Improving network resilience with Middlebox Minions

Lyn Hill, Charalampos Rotsos, Will Fantom, Chris Edwards, David Hutchison
School of Computing and Communications, Lancaster University
{l.hill4, c.rotsos, w.fantom, c.edwards, d.hutchison}@lancaster.ac.uk

***Abstract*—Resilience in networks has often relied on high availability to ensure minimal disruption to end users when faults occur, but this has grown difficult for retaining state with the growing popularity of hardware middleboxes – blackbox hardware network functions that have served as an important part of network design in recent years. There is potential room for the introduction of Network Function Virtualisation (NFV) in the field of resilience in connection with middlebox usage. Rather than relying on overprovisioning, we propose Middlebox Minion (MiMi) VNF, a system design that can be inserted around inaccessible hardware. This recreates state in accordance with the middlebox function, using NFV to establish stateful failover mechanisms without the need to replace existing hardware. The experiment we present is a failover analogy examining the importance of state retention and the complexities involved with inaccessible hardware. Results suggest a promising improvement of connection quality and up to 60% lower loss when state can be preserved across failover instances, as well as potential for further exploration of the topic area.**

I. Introduction

Network operators increasingly use middleboxes to improve network connectivity, designed to overcome limitations in network protocol design by processing packet headers in non-standardised ways supporting operations, like security enforcement, resource control and connectivity. Middleboxes predominantly use specialised hardware components to process traffic at line rate and rely on vertically-integrated control planes. As a result, they operate as a "bump-on-the-wire" and remain external to the network control plane (e.g. routing). Although middlebox failures are sporadic, they can disrupt network functionality [1] as they become a single point of failure in the network. Unlike TCP/IP protocol design principles, network middleboxes maintain essential per-flow state and persistent failures will render the network unusable, with even simple device reboots having a long-lasting degradation effect on network performance. Failure recovery typically relies on re-setting established network flows. A subsequent storm of new connections will force the middlebox to push a large amount of packets to the co-processor, which typically manages the initial state allocation. Nonetheless, the communication channel between the co-processor and the forwarding plane offers limited capacity and is designed to support sporadic packet exceptions [2]. This results in increased packet loss and potential incast effects over a period of time.

Existing network resiliency frameworks [3] rely on multi-stage processes using a mix of failure mitigation mechanisms for short-term remediation as well as long-term recovery. Remediation mechanisms restore partial service delivery, while recovery mechanisms restore service delivery. Several research efforts [4, 5] offer timely VNF failure detection, while the connectionless design of TCP/IP protocols ensures the eventual service recovery. Research on middlebox failure remediation has been limited, due their "black-box" design nature. In this paper we explore a novel view point on the use of VNF technologies: can virtualised software replicas provide an effective remediation mechanism for hardware middleboxes? The lack of standardisation both for design and interface renders fault-recovery infeasible in the case of significant fault, and we lack the control plane mechanisms to manage such failures in a timely fashion. The lack of external API or flexible design also isolates these physical network functions from anything besides its intended function [6]. This limits the number of approaches to resilience that can be utilised with this hardware [7]. State cannot be retained when faults occur across redundant devices, nor can replication be used without the ability to define consensus via protocols such as Paxos [8]. The fault tolerance that a middlebox supports is limited primarily to its own design and very simple redundancy through overprovisioning. The proliferation of middleboxes greatly exacerbates these limitations, with the potential for thousands of these devices distributed throughout the network [9].

This paper presents MiMi-VNF (Middlebox Minion VNF), an architecture to enable general-purpose state-resilience for hardware middleboxes. MiMi-VNF takes advantage of the flexibility of VNFs to create virtualised equivalents of middleboxes that, during periods of disruption or fault, can provide a fast remediation mechanism for service delivery. As built for purpose VNFs, they may be hardware-agnostic, sufficiently replicating the role of the middlebox regardless of the product used. This is achieved through state replication and traffic cloning to generic hardware placed in parallel to the middlebox. The performance limitations of virtualisation are minimised in this role, and their flexible design exploited. Specifically, the contributions of this paper are:

- We present a network architecture that improves hardware middlebox resilience, by replicating state using VNF minions.
- We implement a prototype of the proposed architectures and demonstrate the ability of the architecture to minimise network disruption.

II. Background

Resilience is an umbrella term that encompasses many approaches to network and system design. Sterbenz et al. [3] define resilience as "the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation". It is a vital part of system design, but difficult to do effectively. Middleboxes and NFV are no exception, with a body of work in recent years exploring the topic, especially for the issues of state preservation and the complexities they both pose. Several key challenges have been highlighted by work prior to this paper, including Remus [10], Pico [7] and FTMB [11]. Coarse grained approaches to replication are not faster than the tolerable limit of sub-1ms for live deployments as discussed in [11], whilst fine grain transaction logging requires access to hardware that is unlikely in live deployments. Satisfying all requirements is unlikely; sufficiently generalised, minimally impactful during normal operation and seamless in its recovery. The replacement of middleboxes with virtualisation is not a simple improvement, and has been the subject of considerable exploration for its resource consumption [12], organisation [13] and traffic steering [14].

