Computer Communications 149 (2020) 300–311

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Testing implementation of FAMTAR: Adaptive multipath routing

Piotr Jurkiewicz *, Robert Wójcik, Jerzy Domżał, Andrzej Kamisiński

Department of Telecommunications, AGH University of Science and Technology, Kraków, Poland

ARTICLE INFO

Keywords:
Router
Multipath routing
Adaptive routing
SDN
Traffic engineering
Click modular router
Testing

ABSTRACT

Flow-Aware Multi-Topology Adaptive Routing (FAMTAR) is a new approach to multipath and adaptive routing in IP networks which enables automatic use of alternative paths when the primary one becomes congested. It provides more efficient network resource utilization and higher quality of transmission compared to standard IP routing. However, thus far it has only been evaluated through simulations. In this paper we share our experiences from building a real-time FAMTAR router and present results of its tests in a physical network. The results are in line with those obtained previously through simulations and they open the way to implementation of a production grade FAMTAR router.

1. Introduction

FAMTAR (Flow-Aware Multi-Topology Adaptive Routing) is a new approach to routing in IP networks, which provides multipath and adaptive capabilities. It is based on flows, notion currently incorporated in many architectures, including Software-Defined Networking (SDN). Unlike many SDN solutions, FAMTAR is a **fully distributed** mechanism and does not depend on a central entity. This is a key characteristic, as it significantly improves scalability and resilience.

Our aim was to test the efficiency of FAMTAR in real network conditions and validate the results obtained earlier through network simulations. In order to do that, we needed to build a FAMTAR router. In this paper we present our experiences in building a FAMTAR router. We also show the results of performance evaluation of networks built with FAMTAR routers. In particular:

- We explain how to build a FAMTAR router using the Click Modular Router [1] framework, provide its configuration and describe key components.
- We identify and solve several problems, which were not noticed during simulations:
  - The problem that a routing protocol may not be able to update routing tables on time after a link failure and flow entry deletion, which can result in subsequent packets being routed to a failed link.
  - The problem with overwhelming generation of ICMP Redirect messages.
- We show that, with FAMTAR, the amount of successfully transmitted data increases almost linearly with the number of available parallel paths (also, fewer packets are dropped in the network).
- We show that the average transmission delay in a FAMTAR-enabled network decreases with the increasing number of active parallel paths and is lower than in a standard IP network.
- These results are valid both for artificially generated traffic and real traffic from a BitTorrent network.
- We show that FAMTAR can effectively protect QoS parameters of VoIP flows and eliminate network congestions by redirecting excessive flows to alternative paths.
- We confirm that the TTL-based loop resolution mechanism works as expected, i.e., it resolves permanent loops which may appear in the network due to failures.
- Finally, we make the implementation publicly available as an open source software: https://github.com/piotrjurkiewicz/famtar

2. Related work

2.1. Multipath routing

There are numerous approaches to providing multipath transmissions in IP networks. Multipath solutions can also operate at different layers. The most visible solutions were surveyed in [2].

In the transport layer, Multipath TCP (MPTCP) is the most popular multipath solution. It is implemented in the Linux kernel, FreeBSD and Apple's iOS. It is crucial, because transport layer multipath requires support from both endpoints. MPTCP works by multiplexing a single TCP connection over multiple IP interfaces. Assuming that these interfaces provide disjoint paths to the other endpoint, MPTCP can increase throughput to the sum of available paths throughput. It also improves resilience, as any of the subflows can be disrupted, without losing the main TCP connection (seamless takeover).

* Corresponding author.
E-mail address: piotr.jurkiewicz@agh.edu.pl (P. Jurkiewicz).

https://doi.org/10.1016/j.comcom.2019.10.029
Received 2 March 2019; Received in revised form 17 August 2019; Accepted 23 October 2019
Available online 28 October 2019

As mentioned, MPTCP requires existence of multiple interfaces on both endpoints. It is most beneficial, when these interfaces are connected to different upstream providers (multihoming). This is an uncommon case, so usage of MPTCP is rather limited. It is used in mobile scenarios for applications requiring low latency (e.g. voice assistant), when both cellular and WiFi connections are available. Another use case is bulk data transfer between datacenters. QUIC protocol, which is envisioned as future HTTP transport protocol and already makes up significant amount of Internet traffic, was also designed to support multipath. Details of this feature were postponed to subsequent protocol versions (as of February 2019), but it will probably face the same limitations as MPTCP.

Network layer multipath solutions do not require any support from endpoints. They require, however, additional features on routers, usage of specific protocols in the network or manual configuration from a network operator.

Equal cost multi-path (ECMP) is a standard multipath load balancing technique. It is enabled by default in many devices and can be used with majority of routing protocols. It utilizes the fact that routing protocol may find multiple shortest paths (all with the same cost) to a given destination. Traffic is then distributed equally amongst multiple shortest path next hops in each router. While existence of multiple shortest paths is common in highly-symmetric datacenter networks, it is rare in wide-area networks. For example in the *nobel_eu* topology from SNDLib (http://sndlib.zib.de), average number of disjoint shortest paths is 1.20, whereas average number of disjoint paths (max-flow/min-cut) equals to 2.61. Lack of multiple shortest paths significantly reduces benefits from ECMP in wide-area networks.

Unequal cost multi-path (UCMP) assumes usage of additional paths with cost higher than the shortest one's cost. This is not trivial, as it may lead to routing loops. Loop-free UCMP requires specific metrics and constrains in routing protocol. The only protocol currently in usage supporting UCMP is EIGRP. Its conservative feasibility conditions (the DUAL algorithm), however, significantly limit the number of additional paths possible to use [3]. In order to use all available disjoint paths and achieve maximum flow between selected nodes, other techniques have to be used.

There are also many UCMP algorithms, which were proposed in academia, but have not been implemented in any routing protocol. This includes LFI-type algorithms: MPDA and MDVA, both proposed by the DUAL algorithm inventor [4], and OSPF extensions: OMP [5] and AMP [6], which are limited by a similar conservative feasibility conditions as DUAL.

2.2. Adaptive routing

Per-packet routing imposes significant limitations not only on multipath capabilities (due to routing loop prevention constrains, only a subset of existing alternative paths in the network can be used), but also on adaptivity. Adaptive (load-sensitive) routing is impossible in the per-packet approach, as the dynamic alteration of link costs leads to instability, which ultimately deteriorates network performance. This has been shown by early ARPANET attempts.

The original ARPANET routing algorithm used the current link delay as a metric [7]. It consisted of propagation and transmission times, which were constant, and variable queuing delay, which depended exponentially on the link usage. Its successor (*The New ARPANET Routing Algorithm*), which was a link-state routing protocol, used a 10-seconds average link delay as a metric [8]. This metric was called *Delay Shortest Path First (D-SPF)*. The new algorithm allowed to avoid permanent routing loops and was more stable. However, under high load, it still resulted in route oscillations. In July 1987 the metric was normalized in order to depend linearly on the link usage (*Hop-Normalized Shortest Path First (HN-SPF)*). This resulted in a more smooth handover of traffic from overloaded path, but still did not eliminate the route flapping [9,10].

These early ARPANET adaptive routing stability problems were analyzed in [11]. It was proven in [12] that in some cases adaptive routing can degrade the overall network performance even below the non-adaptive routing. Since that, approaches based on adaptive routing protocol metric changes were abandoned. However, some remnants of them are still present in the currently used routing protocols. For example, EIGRP can make use of link load in metric calculation, but this option is turned off by default [13].

Another approach was explored in an OSPF extension called OMP [5]. The authors proposed to change load-balancing weights instead of routing protocol metrics in response to overload events. They suggested to disseminate load information along with LSA messages. AMP [6] was a similar proposal. The difference was that the information was exchanged directly between neighbor routers using so-called *backpressure messages*. Both these extensions have not been implemented yet and their stability under a realistic traffic have not been evaluated.

In practice, multipath and adaptive routing is currently achieved with the help of Multiprotocol Label Switching (MPLS) [14]. MPLS allows operators to manually create paths (tunnels) and assign certain flows/transmissions to those paths. This approach is currently widely used, and it works. However, there are several drawbacks. Firstly, the operations are not automatic and require human intervention. Secondly, in large and complex networks the existence of multiple paths and many criteria, conditions, parameters, etc., creates havoc. Currently, many major operators have their MPLS nodes configured with such complexity that they are almost afraid to change anything, for fear of impairing the network's operation.

