the system under specific conditions because of it lack of knowledge of BitTorrent strategies. In this particular work, we focus its interaction with BitTorrent[2], a well-known peer-to-peer protocol used on application layer[1] to achieve fast propagation of content. In future work, we expect to construct ifuture solid Reactive Routing implementations that doesn't reduce the network performance when the higher layer protocol in use is BitTorrent or similar.

Our current goal is to identify the sources of degradation of performance and evaluate the degradation produced from the blindly interaction between SDN Reactive Routing algorithms and BitTorrent.

Our work classifies the interaction problems in 3 categories: Fully Internal, Fully External and Half Internal depending on the location of the participants of the communication being deployed inside of the local network or not. We discuss the problems arising from the interaction in the Fully Internal and we show some preliminary results from our experiments.

2. Background

The Open Networking Foundation defines SDN as "the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices"[3]. This definition states an objective, but it doesnt tell about the means to achieve it. SDNs are not standardized, but existing projects agree in using a layered architecture with 3 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.

Project	Southbound Protocols	Northbound APIs	Lang.
NOX/POX[4]	OpenFlow 1.0	undefined	C+++ Python
RYU[5]	OpenFlow 1.0 OpenFlow 1.3	undefined	Python
Floodlight[6]	OpenFlow 1.0	Java RESTful	Java
OpenDaylight[7]	OpenFlow 1.0 OpenFlow 1.3	OSGI RESTful	Java

Fig. 1. Comparison of Open Source Controller projects.

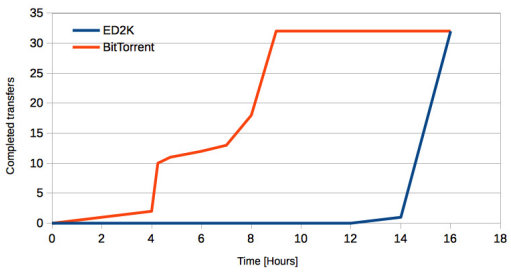

Fig. 2. Compare of propagation of a 300MB in PlanetLab using aMule (eDonkey-2000) and hMule (BitTorrent).

Since the architecture is layered, it needs to define communication between the connected layers to provide services. In the case of communication, between the Infrastructure Layer and the Control Layer, there is as many protocols as switches' manufacturers. To increase compatibility between devices from different manufacturers, easy 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 SDN the controllers[8] are the most important piece, they 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, discovery 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.

Most peer-to-peer content distribution protocols (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[2]. 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[9]. In Figure 2 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 BitTorrent alternative[10].

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 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 action temporary refuse transferring 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 3 cycles, a single peer is unchoked to check if there is a better option than keep going with those peers already unchoked.
  - Anti-snubbing: When not data was 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 to decide is restricted to the content and the known peers, there is no metrics about current traffic involved. We assume this strategies shouldn't be influenced by changes in the underlying network.

On Choking Algorithms strategies decisions are based in 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 underlay topology changes using SDNs Reactive Routing approach and fail to select the best connections available.

3. BitTorrent and Reactive Routing on Software-Defined Networks

As we explained in previous sections, SDNs' technologies introduce programmatic ways to reorganize the network logical topology. A possible practical field of use of SDNs' Reactive Routing. On Reactive Routing the logical topology is continuously evolving based on traffic statistics as load 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 on the Application Layer. Although, layer encapsulation should make them transparent, the reality is far from that because BitTorrent choking strategies are very sensitive to the changes in the underlaying topology since it 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 sensitivity to upper layers traffic is the whole point of Reactive Routing.

