Flow Aggregation Mechanism for Flow-Aware Multi-Topology Adaptive Routing

Jerzy Domżał, Piotr Jurkiewicz, Piotr Gawłowicz, and Robert Wójcik

***Abstract*— Today, the notion of treating the network traffic as flows is popular. The analyzed flow-based architectures are focused on access networks, because for the core, there is simply too many flows to efficiently deal with. Therefore, an important aspect of flow-based networking is flow aggregation. We show that it is possible and profitable to use aggregates of flows in flow-aware multi-topology adaptive routing, which is a new multipath solution for the IP networks. By aggregating flows, core routers do not need to maintain full flow tables, which makes them less expensive to build and operate. Results show that the proposed aggregation mechanism provides the same overall network performance with flow tables significantly smaller in size.**

***Index Terms*— Aggregation, multipath, routing, traffic engineering, flow.**

I. Introduction

FLOW-BASED routing, where the traffic is identified and routed as flows, can provide multipath and load-balancing capabilities. An example of such a routing mechanism is Flow-Aware Multi-Topology Adaptive Routing (FAMTAR) [1]. FAMTAR is a new multipath adaptive routing mechanism that works based on the currently popular concept of flows. FAMTAR provides the ability to use multiple paths between two points in a network. The key idea of FAMTAR, which distinguishes it from the other multipath approaches, is to use alternative paths only if necessary and only for new flows. Upon congestion, new flows use alternative paths, whereas the old ones remain on their primary paths. Alternative paths are found and used dynamically as needed. FAMTAR was extensively compared to other multi-path solutions in [2]. Further details are presented in Section II.

One of the most important problems, which blocks the introduction of flow-based networks in general, is scalability. In large networks, composed of many nodes and equipped with high speed links, millions of flows are served at the same time in each device. When nodes have to maintain the lists of flows, it is difficult to perform quickly and efficiently. Routers need sufficiently large memories to maintain their flow lists. An even bigger problem is scanning these lists for a particular entry with every incoming packet. Those actions significantly affect the device and increase its response time. This is the reason why Software-Defined Networks (SDN), which is a popular flow-based concept was initially deemed unsuitable for core networks [3]. To solve this problem, flow aggregation mechanisms were proposed in the literature. They provided the possibility to group flows into aggregates which decreases the number of entries in routers' flow lists.

In [4] the authors address the problem of limited memory sizes in SDN switches by proposing flow aggregation. Particularly, they propose a solution which uses slicing to produce an instruction-independent partition and applies a bit and subset weaving to merge flow entries in each subset of a partition. The authors compare their proposal with another flow aggregation scheme called Fast Flow Table Aggregation [5] and show that the average compression ratio of their proposal is better. Flow aggregation is also considered in the event of congestions. In [6], a Local Fast Reroute (LFR) algorithm with flow aggregation is proposed. When a link failure is noticed, the idea is to merge all traffic affected by the failure and create a new aggregated big flow. The evaluation presented in the letter shows that LFR enables fast recovery while minimizing the total number of flow entries in an SDN switch.

Using flow aggregates provides problems as was noticed in [7]. The authors observe that when aggregates are constructed as traffic engineering targets, variations of traffic rates in each aggregate may appear. They propose two methods for generating aggregates: one that is based on a greedy algorithm that minimizes the variation in rates, and the other that clusters flows with similar traffic variation patterns into groups.

The presented mechanisms were designed for SDN given that it is a popular approach. However, these methods cannot be applied to FAMTAR, or other distributed approaches, since there is no signaling and no central controller. In this letter, we propose the Flow Aggregation Mechanism for FAMTAR (FAMF) which reduces the number of flow entries by several orders of magnitude and does not rely on additional signaling between end-nodes. By aggregating flows, core routers do not need to maintain full flow tables which makes them less expensive to build and operate. Simulation results, presented in this letter, show that the proposed flow aggregation method provides similar overall network performance with significantly smaller flow tables. The reduction of the required size of flow tables enables FAMTAR to be applied in core networks.