There are proposals of automated systems, which purpose is to modify MPLS tunnels in a response to the changing network load. Usually these systems assume usage of a centralized controller, which makes them incomparable with FAMTAR. However, distributed algorithms, like MATE [15] or TeXCP [16], were also proposed. They collect network load information using probe packets, but no implementations are available.

Non-MPLS based mechanisms were also proposed, for example REPLEX [17]. However, it is complicated and introduces additional signaling protocol to the network. Another similar solution is proposed in [18], but its drawback is that it makes routing decisions basing only on local interfaces load, instead of the whole network state. A short survey of adaptive routing approaches is presented in [19].

2.3. Flow-based routing

FAMTAR was developed to answer the aforementioned problems of per-packet routing by making the use of flow-based approach. It can overcome multipath and adaptivity limitations of per-packet routing by maintaining separate per-flow forwarding entries. As a result, flows between the same endpoints can follow any number of alternative paths without the risk of loops. Furthermore, paths for subsequent flows can be chosen with the current or a predicted network load in mind, effectively resulting not only in the multipath routing, but also in the adaptive routing. Such an approach should also ensure a better stability, as only new flows can be redirected to new paths during a congestion, which can reduce route flapping.

Caspian Networks and, later, Anagran tried to provide flow-based treatment. In [20], it is shown that keeping flow state information is feasible. Moreover, Anagran created FR-1000, a router which provided flow-based treatment and could be used for high speed links. Anagran stores packet forwarding information inside flow tables, but unlike FAMTAR, this is not modified according to network congestions. FAMTAR uses similar flow routing information to that used in Anagran, and combines it with routing adaptability to the current network congestion statuses.

A relatively new proposal for flow management was presented in [21]. In this solution, flows are classified and transmitted using multiple paths. A central manager decides which paths should be used for

each flow. This proposal looks promising, however, it is complex and difficult to implement. Moreover, the existence of the central manager may result in scalability and security problems in large networks.

Central management is also the tendency of currently sound Software-Defined Networks (SDN). There are hundreds of papers proposing SDN flow-based traffic engineering and routing systems. Some of these systems are fairly sophisticated, utilizing recent AI and machine learning advances [22]. All of them are (at least logically) centralized. Therefore, we purposefully do not review and compare them to FAMTAR, which is a fully distributed solution and thus belongs to a completely another class of algorithms.

2.4. FAMTAR origins and possible extensions

The development of FAMTAR was possible thanks to the advances in flow-aware traffic handling, such as Flow-Aware Networking (FAN) [23]. There were attempts to increase the efficiency of routing with the use of FAN, as presented, for example, in [24]. The technique of trunk reservation borrowed from the telephone network is proposed in this paper for route selection. It is assumed that a path for a flow is chosen based on the bandwidth it requires. Also, a simple intelligent routing for FAN is presented in [25], where it is assumed that only non-congested links are considered when forwarding packets. However, it is not specified how to inform all routers about the congested links. FAMTAR provides a solution to that problem. Moreover, FAMTAR is a general idea, not specific to FAN networks.

The basic implementation of FAMTAR can be extended with other existing solutions, like admission [26] and congestion control approaches or source-based routing. Here we present some examples of mechanisms and algorithms which can be considered for future research.

In the paper [27] a delay response BBR (Bottleneck Bandwidth and Round-trip time) algorithm designated for real time video transmission is proposed and analyzed. The main assumption of the proposed algorithm is to reduce sending rate when the link delay exceeds predefined threshold. The goal is to allow routers to maximize the usage of bandwidth by control of buffer occupation. The sending rates of flows are being observed and actively reduced when delay exceeds a predefined threshold. This allows a router to drain queued buffers and, as a consequence, to reduce packet delay and loss rate. The authors proved by simulation experiments that the proposed congestion control algorithm achieves lower frame delivery delay in comparison to benchmark algorithms. The algorithm can be implemented in any multipath architecture to enable maximum bandwidth utilization and to ensure quality of service at the assumed level for selected traffic. This proposal can also be implemented in FAMTAR, which originally has been proposed to serve traffic in a best effort regime.

An approach similar to the described above has been proposed in [28]. The authors present a model which combines a Network Utility Maximization to control rate of flows taking into account end-to-end queuing delays. However, in this solution, a Markovian Traffic Equilibrium is used to decide on routing based on total expected delays. The authors explain that the proposed method uses a routing strategy which is based on a decentralized stochastic version of Wardrop's model. Provided theoretical analysis confirm that it is possible to establish decentralized multipath routing algorithm which control congestions. The presented solution has been described only theoretically. The authors did not show simulation or tests of hardware implementation results. The proposed solution is more complex than FAMTAR. It needs additional signaling to distribute information about queuing delays in each outgoing link of all network nodes. One of the main advantages of FAMTAR is lack of any additional signalization in a network.

Another congestion control approach for multipath implementation has been proposed in [29]. The authors present a rate-based, Multipath-aware Information-centric networking Rate-based Congestion Control (MIRCC) approach. The proposed solution is inspired by the Rate Control Protocol (RCP). MIRCC ensures an acceptable convergence time with less overshoot and oscillation in comparison to the RCP and a multipath transmission along all the available paths, maintaining fairness among active competing flows regardless of number of paths that each flow has.

The MIRCC approach requires an implementation of several algorithms and protocols, e.g. algorithm in each forwarder calculating dual-class rates for each link, protocol mechanisms to communicate rates and path identifiers to consumers in data messages, an algorithm to determine interest sending rates for each class and to determine a sensible distribution of interests across available paths. As one can see, the proposed mechanism is complex and has been proposed for ICN (Information-Centric Networking). Thus the implementation of the described approach in FAMTAR is outside the scope of this paper.

Recently proposed by IETF, RFC 8354 [30], presents use cases for IPv6 Source Packet Routing in Networking (SPRING). SPRING is a networking architecture which leverages the source routing paradigm. The main assumption is that an ingress node steers a packet by including a controlled set of instructions, called segments, in the SPRING header. The mentioned architecture can be used in e.g. small offices, access networks or data-center networks. Of course, it can also be implemented in core networks. The authors list that it can enable to e.g.:

- use selected high-bandwidth links for a specific type (high priority) of traffic and thus avoid the need for overdimensioning the links in the network,
- setup separate path for delay-sensitive flows,
- reach important servers through source-based optimal path,
- use of disjoint paths.

Source-routing approaches are promising solutions for multipath routing of flows. They, however, need additional signaling comparing with FAMTAR. The architecture like SPRING can be implemented in network with FAMTAR. Both architectures can complement each other. However, at this moment, we focus on basic FAMTAR implementation to show its effectiveness and simplicity.

3. Flow-aware multi-topology adaptive routing

FAMTAR was introduced in [31]. It is a multipath adaptive routing mechanism that works based on flows [32]. FAMTAR is placed above the IGP (it does not interfere with the routing protocol operation) and can work with every protocol. A routing protocol is responsible for finding the best path between two endpoints up to its capabilities. In an uncongested network, all transmissions between those endpoints use this path. When a path becomes congested, all new flows are pushed to an alternative path, while flows which are already active remain on their primary path. Therefore, FAMTAR uses the best path provided by the routing protocol, and automatically triggers finding new paths in case of congestion.

To achieve that, every FAMTAR router maintains a Flow Forwarding Table (FFT) alongside the classic routing table. FFT is an associative array in which the keys are flow identifying fields. For each flow, the corresponding FFT entry indicates the interface to which packets of this flow are 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. Entries in the FFT are static and do not change alongside routing table changes. FFT is used to realize most packet routing tasks, as for flows that are present in the FFT, the routing table is not consulted.

