Cross-Entrance Consistent Range Classifier with OpenFlow[*]

Yehuda Afek
Tel Aviv University
Tel Aviv, Israel
afek@post.tau.ac.il

Anat Bremler-Barr
The Interdisciplinary Center
Hertzelia, Israel
bremler@idc.ac.il

Liron Schiff
Tel Aviv University
Tel Aviv, Israel
schiffli@post.tau.ac.il

1 Introduction

In this paper we present a new Openflow based architecture to manage flows across a *multi entrance* SDN network in a consistent way, thus improving in several aspects on previous works [4, 5]. Our contributions are in three levels. At the first level we use OpenFlow features in a sophisticated way to implement a range classification scheme which to the best of our knowledge is more space efficient (only 3 entries per range) than previous known classifiers. In the second contribution we show how to update ranges across multiple switches in an atomic manner - allows to update the set of ranges and their associated actions while packets are classified and the network is changing. Finally, using the two schemes above, we present an architecture suitable for several applications such as load-balancing, and NFV, to manage multi-entrance consistency - keeping Per Flow consistency even when the flow changes the entrance point to our network.

Our scheme utilizes advanced OpenFlow features to efficiently implement a complex computation of a non-overlapping range classification scheme which requires only $3n+2w+1$ table entries (a significant saving is observed already with 4 ranges - see Figure 1(b)), where n is the number of ranges and w is the size of values (e.g. 32 for IPv4 addresses). The three key OpenFlow features (and limitations) that we use are: (i) a packet can be processed by several forwarding tables using 'goto' action command to decide on the next table. (ii) a packet can be extended with an auxiliary field which may be altered (e.g., xored with a constant) and considered as the packet goes through subsequent flow tables. (iii) Openflow lack of support for general computation on packet field values.

Three levels of atomicity are considered and supported by our scheme: (i) Per packet consistency - each packet is handled according to correct configuration either before or after the update. (ii) Per flow Consistency - all packets of a flow are handled according to the same configuration and new flows are handled according to the newest configuration. (iii) Cross-entrance consistency - keeping Per Flow consistency even when the flow changes the entrance point to our network.

Our ranges classification is based on PIDR, a non-OpenFlow system design by Panigrahy and Sharma [3] which uses special hardware (e.g., ASIC or FPGA) and has no notion of atomicity. PIDR creates two patterns for each range called the ELCP0 and ELCP1 of the range (see definition in [3]) and save them separately in two TCAMs associated with range bounds. The two TCAMs are arranged in a very specific manner to support the scheme. In order to classify a value, it should be queried against the two TCAMs, returning two matches that are associated with two possible ranges. Then the value is checked whether it belongs to one of these two ranges.

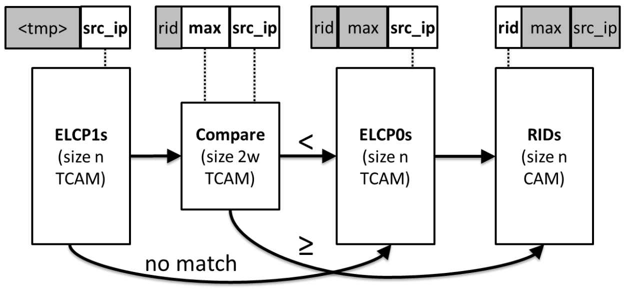
2 OpenFlow Range Classifier

We start by briefly explaining the conversion of the original PIDR design [3] to OpenFlow, later we note important extensions for atomicity and efficiency. Clearly the two TCAMs of PIDR design are converted to flow tables [1] but the main obstacles are the queries to both TCAMs and checking each one of the queries' results - two ranges, if they contain the queried field, by comparing queried field to the ranges' lower or upper bound. To overcome the compare obstacle we utilize a 3rd forwarding table with static entries which act as a comparator.

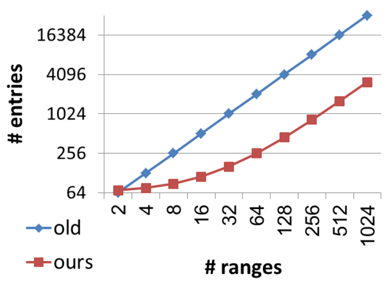
Our ranges implementation uses four TCAMs as depicted in Figure 1(a). Two TCAMS, ELCP0 and ELCP1 are populated with the lower bounds and upper bounds respectively of the ranges as computed by Panigrahy and Sharma. Each entry in ELCPS1 is associated with an action that saves the id and the upper bound of the relevant range in the

[*]Supported by European Research Council (ERC) Starting Grant no. 259085.

[1]Note that the original design also requires to maintain longest prefix first order among ELCP rules in TCAMs which comply by utilizing rule priority supported by OpenFlow forwarding tables (setting priority to the prefix length).



(a)



(b)

Figure 1: (a) Packet range classification process. (b) Log-log flow table entries as a function of number of ranges.

packet temp field and a goto-action to the COMPARE table, and each entry in ELCPS0 is associated with an action to save the id of the relevant range in the temp field and goto the RIDS table. The RIDS table associates the range id in the temp field with the desired action of the relevant range.

The COMPARE table inspects and compares the upper bound in the temp field with the searched field. It contains $2w$ entries. For $0 \leq i < w$, each two entries $2i$ and $2i+1$ considers the $(w-i)$-th MSB of the fields. Even entries are successfully matched when temp field is larger than the searched field and associated with the "goto RIDS table" action. Odd entries are successfully matched when temp field is smaller than the searched field and associated with the "goto ELCP0" action.

Using last construction, packet processing goes as follows: Packet (searched field, e.g., source IP address) is queried in ELCPS1, the matched entry writes the range's id and upper bound in a temp-field and performs "goto Compare table". If the upper bound $\geq$ searched-field a "goto RIDs table" is performed and there the relevant range's action is performed, otherwise a "goto ELCPS0 table" is performed where a new range id is written and "goto RIDs table".

3 Status

We have designed atomic implementation of basic operations in the 3 consistency models, the operations are merge, split and range update. From these operations we are developing an efficient OpenFlow based load-balancer. We also found methods by which our design is improved in the future with the introduction of the following new OpenFlow models: DevoFlow [2], which allows us to save control plane traffic for Per-Flow consistency (a similar idea is suggested in [4]), and the model RMT (recongurable match tables) [1], can be used to replace the Compare table with a computation stage.

References

[1] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. *SIGCOMM Comput. Commun. Rev.*, 43(4):99–110, August 2013.

[2] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. Devoflow: Scaling flow management for high-performance networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):254–265, August 2011.

[3] Rina Panigrahy and Samar Sharma. Sorting and searching using ternary cams. *IEEE Micro*, 23:44–53, January 2003.

[4] Mark Reitblatt, Nate Foster, Jennifer Rexford, Cole Schlesinger, and David Walker. Abstractions for network update. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 323–334, New York, NY, USA, 2012. ACM.

[5] Richard Wang, Dana Butnariu, and Jennifer Rexford. Openflow-based server load balancing gone wild. In *Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE'11, pages 12–12, Berkeley, CA, USA, 2011. USENIX Association.