Improved Path Compression for Explicit Path Control in Production Data Centers

Shuihai Hu[1,2], Kai Chen[1], Gaoxiong Zeng[1,2]
[1]Hong Kong University of Science and Technology [2]Student

{*shuaa,kaichen,gzengaa*}*@cse.ust.hk*

1 Introduction

Explicit path control is required by many data center applications, from fine-grained traffic engineering, performance guarantees, to network diagnosis and troubleshooting. To satisfy the requirement, XPath [1] has recently been proposed for implementing explicit path control in data center networks. The core idea of XPath is to explicitly identifies an end-to-end path with a path ID, and pre-install all the desired paths into TCAM-based IP LPM (longest prefix matching) tables of commodity switches via path compression. With explicit path control, XPath directly benefits the above applications by providing them with flexibility of using desirable end-to-end routing paths explicitly.

However, we find that XPath exposes limitations when we use it in practice. First, XPath was designed to work for arbitrary topologies. But in reality, production data centers such as those in Microsoft, Google and Facebook typically adopt multi-level multi-rooted trees as their network topologies. Thus, XPath has failed to utilize the properties provided by real data center network (DCN) topologies. Second, XPath may require over 100K routing entries (prefixes) in order to pre-install the desired paths. Although some switches can already hold over 100K entries, many commodity switches can still only support 10-20K entries. Therefore, XPath is insufficient to work with those commodity switches at large scale.

Motivated by these limitations, we propose XPath* to complement XPath in production networks. The key idea of XPath* is to leverage multi-rooted tree topologies to significantly optimize the path compression, so that it can be used with any commodity switches at large scale.

2 Algorithm

We compute the desired paths for a given multi-rooted tree topology by constructing multiple spanning trees (ST) rooted at different non-edge switches. Each ST covers a subset of paths traversing the same root switch, and is uniquely identified by a $treeid$. The $treeid$ of an ST is used as a common prefix shared by all the IDs of paths under the ST. The path ID we assign to each path is in the form of $treeid.nodeid$, where $treeid$ identifies the ST a path belongs to, while $nodeid$ indicates the location of this path's destination in the topology.

DCNs	Paths #	**Max. entries # with XPath**	**Max. entries # with XPath***
Fat-tree(8)	**15,872**	**116**	**24**
Fat-tree(16)	**1,040,384**	**968**	**80**
Fat-tree(32)	**66,977,792**	**7,952**	**288**
Fat-tree(64)	**4,292,870,144**	**64,544**	**1,088**
VL2$(20, 8, 20)$	**31,200**	**310**	**120**
VL2$(40, 16, 20)$	**1,017,600**	**2,820**	**440**
VL2$(80, 64, 20)$	**130,969,600**	**49,640**	**1,680**
VL2$(100, 96, 20)$	**575,760,000**	**117,550**	**2,600**

Table 1: Path aggregation performance on the two well-known tree-based DCNs.

Under our path ID assignment, all the end-to-end paths in an ST only need one prefix entry like $treeid$.x.y.z/8 to forward the packets to the root switch of the ST. On the other hand, as we can leverage hierarchical assignment scheme to decide the value of $nodeid$ for each destination, the number of prefix entries needed by downward switch ports can also be significantly reduced.

3 Evaluation

We evaluate the scalability of XPath* on 2 well-known production DCN topologies: Fat-tree and VL2. Our evaluation covers $k^2/4$ paths between any two ToRs in Fat-tree(k) and D_A paths between any two ToRs in VL2(D_A, D_I, T). As we can see from the last two columns in Table 1, XPath* can effectively pre-install up to tens of billions of paths using only a few thousands of forwarding entries for large DCNs, while XPath requires tens to hundreds of thousands of forwarding entries for the same scale. Specifically, for Fat-tree(64), XPath* encodes 4 billion paths with only about $1K$ entries, while XPath requires about $64K$ entries; for VL2$(100, 96, 20)$, XPath* encodes 575 million paths with only $2.6K$ entries, while XPath requires about $117K$ entries.

References

[1] S. Hu, K. Chen, H. Wu, W. Bai, C. Lan, H. Wang, H. Zhao, and C. Guo, "Explicit path control in commodity data centers: Design and applications," in *NSDI'15*.