When a state close to congestion is noticed on one of the links, the corresponding router sets the cost of this link to a predefined high value. From this moment, this link is perceived as congested. The new cost appears as a change in the routing protocol, which disseminates this information as a standard topology change message. Upon receiving this information, routing protocols compute new paths

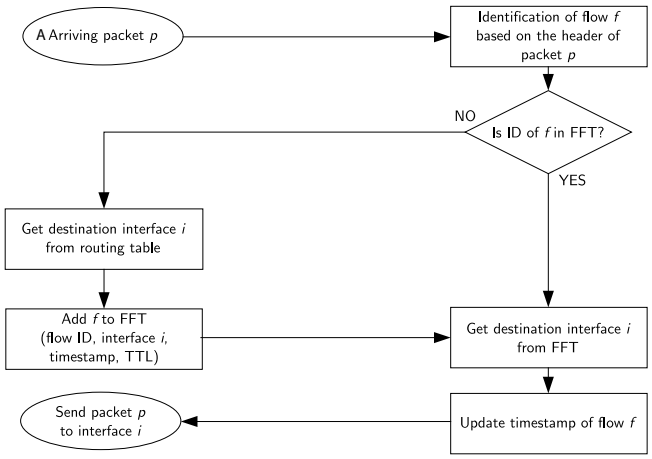

Fig. 1. Block diagram of FAMTAR algorithm.

which are likely to avoid congested links. The newly computed paths cause the related changes in the routing tables. However, these changes affect only new flows. The old flows, which were active before that event, are still routed on their existing paths stored in the FFT entries. Although the congested link still forwards all the flows which were active before the congestion was noticed, new transmissions do not appear. After a while, when the congestion on this link stops, the original cost of the link is restored. Note that FAMTAR requires a router to detect congestion on one of its links. The method to determine the congestion is not specified, although any congestion indicator can be used (e.g., link load, queue occupancy, packet queuing delay, and so on). The block diagram of FAMTAR algorithm is presented in Fig. 1.

The key idea of FAMTAR is keeping the system distributed by leveraging the IGP to perform route computations instead of communicating with a central controller. Moreover, unlike traditional best-effort approaches, which usually can utilize only multiple shortest paths (ECMP), FAMTAR can use all available paths between transmitting nodes. The Table 1 presents statistics of three real WAN topologies from the SNDLib database (http://sndlib.zib.de). The number of disjoint paths between nodes is 2–3 times greater than the number of disjoint shortest paths. Usage of these additional paths is impossible in ECMP. Unequal cost multipath algorithms (UCMP), such as EIGRP, allow to use some of these additional paths, but their set is still very limited by a conservative feasibility condition. FAMTAR's flow-based approach makes it possible to overcome these constrains and utilize all existing paths. This increases the efficiency of network resource utilization and reduces the need for over-provisioning, since it is possible to use all already available resources.

Another key factor, which distinguishes FAMTAR from the other multipath approaches, is the usage of alternative (and potentially sub-optimal) paths only when necessary and only to new transmissions. A path with a higher cost is used only when paths with lowest costs are congested, whereas EIGRP uses all paths regardless of their load, which means that some packets are routed on worse paths even when there is no congestion on the best path. In FAMTAR, upon congestion, only new flows use alternative paths, whereas the old ones remain on their primary paths.

Flow-based approach also improves stability and reduces route flapping, as only the new flows are forwarded to new paths. Traditional adaptive routing approaches (such as ARPANET routing algorithms) had stability issues, because when the original path was becoming congested, the whole traffic was switched to the alternative path, which resulted in oscillations between these two paths. Simulation results presented in [33] show that FAMTAR is able to significantly increase the amount of traffic sent in a network, while reducing packet delays.

Table 1
Statistics of selected SNDLib real WAN topologies.

Topology name	polska	nobel_eu	germany50
Number of nodes	12	28	50
Number of edges	18	41	88
Graph density	0.27	0.11	0.07
Avg. vertex degree	3.00	2.93	3.52
Avg. num. of disjoint shortest paths	1.27	1.20	1.23
Avg. num. of all disjoint paths	2.67	2.61	3.19

4. Implementation environment

To implement FAMTAR, we had to find a router environment which would allow modifications in packet processing. This environment had to be expanded by introducing FFT and all actions related to using this table. Therefore, the implementation environment for an experimental FAMTAR router must allow easy modifications and debugging of the packet processing path. This requirement led us to choose the Click Modular Router suite (Click) as a data plane provider.

Click is a suite for building flexible software packet processors, designed with research and experimental applications in mind. It was developed by Kohler [34] in his Ph.D. dissertation. Click is widely used for building experimental software routers and switches. Its advantages include considerable flexibility, clear and scalable architecture, ease of adding new features, and high performance.

Click achieves flexibility due to its modular and object-oriented architecture. Routers are assembled from fine-grained packet processing modules called elements, which are C++ classes. Each individual element performs a simple operation on a packet, like queuing or decrementing a packet's time to live (TTL) field. Each element has input and output ports, which serve as the endpoints of connections between them. A user builds a complete router configuration by connecting individual elements into a directed graph. During router operation, packets are processed sequentially by individual elements.

Click can run as a user-level application or as a Linux kernel module. In the kernel mode, the Click module replaces the operating system (OS) networking stack, and packet processing is done only by Click. In the user-level mode, Click uses the system to receive packets.

Choosing Click as the data plane platform determined another router components including the routing daemon. Since Click does not implement dynamic routing protocols, its routing table must be populated by an external routing daemon. The only daemon that cooperates with Click is XORP (eXtensible Open Router Platform) [35]. XORP is an open source IP routing software suite originally designed at the International Computer Science Institute in Berkeley, California. It supports various routing protocols, including OSPF, BGP and RIP. Click combined with XORP provide all the required functions that each router supports, and this was the point in which our FAMTAR implementation started.

5. Router components

Fig. 2 presents the FAMTAR prototype router build on top of Linux Debian. The system runs Click and XORP with our extensions as well as auxiliary scripts written in Python. Click is responsible for forwarding packets and realizing FFT functions. Its routing table is updated by XORP which runs an OSPF protocol. Any link cost change in a network triggers XORP functions which results in routing table updates in Click.

The network monitor collects load information from physical interfaces. The monitor changes link costs in XORP for links whose traffic load exceeds the congestion threshold. The analysis of link loads is performed only a few times per second. Therefore, the implementation of this mechanism does not need to be extremely effective, so we implemented it in an external script.

The FAMTAR router configurator, which is a graphical front-end for configuration of the XORP daemon and the monitor, allows configuring

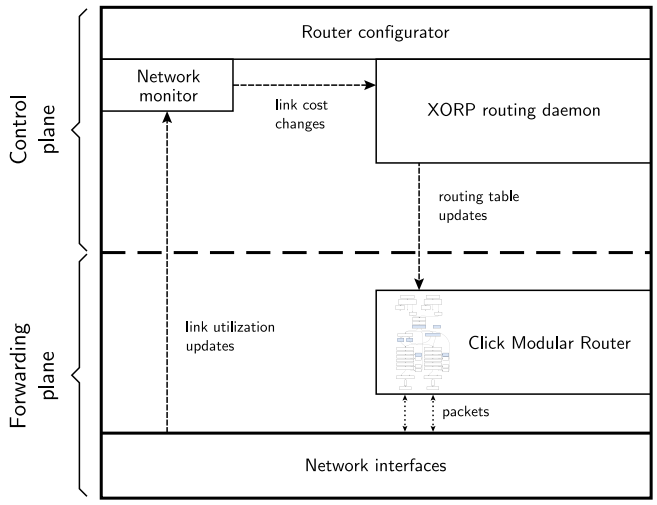

Fig. 2. FAMTAR router architecture on the Linux OS.

the IP addresses of interfaces, congestion thresholds, and other router parameters.

The most important element, however, is the extension of Click. To implement FAMTAR in Click, it was necessary to add a few additional elements to the standard IP router. The block diagram of the FAMTAR router in Click is presented in Fig. 3. This graph is an extension of the standard IP router graph, presented in [1]. The blocks related to FAMTAR are marked in blue shading. New elements: *CheckFFT*, *AddFFT*, *RouteFFT*, and auxiliary element *FFT* are not part of the Click library. They are written by us and grouped in the Click package `famtar` [36]. This package may be downloaded and imported to Click as a shared library without Click modification or recompilation.

The first FAMTAR-related element is *CheckFFT*. It is placed before the *LookupIPRoute* block, where routing is executed. The aim of the *CheckFFT* element is to check whether the FFT contains an entry for the flow related to the incoming packet. If the FFT does not contain the flow entry (i.e., it is a new flow), the incoming packet is sent to the first outgoing port of the *CheckFFT* element. Next, this packet is served in the *LookupIPRoute* element and sent to the correct outgoing port based on the current routing table. After this operation, the new entry for the particular flow is added to the FFT in the *AddFFT* element. The entry includes the incoming time of the packet, its TTL value, the identifier of the outgoing port and the IP address of next router on the path.