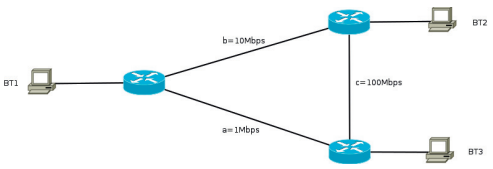

Fig. 3. Example topology

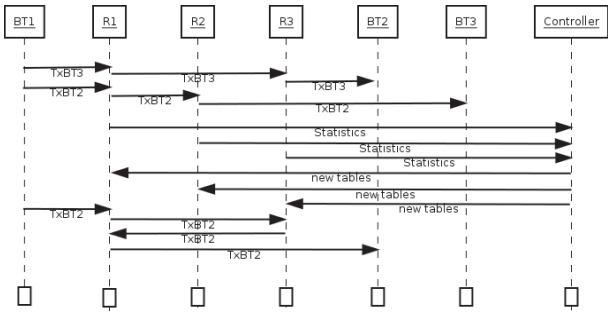

Fig. 4. Sequence diagram of the example. In the first transfer to BT2 the path b is used, but after controller updates the flow tables, the path ac is used reducing the throughput.

Let's assume we need to study the example presented in Figure 3. Let's 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. In the initial state, before any transfer out of BT1 starts, 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 happens, the new transfer is using 10Mbps, then the client decides that BT2 is a better option, so in the next round it will choke BT3. 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 taken to choke BT2, therefor they will keep using the lowest possible bandwidth between peers. Figure 4 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 2 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 2 nodes significantly degrading the propagation rate.

Our analysis of the examples suggest that (blindly) Reactive Routing the BitTorrent traffic carrying networks can have a negative impact in the system. What we do not know is how frequently they appear, and if the impact is negligible in practice.

To reduce the complexity of the experiments, we propose a classification of the possible interactions between BitTorrent and the SDNs in 3 categories: Fully Internal, Half Internal and Fully External. We consider an interaction to be Fully Internal when both BitTorrent participants 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 BitTorrent participants 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 network as the Amazon S3 service does. We consider an interaction Fully External when neither of the BitTorrent participants are inside the SDN domain, this is the case of traversing BitTorrent traffic. An example of this kind is when using SDNs to manage an Internet Service Provider's network, usually they dont have BitTorrent clients running in their networks, but they have several users running them.

The proposed classification takes in account only 1 connection between 2 participants. BitTorrent clients usually have about 300 open connections for each swarm it is participating, so hybrid scenarios where a client has connections with Fully External, Half Internal and Fully Internal interactions are expected in practice.

In the case that experiments show the problem is significant enough to worry about, two different approaches can be taken to reduce it: making BitTorrent clients aware of the Reactive Routing policies or making the Controllers aware of the BitTorrent activities. The first option needs custom clients for all the participants which may not even be in the same Autonomous System. In the case we want to make the controller aware of the BitTorrent clients we have some alternatives: We could modify the BitTorrent clients to signal the controller about how longer a path is planned to be used. This is a "resource allocation" approach and still needs modification in the clients, although it is less intrusive than previous approach, still has the same drawbacks. Another possibility is to add application logic that interacts with the controller processing traffic log and statistics (possible since OpenFlow 1.3) to fingerprint the traffic and detect the BitTorrent messages without including special logic in the client. The problem with this approach is that find a fingerprinting algorithm to recognize the flow is extremly hard when BitTorrent traffic obfuscation is enabled (as usually is because of Internet Service Providers filtering the peer-to-peer traffic [11]). Anyway, since enabling protocol obfuscation requires both parties in the communication to agree to do so, it is still possible to use this approach in the case of Fully Internal and Half Internal forcing traffic to be clear from the end inside the owned network. A third possibility is to create a new network virtual device that can be added to the Service Path, similarly to a Firewall or a NAT virtual component to fingerprint the traversing traffic and signal the controller. This is a mix of the previous options: the virtual device only increases the scalability and performance of the solution, but has same limitations as previous one based in the fingerprinting problems.

4. 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 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 2 which never saturated and use them as validation of the emulated network.

We implemented this network setup in Mininet and we used POX as controller. 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 like to take down in a Round Robin fashion.