II. FAMTAR

FAMTAR is a new multipath adaptive routing mechanism introduced in [1] that works based on the currently popular concept of flows. FAMTAR can work with every routing protocol which is responsible for finding the best paths between any two endpoints. When there are no congestions in a network,

Manuscript received July 4, 2017; revised August 17, 2017; accepted August 21, 2017. Date of publication September 1, 2017; date of current version December 8, 2017. This research was carried out with the support of the project "High quality, reliable transmission in multilayer optical networks based on the Flow-Aware Networking concept" founded by the Polish National Science Centre under the project no. DEC-2011/01/D/ST7/03131. The associate editor coordinating the review of this paper and approving it for publication was T. Han. *(Corresponding author: Jerzy Domżał.)*

The authors are with the Department of Telecommunications, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: jdomzal@kt.agh.edu.pl).

Digital Object Identifier 10.1109/LCOMM.2017.2748101

all transmissions use these paths. However, the optimal paths change according to the current congestion status of links. When a link becomes congested, all new flows that would traverse via the congested link are pushed to a new path, while flows which are already active remain on their path. Therefore, FAMTAR is responsible for using the optimal paths provided by the routing protocol, and automatically finding new paths in case of congestion.

To achieve that, a FAMTAR router maintains Flow Forwarding Table (FFT) alongside a classic routing table. For each flow, the corresponding FFT entry indicates the interface to which packets of this flow are to be forwarded. This information is taken from the current routing table when the flow is added to the FFT, i.e., when its first packet appears. FFT is used to realize most packet routing tasks, as for flows that are present on the FFT, the outgoing interface is taken from there and the routing table is not consulted. Entries in the FFT are static and they do not change even when the routing table changes.

When a state close to congestion is noticed on one of the links, the corresponding router increases the cost of this link by a predefined high value. From this moment, this link is perceived by FAMTAR as congested. The new cost appears as a standard change in the used routing protocol, which disseminates this information. Upon receiving this information, routers compute new paths which are likely to avoid congested links. The newly computed paths are stored in the routing tables. However, these changes affect only new flows. The already active flows are still routed on their existing paths stored in the FFT. Congested links still forward all the flows which were active before the congestion was noticed, however, new transmissions do not appear. After a while, when the congestion on the link stops, the original cost of the link is restored. Note that FAMTAR requires a router to monitor its links and to detect congestion on them. The method to determine the congestion is not specified and any congestion indicator can be used, such as link load, queue occupancy, packet queuing delay, and so on.

In case of a link or node failure, active flows, with identifiers registered to the FFT, are still routed according to their old routes stored in the FFT. As a result, in such a case, active flows can still be routed to the failed links. This problem was solved in [8] by introducing a TTL-based mechanism which can detect failures and routing loops and act accordingly.

III. Flow Aggregation Mechanism for FAMTAR

The Flow Aggregation Mechanism for FAMTAR (FAMF) assumes that only border nodes of autonomous systems maintain flow lists, which means that they are not present in the core nodes. Edge routers are able to operate with flow lists because the number of flows they need to process is significantly lower than for similar routers in the core.

The border routers in FAMF have to maintain two lists:

- a Flow Forwarding Table (FFT) which contains flow identifier (*flow_id*), aggregate identifier (*aggr_id*), output interface (*int_id*) and arriving time of last packet (*timestamp*),
- an Aggregation Table (AT) which contains aggregate identifier (*aggr_id*) and output interface (*int_id*).

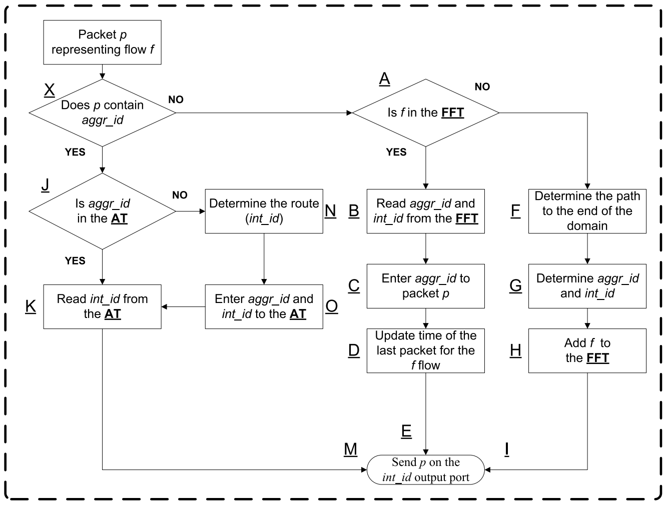
Packet p representing flow f
X
Does p contain aggr_id
NO
A
Is f in the FFT
NO
YES
YES
J
Is aggr_id in the AT
NO
Determine the route (int_id)
N
B
Read aggr_id and int_id from the FFT
F
Determine the path to the end of the domain
YES
C
Enter aggr_id to packet p
G
Determine aggr_id and int_id
K
Read int_id from the AT
Enter aggr_id and int_id to the AT
O
D
Update time of the last packet for the f flow
H
Add f to the FFT
E
M
I
Send p on the int_id output port

Fig. 1. Block diagram of the packets handling in edge routers in FAMF.