When the active flow entry exists, the timestamp in this flow is updated and the incoming packet is sent to the second outgoing port of the *CheckFFT* element. Then, the packet is processed by the *RouteFFT* element. This element operates similarly to *LookupIPRoute* and is responsible for packet routing. However, the packet is directed based on the outgoing port number stored in the FFT entry, instead of the one from the current routing table.

Additional operation is executed in the *AddFFT* element, when a failure occurs in the network. When a router detects lack of carrier on the outgoing link, the *AddFFT* element blocks for a fixed period (we assumed 5 s) a possibility to add new flows for that particular link. This operation is necessary for the correct implementation of the loop-resolution mechanism proposed in [37]. This mechanism assumes that after a link failure, all flow entries for this link are deleted from FFT. However, it takes some time to calculate new paths by the routing protocol. In this period, new flows outgoing via the recently-failed interface cannot be accepted to FFT. This means that during this period, these flows are being routed using solely the current routing table.

Moreover, it was necessary to switch off the mechanism implemented in the standard IP router, which sends error ICMP Redirect messages to the source node when a packet is rerouted to the same interface it arrived from. This mechanism generates traffic which is not necessary in FAMTAR network, because possible loops are solved by the TTL-based mechanism. To switch off this mechanism in Click, we changed the *ICMPError* element with *Discard* element presented in Fig. 3.

All described elements: *CheckFFT*, *AddFFT* and *RouteFFT* operate on the same FFT, which is implemented in the auxiliary element known as *FFT*. This element is not placed directly on the packet processing path. However, it offers the other elements some functions to modify the FFT. The structure of key for FFT is as follows:

```
struct FlowKey
{
   uint32_t srcaddr;
   uint32_t dstaddr;
   uint16_t srcport;
   uint16_t dstport;
   uint8_t  ip_prot;
}
```

FFT stores time of last packet of a flow, routing information and TTL value of the first packet of the flow. The structure of values stored in the flow forwarding table is presented below:

```
struct FlowValue
{
   uint32_t ts;
   uint8_t port;
   IPAddress gateway;
   uint8_t ttl;
}
```

Each flow key contains 104 bits (13 bytes) in which flow identification fields are stored: IPv4 source and destination addresses ($2 \cdot 32$ bits), source and destination transport layer port numbers ($2 \cdot 16$ bits), and transport layer protocol number (8 bits). Each flow entry contains the following information: the time of last packet (32 bits), forwarding interface (8 bits), IP address of the next hop router (32 bits), and the TTL value of the first packet of the flow (8 bits). The timestamp is used to determine whether the particular flow entry expired or not. If no packets arrive for a predefined period of time, the flow entry is deleted. The number of the outgoing interface and the IP address of the next router on the path are necessary to perform routing in the *RouteFFT* element. The TTL value is required by the loop resolution mechanism. The total size of a flow entry is 80 bits (10 bytes). The total amount of information required to store each flow is therefore 104 bits for a key plus 80 bits for data, which gives 184 bits or 23 bytes. This means that to process 1 million simultaneous flows, a router needs 23 MB of memory for FFT, which is acceptable.

The `HashTable<>` container from the standard Click library is used to implement this associative array as a hash chain array. This make it possible to ensure low computing complexity – $O(1)$ – for operations made on packets. When a flow finishes transmission, its identifier should be removed from FFT. While routers are not aware explicitly when flows finish transmission, we had to implement an additional mechanism — garbage collection. The content of buckets is analyzed when new flow entries are added to them and these entries, whose timestamp is older than a predefined timeout, are removed from FFT

6. Tests of the FAMTAR router

So far, the functionality of FAMTAR has been tested and the results have been presented only basing on simulation experiments conducted in the ns-3 simulator. They showed advantages of the FAMTAR routing over the standard IP routing. However, we have to note that the simulation analysis is usually provided with limitations. For example, not all factors from real networks can be taken into account in simulations.

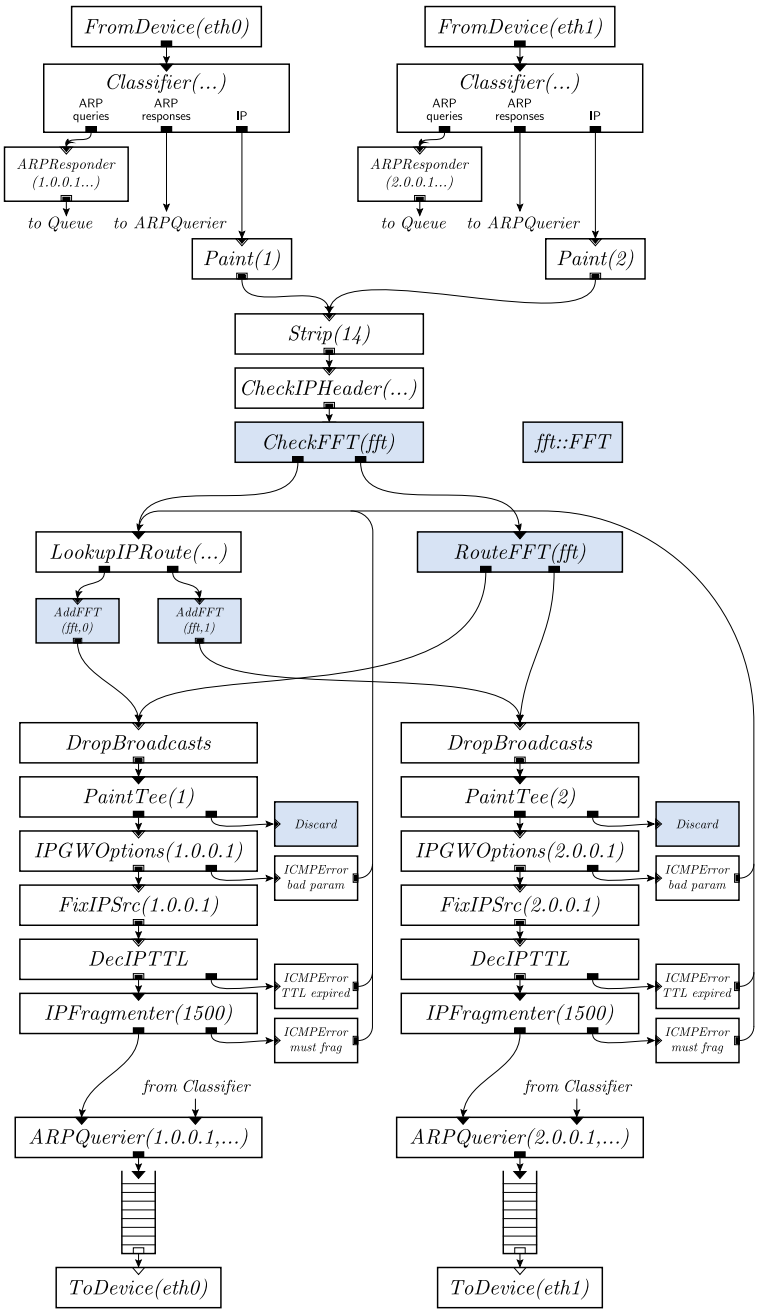

Fig. 3. Block diagram of FAMTAR implementation in Click Modular Router. Blocks marked in blue are related to FAMTAR and were added to Click. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

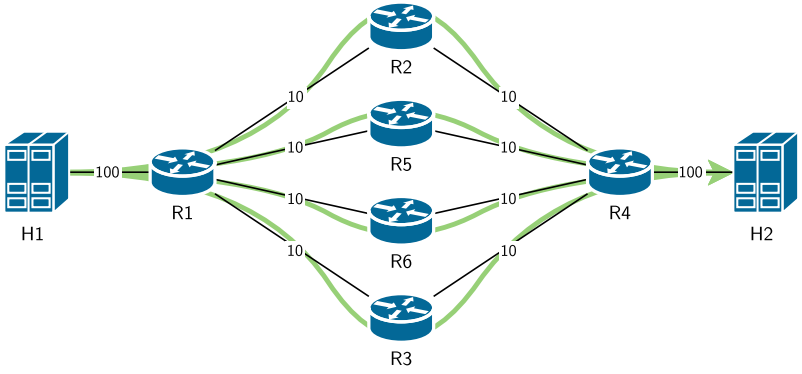

Fig. 4. Scenario I — Testing topology.

