Robust Network Coding in Transport Networks

Bence Ladóczki*, Carolina Fernandez†, Oscar Moya†, Péter Babarczi*, János Tapolcai*, Daniel Guija†
*MTA-BME Future Internet Research Group, Budapest University of Technology and Economics (BME), Hungary
†Distributed Applications and Networks Area (DANA), i2CAT Foundation, Barcelona, Catalonia (Spain)
*{ladoczki, babarczi, tapolcai}@tmit.bme.hu, †{carolina.fernandez, oscar.moya, dani.guija}@i2cat.net

***Abstract*—After several years of ignorance in practice, Network Coding (NC) is gaining more and more attention in wireless networks. On the other hand, performing complex in-network operations requires extra hardware, which is still a real barrier of a widespread deployment of network coding in transport networks. However, recent technological trends, such as Network Function Virtualization, enable the deployment of network coding capable middleboxes at network nodes without complicated hardware-update, while Software-defined Networking (SDN) makes it possible to steer traffic to these middleboxes. Furthermore, a practical networking scenario was recently identified – namely the single link failure resilient case when user data can be split into two parts – for which simple coding operation at the edge nodes is sufficient to reach all benefits that network coding can provide. Built on these results, we demonstrate in the GÉANT OpenFlow Facility that network coding can be easily deployed in transport networks and brings real benefits for video streaming and distributed storage use cases.**

***Index Terms*—network coding, instantaneous recovery, network function virtualization, software defined networks, GÉANT**

I. Introduction

Although the benefits of network coding were already demonstrated in wireless networks [1], it took a long journey to manifest its real benefits for transport networks (i.e., increased throughput, lower computational complexity, robustness against failures, security). This effort evolved from a purely information theoretic perspective, through a really efficient algebraic representation, to a combinatorial approach [2]. We demonstrate in the pan-European research and education network (GÉANT) that NC is a viable approach for providing instantaneous recovery from link failures, while it can also be applied for additional security in storage systems.

In transport networks, a survivable routing scheme has three utmost important features: fast recovery time, simplicity and capacity efficiency. The $1+1$ path protection – the most widespread single link failure resilient protection method – sends the user data along two disjoint paths (primary and backup). It is simple to calculate a routing (i.e., disjoint path-pair) and it provides fast recovery from any single link failure. On the other hand, it consumes twice as much capacity as the primary path. In [2] the Resilient Flow Decomposition (RFD) algorithm, based on network coding, proved able of maintaining simplicity and fast recovery as in $1+1$, while reducing its capacity consumption by $20-30\%$. RFD reaches these merits by showing that each optimal single link failure resilient routing – including $1+1$ – can be decomposed into three end-to-end directed acyclic graphs (DAGs). Thus, splitting the data at the source node into two parts A and B, incorporating redundancy through $A \oplus B$ ($\oplus$ denotes the exclusive OR (XOR) binary operator), and sending the three flows through these DAGs (which are in fact the robust network codes) ensures single failure resilience for the connections (shown in Fig. 1(a)). The *RFD recovery does not require packet retransmission or flow rerouting* (i.e., the time-consuming post-failure signaling is completely eliminated from the recovery process), performing instantaneous recovery upon failure.

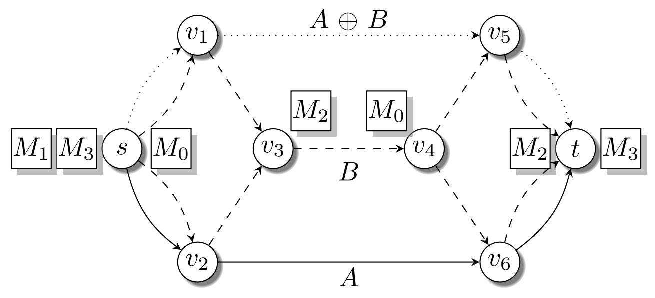
In this demonstration, we present the benefits of NC in transport networks through our RFD implementation on the well known butterfly topology, using the GÉANT OpenFlow Facility (GOFF) for the video streaming and distributed storage use cases [3].

II. Architecture

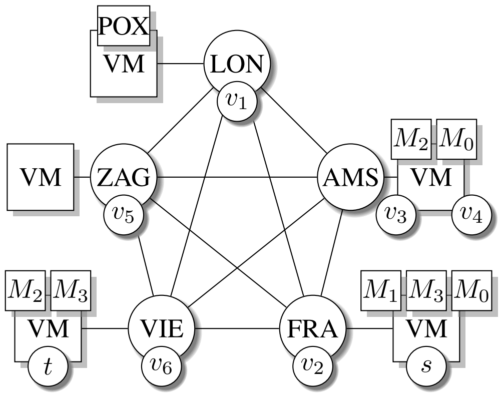
In RFD, coding and decoding is required only at the source and destination nodes. At the intermediate nodes the three DAGs can be routed independently from each other, where at most splitting and merging have to be performed besides simple forwarding. In order to identify the DAGs (flows) A, B and $A \oplus B$ we used a stacked MPLS label hierarchy as in [4]: the outer label identifies the DAG, while the other two contain the sequence number of A and B, respectively (or 0 if the flow is not included). Thus, for a successful RFD implementation, the following Network Functions (NFs) are required [5]:

- *Splitter* (M_0): duplicates incoming packets and forwards them through two different links (e.g., s and v_4 in Fig. 1).
- *Sequencer* (M_1): divides the input stream at the source node s into flows A and B (e.g., based on parity) and marks each with its own MPLS label.
- *Merger* (M_2): receives the same flow (i.e., with the same MPLS label) on two incoming links and forwards one of them (or the intact one upon link failure) through its single outgoing link. See nodes v_3 and t in Fig. 1.
- *Coding/Decoding* (M_3): these NFs are similar, as they perform fast packet processing using XOR operation and queues to handle the incoming packets. They are always placed at s and t.