FFT is required to process packets that do not yet contain *aggr_id*. These are the packets that are incoming to the domain. The Aggregation Table is required to process packets that already contain the *aggr_id*. The core routers are equipped with only ATs as FFTs are not necessary.

The essence of the proposed mechanism is a method of establishing the entire route of a flow in an edge router of an autonomous system. The route can be determined based information collected by a routing protocol, e.g. by OSPF. In OSPF, all routers within an area have the exact link-state database. After each update, routers run Dijkstra algorithm to determine the shortest path. While they put only outgoing interface to the routing table, they also know the strict path to the destination node. The route is represented by the identifier (*aggr_id*) and is recorded to FFT. Moreover, it is also added to AT along with the output identifier for the selected path. All core routers maintain only ATs. The *aggr_id* is a concatenation of all routers IDs on the path from the edge router's perspective. To be more efficient, instead of concatenation a one-way hash function, such as *md5* or*sha1* can be used. It is not strictly decided which function should be used. However, we indicate that implementing a hash function will allow to minimize the amount of memory used by ATs.

The *aggr_id* used in AT is written to a packet header when it arrives at a border router. This value is equal to *aggr_id* from FFT, determined for flow represented by the incoming packet. In further routers on the path, this value is not changed. Based on the AT table, core routers always know where to send the packet. With the IPv4 protocol, the Type of Service field (Differentiated Services Field acc. to RFC 2474) can be used to write the *aggr_id*. With the IPv6 protocol, the Traffic Class field or a 20-bit Label Flow field can be used.

The detailed operation of the FAMF is presented in Fig. 1. The first step is to check whether the incoming packet contains the *aggr_id* (**X**). Then, depending on whether the packet contains the *aggr_id* or not, the router acts as follows:

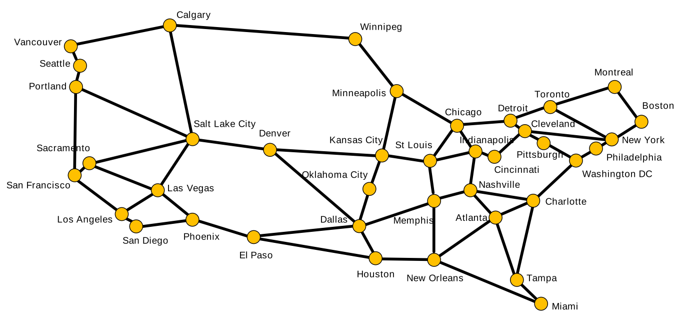

Fig. 2. The topology of simulated network.

For each packet without *aggr_id*, (entering a domain):

1) The router checks whether the flow ID associated with the packet exists in FFT (**A**).
2) If the flow ID is in the FFT:
   - The router reads the *aggr_id* from the FFT and also reads the output interface (*int_id*) (**B**).
   - The router writes the *aggr_id* to the packet (**C**).
   - The router updates the *timestamp* field by putting the current timestamp (**D**).
   - The router forwards the packet to the output interface that was read from the FFT (**E**).
3) If the flow ID is not in the FFT:
   - The router determines the path to the end of domain (to the destination node or the last node in domain, if the destination node is outside domain). The path is represented as a list of nodes (**F**).
   - The router determines the *aggr_id* and also reads the output interface for packets (**G**).
   - The router creates a new entry in the FFT and puts the designated values in the appropriate fields (**H**).
   - The router forwards the packet to the designated output interface (*I*).

For each packet which already has *aggr_id*:

1) The router checks whether the *aggr_id* read from the packet is in the AT (**J**).
2) If the route appears in the AT:
   - The router reads the *int_id* from the AT (**K**).
   - The router forwards the packet to the next node based on the *int_id* (**M**).
3) If the route does not appear in the AT:
   - The router determines the *int_id* (**N**).
   - The router adds an entry in the AT (**O**).
   - The router forwards the packet to the next node based on the *int_id* (**M**).

For each packet leaving the domain:

1) The router deletes the *aggr_id* from the IP header.
2) The router sends the packet outside the domain according to the operation of a traditional IP network.