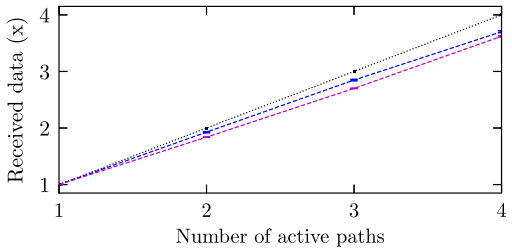

Fig. 5. Scenario I — Relative improvement in the amount of received data (blue line — Click implementation, magenta line — ns-2 simulation [33], black dotted line — linear scaling (for reference)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The most valuable results can be obtained when a router is tested in a network where real users generate traffic. We can assume that a device which passes such tests works properly and is scalable. Unfortunately, it is difficult and risky to test a prototype in a production network. That is why we conducted our tests in a laboratory environment, using either real of artificially generated traffic. We prepared a small network, as presented in Figs. 4, 7, 10, 11 and 13. All links in our network were bidirectional with capacity equal to 10 Mbit/s. Only links between border routers and hosts had 100 Mbit/s capacity. We have chosen such a low capacities in order to make sure that results will not be influenced by insufficient computing power of PCs we were using as routers.

Congestion thresholds were set to $Th_{min} = 0.7$ and $Th_{max} = 0.9$, as previous simulation papers showed that these values provide the best compromise between stability and performance. Link usage counters were refreshed every 200 ms and the exponential moving average with $\alpha = 0.2$ was used to smooth out the readings. Moreover, frequency of route table recalculations was also limited by OSPF hold timer (similar to BGP's route dampening), which in the case of XORP daemon used by us was equal to 1 s. The default routing protocol cost of all links was equal to 1 (with the exception of scenario II). When a congestion on a link was detected, its cost was being raised to 100.

Traffic was generated in H1, which was a PC computer. We used the D-ITG [38] application to generate traffic and to collect statistical data. In scenario II and III, we did not use the D-ITG traffic generator. Instead, in scenario II, an UDP packet generator was used in order to achieve realistic flow length and size distributions. In scenario III, R1 was connected to the Internet and we observed traffic in the network in the case when the destination host was downloading a file from the Internet through the P2P protocol. We repeated each experiment five times to collect statistically credible results. In most scenarios we observed four cases, with one, two, three and four possible paths between edge routers R1 and R4. Each time, we compared the results obtained for routers with turned on FAMTAR mechanism with the results obtained for routers doing classic IP routing.

The following sections present the ideas and results from four scenarios that we analyzed.

Scenario I

To verify the performance of FAMTAR with regard to its multipath capabilities, 500 UDP flows were transmitted between nodes H1 and H2 (Fig. 4). Packet size was set to 1000 B, whereas the flow size was selected according to the Pareto distribution (average: 1 MB, shape: 1.25). Each flow was transmitting data at a constant rate of 100 kB/s and the time between flow starts was given by the exponential distribution, with the average value of 0.5 s. We collected the results between the 20th and 230th seconds of each experiment.

We conducted this experiment in networks with 1, 2, 3, and 4 parallel paths available. We assessed the performance basing on the following metrics:

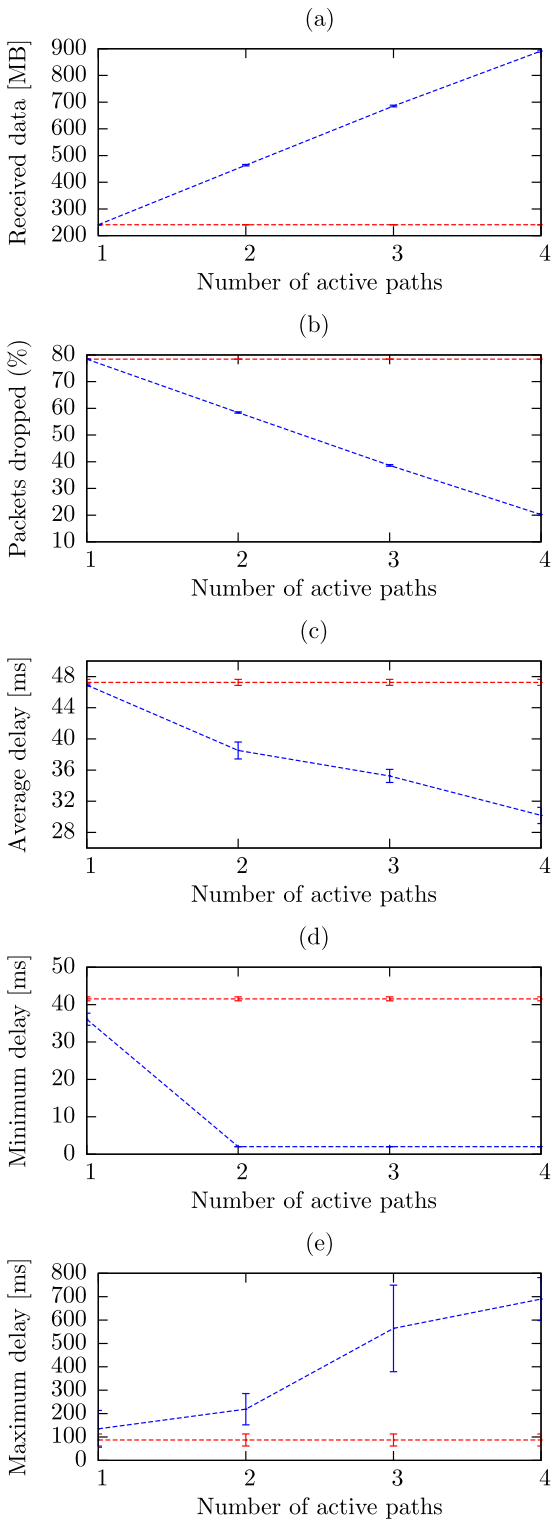

Fig. 6. Scenario I — Relations between the number of active paths and (a) the amount of received data, (b) the percentage of packets dropped, (c) the average transmission delay, (d) the minimum transmission delay, and (e) the maximum transmission delay in the standard IP and FAMTAR-enabled networks (red and blue lines, respectively). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- number of bytes received at node H2,
- dropped packets ratio,
- average packet delay,

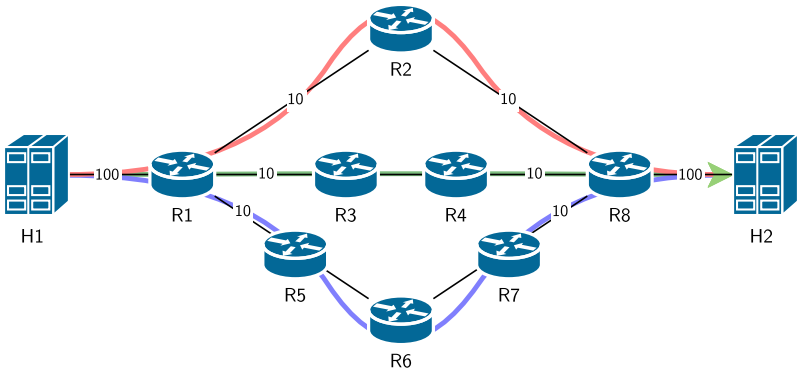

Fig. 7. Scenario II — Testing topology.

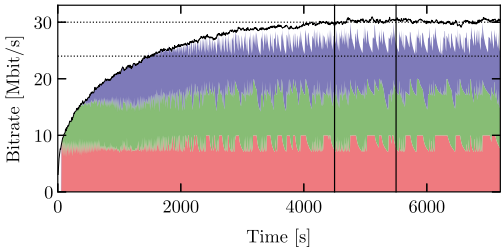

Fig. 8. Scenario II — Stackplot of UCMP links utilization. The black solid line shows the generated traffic. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

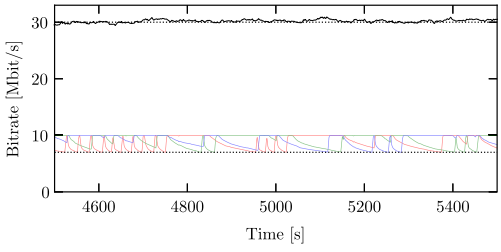

Fig. 9. Scenario II — Link loads between 4500 and 5500 s of the experiment, plotted as separate lines. The black solid line shows the generated load.

- minimum packet delay,
- maximum packet delay.

The results are presented in Fig. 6. In the case when only one path is available, the results for the FAMTAR-enabled network are similar to the results for the standard IP network. This is in line with our expectations, since if no additional paths are available, FAMTAR cannot provide multipath transmissions and there is no gain.