This demo has been produced with the financial assistance of the European Union under the FP7 GÉANT project grant agreement number 605243 as part of the MINERVA Open Call project. P. Babarczi was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences (MTA).



(a) The single link failure resilient, robust network coding based solution of RFD (i.e., three end-to-end DAGs in the butterfly topology).



(b) The mapping of the butterfly to the physical GOFF topology and the corresponding NF placement at the VMs.

Fig. 1. Experimental setup in GÉANT (the OpenFlow switches and corresponding VMs are in London, Zagreb, Vienna, Frankfurt and Amsterdam).

III. Use Cases for Demonstration

In order to prove the usefulness and performance of RFD in transport networks we envisioned a pair of typical applications: video streaming and distributed storage. The experiments are deployed on the GOFF, on top of a 5-node full-mesh topology (see Fig. 1(b)). An SDN controller identifies or adapts a resilient topology (e.g., the butterfly in Fig. 1(a)) on top of the 5-node full-mesh and routes the DAGs accordingly between the nodes. Different NFs are placed on the VMs and connected to the network devices, thus intercepting and operating on incoming traffic as needed.

A. Video Streaming

In this scenario, a video is transmitted via UDP from the source s to the destination t through the end-to-end DAGs in Fig. 1(a). By placing appropriate NFs in the topology (see Fig. 1(b)), we are able to ensure instantaneous recovery after a single link failure occurring anywhere in the network. First, the video flow in s (Frankfurt VM) is split into flows A and B using NF M_1. A coded variant $A \oplus B$ is also generated in M_3. The three flows are then routed along their corresponding DAGs, i.e., A is sent to the Frankfurt switch (acting as v_2), $A \oplus B$ is sent to v_1 in London (tunneling through the Frankfurt switch) and B is sent to both nodes after traversing NF M_0. It is worth noting that Frankfurt and London switches send each a flow B to the Amsterdam VM (v_3), where NF M_2 merges them to avoid forwarding duplicated packets. After the B flow is split again in Amsterdam VM (v_4), the DAGs are routed to the destination t (Vienna VM) through the Vienna switch (directly via v_6 and tunneling from v_5 in Zagreb).

In the no-failure scenario, the video is reconstructed from A and B flows by removing the MPLS labels and restoring the corresponding checksums. If either A or B is broken (in RFD at most one DAG can be disrupted), the video stream can be instantaneously recovered by matching packets from two flows: any of the original (A or B) and $A \oplus B$.

B. Distributed Storage

RFD can also provide efficient solutions for bidirectional transmissions between a client device and multiple servers. The data is split into A, B and $A \oplus B$; then stored in different physical locations. Similarly to the video streaming use case, data can be instantaneously recovered with the application of RFD even if one of the storages is unavailable or a link failure occurred during the read or write operations. Furthermore, in order to enhance security, the three storages may belong to different clouds as well (e.g., Google Drive, Dropbox, etc.); thus distributing and minimizing the risk of a potential breach of the user's account data.

In this use case, a user attempts to either write or read some data through a client (e.g., laptop) that interfaces with a data center, consisting of a number of servers that store specific portions of the user's data (i.e., three different VMs in the GOFF in our demonstration). When the user attempts to write some information, the client applies a subset of NFs to split and encode the data. After that, the controller performs the operations related to network management, such as identifying three available servers from the underlying topology; and defines the appropriate rules in the network devices that allow routing information to each server. When the subsequent read petition is issued, the client must decode and merge as needed, taking into account any link or storage node failure.

References

[1] M. Hundeboll et al., "Catwoman: Implementation and performance evaluation of ieee 802.11 based multi-hop networks using network coding," in *IEEE Vehicular Technology Conference*, Sept 2012, pp. 1–5.

[2] P. Babarczi, J. Tapolcai, L. Rónyai, and M. Médard, "Resilient flow decomposition of unicast connections with network coding," in *Proc. IEEE Intl. Symposium on Inf. Theory (ISIT)*, June 2014, pp. 116–120.

[3] "MINERVA: Implementing network coding in transport networks to increase availability," 2013, [Accessed 15-December-2014]. [Online]. Available: http://www.geant.net/opencall/SDN/Pages/MINERVA.aspx

[4] F. Németh et. al, "Towards smartflow: Case studies on enhanced programmable forwarding in openflow switches," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 85–86, Aug. 2012.

[5] P. Babarczi, A. Pasic, J. Tapolcai, F. Németh, and B. Ladóczki, "Instantaneous recovery of unicast connections in transport networks: Routing versus coding," *accepted to Elsevier Computer Networks*, 2015.