FAMF ensures that the number of entries in the list maintained by core nodes is limited to the number of possible routes in the network, because different flows transmitting traffic along the same route are aggregated under one entry in the table. The number of entries in the table is significantly lower than the number of flows currently served by the router, which can be very high. Finally, the problem of scalability associated with handling of flows in FAMTAR is solved.

The FAMF mechanism is more complex than the pure FAMTAR. However, the level of complexity is only slightly higher comparing with FAMTAR and depends on type of a node. In edge routers more operations are necessary to conduct for the first packet of a flow. Besides adding a new registration to the FFT, we also need to calculate *aggr_id* and add it to the AT. On the other hand, for further packets we only analyze registrations in AT, which usually is less loaded than FFT. The computational savings in this area are even more significant in core routers, when difference between number of flows and aggregates is higher than in edge routers. Replacing FFTs with ATs, the operational complexity in core routers is reduced.

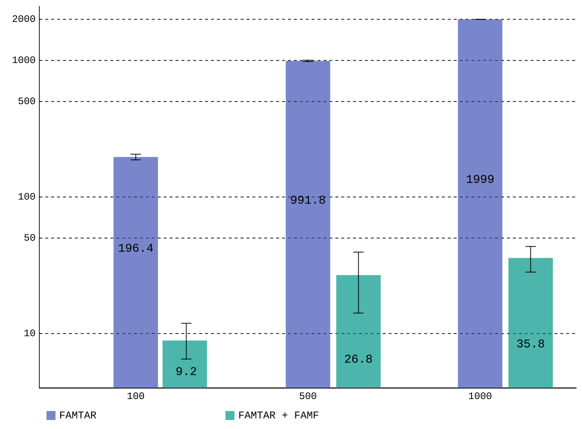

Fig. 3. Total number of flow table entries (logarithmic scale).

IV. Evaluation

In order to evaluate the proposed aggregation mechanism, we conducted network simulations using the *ns-3* simulator. We used the US-backbone topology (see Fig. 2), with 39 nodes. They were connected with 1 Mbps, 1 ms delay links. We chose such a low links bandwidth in order to speed up simulations. The obtained results are scalable. We generated traffic from group of 50 hosts connected to node *Portland* to group of 50 hosts connected to the node *Atlanta*. Each host was connected to its backbone node using 0.1 Mbps, 1 ms delay access link. There was no other traffic in the network.

The duration of simulation was 250 s of network operation. During that time, 100, 500 and 1000 uni-directional TCP flows were generated, between randomly selected hosts from sender and receiver groups. Flows start times were uniformly distributed over the simulation time. The average size of a flow was 1 MB, 200 kB and 100 kB, respectively for each scenario, modeled with Pareto distribution, with the shape factor equal to 1.5. Average total amount of data trasmitted in each scenario was equal to the number of flows multiplied by the average flow size. That means that in all scenarios, the average amount of transmitted data was constant and equal to 100 MB. All scenarios were repeated 10 times with warm-up period equal to 50 s. The results of simulations are shown in Fig. 3,

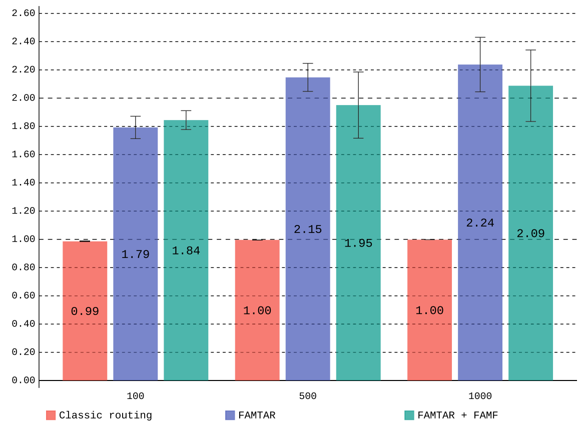

Fig. 4. Total throughput [Mbps].