In the case of a standard IP network, the results do not change with increasing number of additional paths. At the same time, in the case of the FAMTAR-enabled network, we note that the amount of successfully transmitted data increases linearly with the number of parallel paths. In addition, fewer packets are dropped in the network. This observation confirms one of the most significant advantages of FAMTAR, which is the ability to efficiently provide parallel multipath transmissions.

We also note that the average transmission delay in a FAMTAR-enabled network decreases with the increasing number of active parallel paths and is lower than in the case of a standard IP network.

Additionally, we compare our results with the simulation results presented in [33]. The comparison is shown in Fig. 5. Because the traffic workload used in simulations was different than one in our tests, we compare a relative improvement rather than the absolute amount of received data. The dotted black line shows an ideal linear-scaling. It can be seen, that our FAMTAR implementation achieves even a slightly higher multipath throughput improvement than the simulation, but still slightly below the ideal linear scaling.

Scenario II

This scenario was focused on the verification of FAMTAR under a realistic traffic. The key factor affecting FAMTAR's performance and stability is flow length and size distribution. In the edge case, when all flows in the network are single-packet long, flow caching in FFT becomes meaningless and FAMTAR degrades to a traditional adaptive routing known from ARPANET (which had stability issues).

In order to investigate FAMTAR performance and stability under realistic traffic, we used an UDP flow-based packet generator. The traffic was generated using realistic flow length and size distribution models presented in [39]. These are the most accurate models currently available in the literature. The flow interarrival rate was set to match 30 Mbit/s of generated traffic in the steady state.

Moreover, to verify FAMTAR's UCMP capabilities, in this scenario each of three available paths has a different length/cost (Fig. 7). In such a scenario, traditional ECMP approaches would be able to use only a single path (the shortest one through the R2). FAMTAR should be able to use all the available paths, starting from the shortest one.

The Fig. 8 contains a stackplot of bitrates of all three paths. Red, green and blue plots correspond to one-, two- and three-hops paths respectively. The black solid line presents the total bitrate of generated traffic.

First of all, it can be seen that it takes more than one hour to reach the steady state by the generator. This is due to the heavy-tailed nature of IP flows. The total throughput of all three paths confirms FAMTAR's UCMP capabilities. FAMTAR can load-balance traffic on the all available paths, even when costs of these paths are different. Moreover, it can be seen that paths with higher cost are started to being used only when all paths with lower costs are congested.

In terms of stability, in the steady state FAMTAR's total throughput varies between 24 and 30 Mbit/s (the dotted horizontal lines on the plot). This is in line with the theory, as 24 Mbit/s corresponds to 0.8 of all links throughput and 0.8 is the midpoint value between used congestion thresholds. Simulations presented in previous papers showed a similar results.

In Fig. 9 we present a zoomed-in part of Fig. 8 (between 4500 and 5500 s). In this case plots are not stacked, but presented as separate lines. It can be seen, that traffic oscillations in FAMTAR are significantly reduced thanks to the usage of FFT, which caches routes for existing flows. When congestion occurs, only new flows are routed to the new path, whereas the old ones remain on their existing paths. This introduces an inertia, which reduces both oscillations frequency and their depth. It can be seen that during this 1000-second period, only 25 routing changes happened. The depth of oscillations was limited to 0.7 of each link throughput, which is the value of Th_{min}.

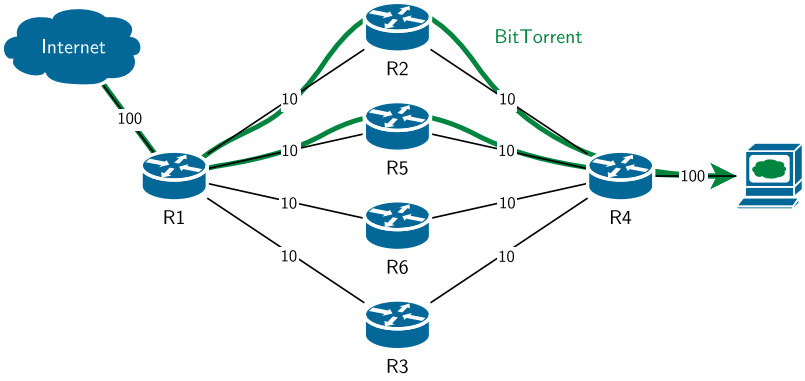

Fig. 10. Scenario III — Testing topology.

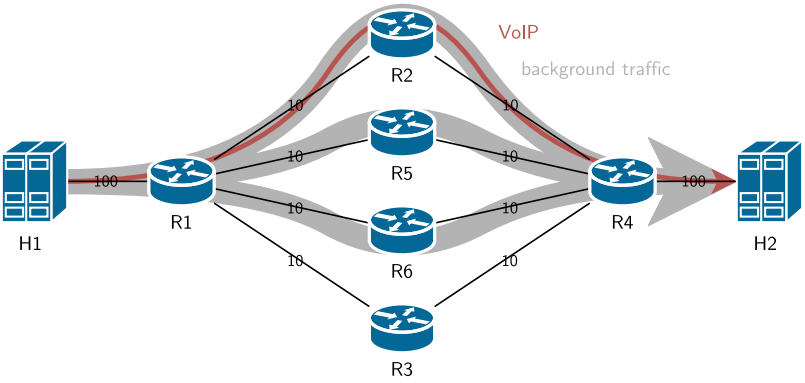

Fig. 11. Scenario IV — Testing topology.

Table 2
Scenario III — BitTorrent download statistics of the file (Ubuntu ISO CD image, 1 028 653 056 B).

	Without FAMTAR	With FAMTAR	
		2 active paths	4 active paths
Average download time [s]	944.3 ± 8.72	492.7 ± 15.18	313.0 ± 22.36
Average download rate [Mbit/s]	8.7 ± 0.08	16.7 ± 0.52	26.3 ± 1.88
Relative gain in throughput	0%	92%	202%

Scenario III

The goal of the second scenario was to verify the capability of FAMTAR to provide multipath transmission in a real computer network. A large file (Ubuntu ISO CD image, 1 GiB) was downloaded through the network using the BitTorrent protocol. During the experiment, router R1 was connected to the Internet, whereas host H2 was downloading the file from external peers (see Fig. 10). Connections with peers were established using the TCP and UDP protocols.

We compared a standard IP network to a FAMTAR-enabled network with 2 and 4 parallel paths. The performance was evaluated based on the following metrics:

- average download time,
- average download rate,
- the relative gain in throughput.

The results are presented in Table 2. According to the results, the FAMTAR-enabled network was able to provide much higher transmission speeds (thus shorter download time) than the standard IP network in all considered cases. This means that the FAMTAR-enabled network performed better than the standard IP network with respect to all three considered metrics.

Scenario IV

The goal of this scenario was to verify the capability of FAMTAR to provide a load-adaptive routing. One important issue related to the presence of congestions in a network is the effect of background traffic on the Quality of Service (QoS) experienced by certain flows, such as for example related to the Voice-over-IP (VoIP) service. To evaluate the performance of FAMTAR in this context, we started one VoIP flow (G.711.1 RTP) between nodes H1 and H2 (Fig. 11), and then the following background flows were scheduled and launched one after another every 200 ms to gradually consume the available resources on the VoIP flow path:

- 50 UDP flows (100 kbit/s, beginning at the 6th second),
- 150 UDP flows (100 kbit/s, beginning at the 25th second).

The background flows from the second group were terminated in the same order, with a 200 ms interval, starting at the 70th second. The remaining flows were active until the end of the measurement period. The total amount of generated background traffic during the experiment is presented in Fig. 12(d).

Fig. 12 show three different QoS-related metrics for the VoIP flow during its activity. We can see that in the case of a standard IP network, packet losses occur in the VoIP flow, starting from the 30th second when the background traffic begins to exceed 10 Mbit/s. This observation is in line with our expectations, since the classic IP routing can utilize only one 10 Mbit/s link. When the offered traffic starts to exceed this level, a portion of packets needs to be dropped in the router's output queue.

In the FAMTAR-enabled network, the VoIP flow remained almost unaffected by the background traffic. The results have proven that FAMTAR eliminates network congestions by redirecting the excessive flows (which would otherwise overload the link) to alternative paths.