We run emulations using period values of 5, 10, 30 and 60 seconds and one run with static configuration to use as reference. The following results are still preliminary given that we only obtained 10 repetitions of each configuration at the submission of this paper.

In Figure 5 we can see the the static plot curve has some resemblance to the one showed in Figure 2 even when the time is not the same because the content has different size. Using larger files is limited by hardware resources availability. All nodes run in same machine, thus the use of larger files will make the emulator swap and ruin the experiments.

In Figure 5 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 results for us and we are currently working on understanding which one of the BitTorrent Strategy is responsible of this improve.

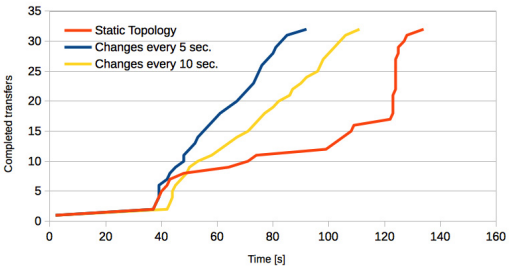

Fig. 5. Compare of propagation of a 10MB content to 32 nodes using different frequency of topology changes.

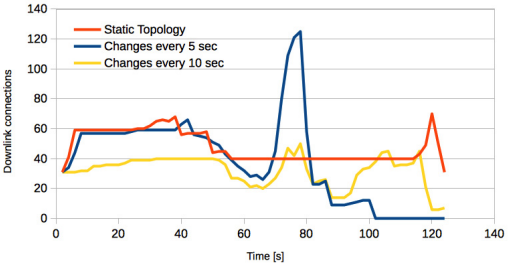

Fig. 6. Compare of simultaneous open connections for download for distribution of a 10MB content to 32 nodes using different frequency of topology changes.

In Figure 6 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 improve in the propagation. We still need to complete the processing the traces to confirm and understand the phenomena.

5. 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, use cases, and we proposed a classification for them in 3 categories based in the kind of interaction the nodes have.

We showed partial results of experiments on ring topology with an SDN controller periodically changing the logical topology confuse the BitTorrent strategies making them affect the propagation and connections distribution.

In the near future we plan to complete of current experiments and analyze the results to figure out which are the strategies taking the major role in the propagation improvement. Also, we plan to experiment in a indirect-torus topology where the same logical topology can be applied using several different links and maintain exactly same delay and bandwidth between the pair of peers. And finally start working with current Reactive Routing implementations based in OpenFlow 1.3 stats.

In the long term, we expect to identify which algorithms are affected and how by the topology changes and implement the previously proposed solutions for OpenDaylight controller to positively collaborate with the peers.

References

1. Peterson, L.L., Davie, B.S.. *Computer networks: a systems approach*. Elsevier; 2007.
2. Cohen, B.. Incentives build robustness in bittorrent. In: *Workshop on Economics of Peer-to-Peer systems*; vol. 6. Citeseer; 2003, p. 68–72.
3. Foundation, O.N.. Openflow (https://www.opennetworking.org/sdn-resources/onf-specifications/openflow). 2013.
4. et al., N.. Pox/nox (http://www.noxrepo.org/). 2008.
5. Telegraph, N., Corporation, T.. Ryu (http://osrg.github.io/ryu/). 2013.
6. Project, F.. Floodlight (http://www.projectfloodlight.org/). 2011.
7. Foundation, L.. Open daylight (http://www.opendaylight.org/). 2013.
8. Nadeau, T.D., Gray, K.. *SDN: Software Defined Networks*. O'Reilly Media; 2013.
9. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.. The bittorrent p2p file-sharing system: Measurements and analysis. *Peer-to-Peer Systems IV* 2005;:205–216.
10. Vicino, D., Timpanaro, J.P., Chrisment, I., Righetti, C.. Using kad-bittorrent hybrid clients to share contents. 2012.
11. Pung, W., 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; 2011, .