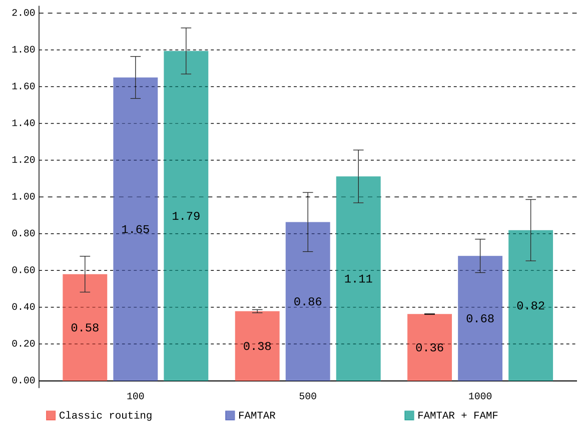

Fig. 5. Mean packet delay [s].

Fig. 4 and Fig. 5. Error bars represent 95% confidence intervals.

Total number of table entries is the number of forwarding table entries in *Portland* (the first core) router. When no aggregation scheme is utilized, this number is twice as large as the number of TCP generated flows, because reverse streams of acknowledgement (ACK) packets are counted as separate flows. With aggregation, the number of entries is significantly lower (95.4%, 97.3% and 98.2%, respectively for each scenario) and grows much slower than the number of flows. Furthermore, number of aggregate entries is bounded by network topology, whereas number of flows can grow infintely.

Throughput traffic is measured in the destination node. The throughput of 1 Mbps can be achieved with classic routing, because only one path can be used. In FAMTAR-enabled network, the throughput is higher. This is because FAMTAR utilizes multiple paths between the same nodes. The topology limits throughput between these two nodes to theoretical 3 Mbps (This is the value of max-flow/min-cut between these two nodes). When using FAMTAR with aggregation, the achieved throughput is more various, but within the confidence intervals with pure FAMTAR throughput. In fact, we observed high variance of throughput in both FAMTAR and FAMTAR with aggregation scenarios, as FAMTAR behavior highly depends of flows arrival pattern.

Mean packet delay was another measured quality parameter. In FAMTAR network, delay is significantly higher than in the classic routing network. We think that it can be attributed to higher queuing delay in access nodes, which is a result of higher overall traffic being served in FAMTAR. Aggregation increases delay slightly, however, errors bars show that the results for FAMTAR with aggregation are comparable with those for FAMTAR. The aggregation mechanism alone provides additional operations only for first packets of flows. Observing the whole transmission, impact of these operations is negligible.

Overall, results show that FAMF affects performance of FAMTAR insignificantly. At the same time, it allows massive reduction of the number of entries in forwarding tables.

V. Conclusion and Future Studies

Efficient flow aggregation is not an easy task, one that researchers and companies have been struggling with for a while now. Some universal ideas have been proposed but they usually do not work for particular cases. In this letter, we have proposed a flow aggregation mechanism designed specifically for FAMTAR. By aggregating flows, core routers do not need to maintain full flow tables which makes them less expensive to build and operate. Results show that the proposed aggregation provides similar overall network performance with significantly smaller flow tables.

In future, we plan to extend the presented flow aggregation method and increase its efficiency. In the current approach, presented in this letter, aggregates are formed only by the edge nodes. It would be beneficial if aggregates could be seen globally, i.e., the same aggregates were used for flows originating in different nodes if they share the same path.

References

[1] R. Wojcik, J. Domżał, and Z. Duliński, "Flow-aware multi-topology adaptive routing," *IEEE Commun. Lett.*, vol. 18, no. 9, pp. 1539–1542, Sep. 2014.
[2] J. Domżał *et al.*, "A survey on methods to provide multipath transmission in wired packet networks," *Comput. Netw.*, vol. 77, pp. 18–41, Feb. 2015.
[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 136–141, Feb. 2013.
[4] A. Mimidis, C. Caba, and J. Soler, "Dynamic aggregation of traffic flows in SDN: Applied to backhaul networks," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 136–140.
[5] S. Luo, H. Yu, and L. M. Li, "Fast incremental flow table aggregation in SDN," in *Proc. 23rd Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2014, pp. 1–8.
[6] X. Zhang, Z. Cheng, R. Lin, L. He, S. Yu, and H. Luo, "Local fast reroute with flow aggregation in software defined networks," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 785–788, Apr. 2017.
[7] N. Kamiyama *et al.*, "Flow aggregation for traffic engineering," in *Proc. IEEE GLOBECOM*, Dec. 2014, pp. 1936–1941.
[8] R. Wojcik, J. Domżał, Z. Duliński, P. Gawlowicz, and P. Jurkiewicz, "Loop resolution mechanism for flow-aware multi-topology adaptive routing," *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1339–1342, Aug. 2015.