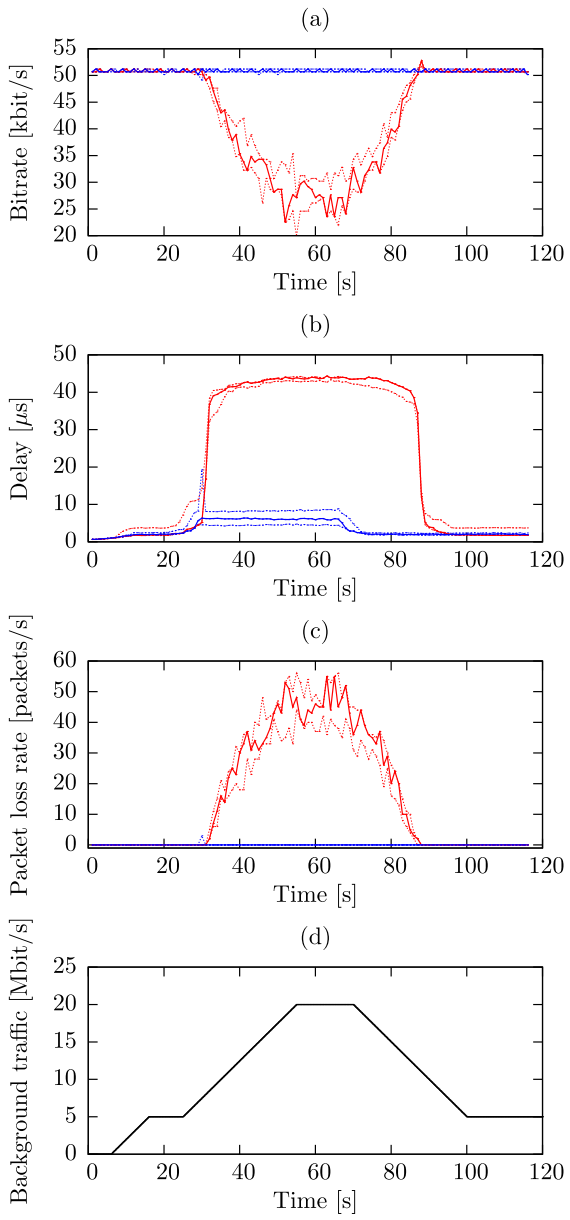

Fig. 12. Scenario IV — (a) Bitrate, (b) delay, and (c) packet loss rate of the VoIP flow in the congested standard IP and FAMTAR-enabled networks (red and blue solid lines, respectively). The dotted lines reflect the minimum and maximum values acquired in all trials, whereas the solid lines correspond to a single trial. The amount of generated background traffic during the experiment is shown in (d). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the standard IP network, the VoIP stream was mixed with the background traffic forwarded along the same path, and suffered from increased delay, decreased bitrate, and high packet loss rate. We can see it in Fig. 12(b). Until the 30th second, the delay in the FAMTAR-enabled network rised in a similar way as in the standard IP network. This is because the active link was being gradually loaded with the background traffic, so the VoIP packets had to statistically wait longer in the router's queue. However, in the 30th second, delay stopped increasing in a FAMTAR network. This was caused by the FAMTAR adaptive mechanism which started to use an alternative path for new flows in order to avoid congestion. Consequently, the observed delay remained constant. It started to decrease from the 70th second when the background flows assigned to the first path started to terminate (see Table 3).

Table 3
Scenario IV — Effect of traffic congestions on a VoIP flow.

	Without FAMTAR	With FAMTAR
Min. bitrate [kbit/s]	23.04 ± 2.95	49.97 ± 0.57
Max. delay [μs]	44 ± 0.50	9 ± 7.10
Max. packet loss rate [pkt/s]	54.0 ± 3.62	0.6 ± 1.67

Table 4
Scenario V — Number of packets dropped before the traffic was moved on a new path.

Without FAMTAR	With FAMTAR	Avg. difference	Relative gain
3841.9 ± 631.5	4656.2 ± 48.1	−814.3	−21.2%

Scenario V

Transient loops can occur in routing protocols during link cost updates. In traditional approach, such loops are resolved as soon as the state of the routing protocol tables converges on all routers. However, in FAMTAR, routes are being frozen in the FFT when the first packet of a flow passes through the network, which can result in such a transient loop becoming permanent. Such transient loops are resolved immediately by the mechanism presented in [37].

The same mechanism can also resolve permanent loops, which can occur due to link failures. Such loops are, however, not being resolved immediately. Thus, the goal of the last experiment was to verify the performance and delay of the loop resolution mechanism during a real failure.

In this scenario, only one flow (constant bitrate of 2.84 Mbit/s, 64 B packets) was transmitted via a single path between nodes H1 and H2 (Fig. 13). This corresponds to approximately 55k packets per second. This means that the amount of traffic did not trigger the multipath mechanism of FAMTAR. We selected a relatively low bitrate to ensure that the results could be compared to the standard IP network.

A failure of the active link was simulated in the network by physically unplugging the network cable from the interface of router R2. It was expected that the network will move the existing flow on a new path. We compared the number of dropped packets during the restoration phase for both the standard IP and FAMTAR-enabled networks. Traffic generation and measurements were done using the D-ITG software tool. We summarized the results in Table 4.

Based on the results, we admit that the deployment of FAMTAR may increase the number of dropped packets during the path restoration phase, which also means that the total restoration time may be longer than in the case of a standard IP network. However, the difference is not large. Having in mind that the generated traffic was 55 kpps, this corresponds to 15 ms of additional outage comparing to classic OSPF routing. Thus, we believe that FAMTAR still demonstrates reasonable performance in the presence of failures. Moreover, the experiment has confirmed that the TTL-based loop resolution mechanism works as expected, i.e., it resolves permanent loops which may appear in the network due to failures.

7. Conclusion

FAMTAR is a promising concept for multipath routing in IP networks. So far, the approach was presented and evaluated through simulations. Now, we have a prototype which we tested in physical networks.

We presented the practical aspects of a prototype implementation. We also documented and solved challenges and problems encountered during the implementation process, which were not noticed during simulations. After basic tests verifying the proper operation of each component, we benchmarked physical networks built with the FAMTAR routers under real traffic conditions. The results show significant advantages of FAMTAR compared to standard IP routing. Most importantly, the test results are in line with simulation analysis published earlier.

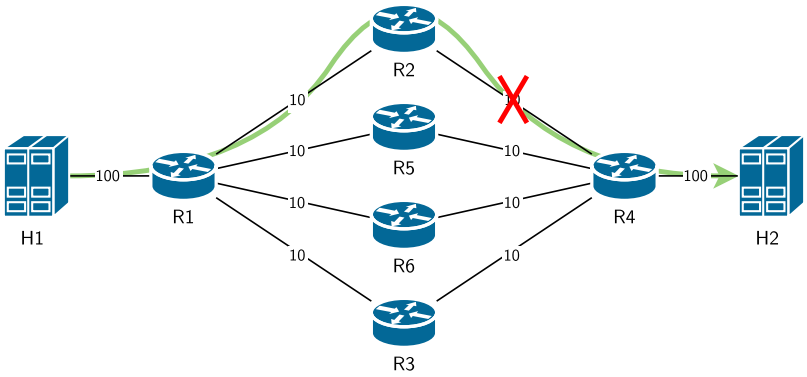

Fig. 13. Scenario V — Testing topology.

We believe that the implementation and test results presented in this paper will accelerate further development of FAMTAR. Now, FAMTAR is a robust solution which is also easy to implement.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The research was carried out with the support of the project "Flow-Aware Multi-Topology Adaptive Routing" funded by the National Centre for Research and Development in Poland under the LIDER programme.

References

[1] E. Kohler, R. Morris, B. Chen, J. Jannotti, M.F. Kaashoek, The click modular router, ACM Trans. Comput. Syst. 18 (2000) 263–297, http://dx.doi.org/10.1145/354871.354874.

[2] J. Domżał, Z. Duliński, M. Kantor, J. Rzasa, R. Stankiewicz, K. Wajda, R. Wójcik, A survey on methods to provide multipath transmission in wired packet networks, Comput. Netw. 77 (2015) 18–41, http://dx.doi.org/10.1016/j.comnet.2014.12.001.

[3] J.J. Garcia-Lunes-Aceves, Loop-free routing using diffusing computations, IEEE/ACM Trans. Netw. 1 (1) (1993) 130–141, http://dx.doi.org/10.1109/90.222913.

[4] S. Vutukury, Multipath routing mechanisms for traffic engineering and quality of service in the internet (Ph.D. thesis), University of California, Santa Cruz, 2001, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.1725.

[5] C. Villamizar, OSPF optimized multipath (OSPF-OMP), Internet-Draft draft-ietf-ospf-omp-02, URL http://tools.ietf.org/html/draft-ietf-ospf-omp-02.

[6] I. Gojmerac, Adaptive multi-path routing for internet traffic engineering (Ph.D. thesis), TU Wien, 2007, http://www.gojmerac.com/Gojmerac_PhD-Thesis.pdf.

[7] J.M. McQuillan, G. Falk, I. Richer, A review of the development and performance of the ARPANET routing algorithm, IEEE Trans. Commun. 26 (12) (1978) 1802–1811, http://dx.doi.org/10.1109/TCOM.1978.1094040.

[8] J.M. McQuillan, I. Richer, E. Rosen, The new routing algorithm for the ARPANET, IEEE Trans. Commun. (1980) http://dx.doi.org/10.1109/TCOM.1980.1094721.

[9] A. Khanna, J. Zinky, The revised ARPANET routing metric, ACM SIGCOMM Comput. Commun. Rev. 19 (4) (1989) 45–56, http://dx.doi.org/10.1145/75247.75252.

[10] J. Zinky, G. Vichniac, A. Khanna, Performance of the revised routing metric in the ARPANET and MILNET, in: IEEE MILCOM '89, 1989, pp. 219–224, http://dx.doi.org/10.1109/MILCOM.1989.103929.

[11] D. Bertsekas, Dynamic behavior of shortest path routing algorithms for communication networks, IEEE Trans. Automat. Control 27 (1) (1982) 60–74, http://dx.doi.org/10.1109/TAC.1982.1102884.

[12] Z. Wang, J. Crowcroft, Analysis of shortest-path routing algorithms in a dynamic network environment, ACM SIGCOMM Comput. Commun. Rev. 22 (2) (1992) 63–71, http://dx.doi.org/10.1145/141800.141805.

[13] S. Low, P. Varaiya, Stability of a class of dynamic routing protocols (IGRP), in: IEEE INFOCOM '93, vol. 2, 1993, pp. 610–616, http://dx.doi.org/10.1109/INFCOM.1993.253311.

[14] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, RFC 3031, 2001, http://tools.ietf.org/html/rfc3031.

[15] A. Elwalid, C. Jin, S. Low, I. Widjaja, MATE: MPLS adaptive traffic engineering, in: IEEE INFOCOM 2001, vol. 3, 2001, pp. 1300–1309, http://dx.doi.org/10.1109/INFCOM.2001.916625.

[16] S. Kandula, D. Katabi, B. Davie, A. Charny, Walking the tightrope: responsive yet stable traffic engineering, ACM SIGCOMM Comput. Commun. Rev. 35 (4) (2005) 253–264, http://dx.doi.org/10.1145/1090191.1080122.

[17] S. Fischer, N. Kammenhuber, A. Feldmann, REPLEX: dynamic traffic engineering based on wardrop routing policies, in: ACM CoNEXT '06, 2006, pp. 6–17, http://dx.doi.org/10.1145/1368436.1368438.

[18] A. Kvalbein, C. Dovrolis, C. Muthu, Multipath load-adaptive routing: Putting the emphasis on robustness and simplicity, in: IEEE ICNP 2009, 2009, pp. 203–212, http://dx.doi.org/10.1109/ICNP.2009.5339682.

[19] N. Skrypnyuk, Load-sensitive routing (Master's thesis), TU München, 2006, http://www.net.t-labs.tu-berlin.de/papers/S-LSR-06.pdf.

[20] L.G. Roberts, The next generation of IP - flow routing, in: Proc. SSGRR 2003S International Conference, L'Aquila, Italy, 2003.

[21] I. Rubin, R. Zhang, Max–min utility fair flow management for networks with route diversity, Int. J. Netw. Manag. 20 (6) (2010) 361–381, http://dx.doi.org/10.1002/nem.740.

[22] J. Xu, K. Wu, Living with artificial intelligence: a paradigm shift toward future network traffic control, IEEE Netw. 32 (2018) 92–99, http://dx.doi.org/10.1109/MNET.2018.1800119.

[23] J. Roberts, S. S. Oueslati, Quality of service by flow aware networking, Phil. Trans. R. Soc. Lond. 358 (2000) 2197–2207, http://dx.doi.org/10.1098/rsta.2000.0641.

[24] S. Oueslati, J. Roberts, Comparing flow-aware and flow-oblivious adaptive routing, in: Proc. 41st Annual Conference on Information Sciences and Systems, CISS 2007, Baltimore, MD, USA, 2007, http://dx.doi.org/10.1109/CISS.2006.286549.

[25] J. Domżał, Intelligent routing in congested approximate flow-aware networks, in: IEEE GLOBECOM, 2012, http://dx.doi.org/10.1109/GLOCOM.2012.6503368.

[26] J. Domżał, R. Wójcik, D. Kowalczyk, P. Gawłowicz, P. Jurkiewicz, A. Kamisiński, Admission control in flow-aware multi-topology adaptive routing, in: IEEE ICNC 2015, Garden Grove, CA, USA, 2015, http://dx.doi.org/10.1109/ICCNC.2015.7069352.

[27] S. Zhang, W. Lei, An optimized BBR for multipath real time video streaming, 2019, arXiv:1901.09177.

[28] R. Cominetti, C. Guzmán, Network congestion control with Markovian multipath routing, Math. Program. 147 (1) (2014) 231–251, http://dx.doi.org/10.1007/s10107-013-0719-z.

[29] M. Mahdian, S. Arianfar, J. Gibson, D. Oran, MIRCC: multipath-aware ICN rate-based congestion control, in: Proceedings of the 3rd ACM Conference on Information-Centric Networking, ACM-ICN '16, ACM, New York, NY, USA, 2016, pp. 1–10, http://dx.doi.org/10.1145/2984356.2984365.

[30] J.J. Brzozowski, J. Leddy, C. Filsfils, R. Maglione, M. Townsley, Use cases for IPv6 source packet routing in networking (SPRING), RFC 8354, 2018, http://tools.ietf.org/html/rfc8354.

[31] R. Wójcik, J. Domżał, Z. Duliński, Flow-aware multi-topology adaptive routing, IEEE Commun. Lett. 18 (9) (2014) 1539–1542, http://dx.doi.org/10.1109/LCOMM.2014.2334314.

[32] R. Wójcik, A. Jajszczyk, Flow oriented approaches to QoS assurance, ACM Comput. Surv. 44 (1) (2012) 5:1–5:37, http://dx.doi.org/10.1145/2071389.2071394.

[33] R. Wójcik, J. Domżał, Z. Duliński, P. Gawłowicz, D. Kowalczyk, Performance evaluation of flow-aware multi-topology adaptive routing, in: IEEE CQR 2014 International Workshop, Tucson, USA, 2014, http://dx.doi.org/10.1109/CQR.2014.7152450.

[34] E. Kohler, The click modular router (Ph.D. thesis), MIT, 2000, http://pdos.csail.mit.edu/papers/click:kohler-phd/thesis.pdf.
[35] M. Handley, E. Kohler, A. Ghosh, O. Hodson, P. Radoslavov, Designing extensible IP router software, in: Proc. Symposium on Networked Systems Design & Implementation, 2005, pp. 189–202.
[36] P. Jurkiewicz, FAMTAR: An adaptive multipath software IP router implementation, https://github.com/piotrjurkiewicz/famtar.
[37] R. Wójcik, J. Domżał, Z. Duliński, P. Gawłowicz, P. Jurkiewicz, Loop resolution mechanism for flow-aware multi-topology adaptive routing, IEEE Commun. Lett. (2015) http://dx.doi.org/10.1109/LCOMM.2015.2439679.
[38] A. Botta, A. Dainotti, A. Pescapè, A tool for the generation of realistic network workload for emerging networking scenarios, Comput. Netw. 56 (15) (2012) 3531–3547, http://dx.doi.org/10.1016/j.comnet.2012.02.019.
[39] P. Jurkiewicz, G. Rzym, P. Boryło, How many mice make an elephant? modelling flow length and size distribution of internet traffic, 2018, arXiv:1809.03486.