There is no approach to resilience that will be sufficient on its own, and the growing complexity of networks only complicates the design of recovery efforts. The restrictions on performance impact and the black box nature of middleboxes further compounds this. Changing the nature of the system as it has evolved is tough. Several recent approaches attempt to remove middlebox reliance entirely through frameworks to execute their functionality as a rentable service, such as the In-Net framework by Stoenescu et al. [15] for rentable edge processing, or the APLOMB system by Sherry et al. [16] that pushes all NFV to the cloud. There has been little adoption of such system-changing approaches for both technical and financial reasons, leaving middleboxes firmly rooted in network design for the near future. These demonstrate that creating a generic, resilient and state-aware failover system, especially under these strict criteria, is a complex and difficult task. FTMB is a significant step in the right direction towards this high bar, but still requires that the middlebox be modifiable/accessible.

III. Middleboxes and Network Resilience

The work established in [11] is a step in the right direction, but its assumption on the capacity to modify middleboxes may be unreasonable. The argument of this paper is that middleboxes are unlikely to be replaced, but their significant presence in enterprise networks and minimal approach to resilience offers a niche for NFV to be introduced. Rather than replacing or modifying their existing deployments like prior research, a stateful failover system can be placed around these black boxes and facilitate the retention of state better than mere hardware redundancy. Areas in which this approach may be lacking such as performance are less important if the existing hardware remains the primary packet processor and the virtualised failover system used only in the event of failure. The motivation for this argument is not without grounding from work by others. Technologies such as DPDK and Nvidia's Cumulus [17] have pursued softwarised packet processing without compromising on the advantages of hardware via APIs to proprietary platforms such as Intel or ARM. These hybrid ASIC architectures can be observed as a technological trend where softwarisation is pursued in conjunction with accessible hardware without compromising on programmability or performance. Whilst MiMi in its early phases primarily explores both the concept and its benefits, the argument it also establishes forms the basis of its future work.

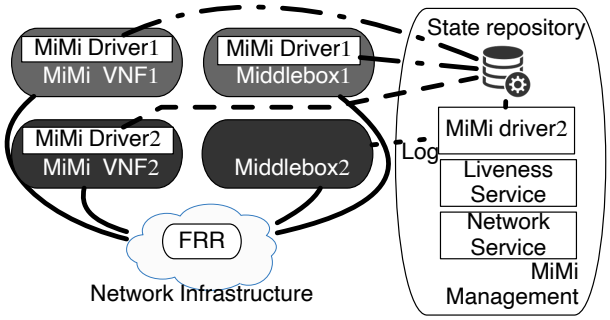

Fig. 1. MiMi Middlebox resilience architecture.

Architecture: Figure 1 presents MiMi's architecture. It is built on the basic assumptions that the network infrastructure supports VNFs and fast-failover routes such as MPLS [18] are in place when failures occur. Our architecture uses a best-effort mechanism to construct state and maintain partial packet processing and forwarding correctness during middlebox failures. This is achieved through 1:1 pairings of key middleboxes to VNFs, where a set of modules maintain state synchronisation between the primary middlebox and the minion VNF. In parallel, it uses the northbound API of the network control plane to establish a backup path to the VNF if the middlebox fails. The management layer uses off-the-shelf database software (e.g. Redis) to store the forwarding state of each middlebox. It is worth highlighting that the architecture has a minimal impact on the resource of the infrastructure during idle mode, as the backup VNF functionality will be limited to collecting state updates and signal liveness to the management layer. Furthermore, modern memory ballooning techniques in VMs can also minimise the need for internal memory fragmentation on the virtualised servers.

To minimise impact on service liveness, the middlebox state must be retained and transferred to the MiMi VNF during either operation or at the point of failure. The middlebox ecosystem is diverse however, and what can be defined as middlebox state is both traffic and application-dependent. For example, a load balancer will need to persist "client IP - outport" pairings and the target backend server for long streams, or current backend server load for load spread decision making. An application's tolerance for disruption defines the speed at which the VNF must operate to mask failures. Middlebox platforms vary significantly in their programmability and

transparency. Vendors typically keep their firmware closed for security purposes and do not allow user-defined applications to run on the co-processor. Nonetheless, the increasing adoption of network programmability has motivated network operators to adopt more open middlebox designs that allow user to script custom management policies [19].

As a result, our architecture adopts a flexible middlebox integration model. Specifically, to integrate a hardware middlebox with the MiMi platform, a VNF image must be defined that supports similar packet operations and manipulate traffic in an appropriate manner. Software such as Open Vswitch and the Click modular router can implement a wide range of packet processing operations at high speeds, as well as several subsystem from the Linux kernel. As part of this VNF selection process, the network manager much also define a custom schema from the middlebox state and develop drivers that can extract state from the primary middlebox, serialise it in the state repository and translate it into appropriate state setup from the MiMi VNF.

The difficulty of this depends on the nature of the middlebox; for a white box, an internal driver can be inserted into the box to serialise state, to be sent to the state repository and then to the VNF. The number of middlebox products that are modifiable in this fashion is unknown however, with the prevailing position being most devices disallow external modification. For these devices, external drivers can recreate state using information derived from system logs generated by the middlebox itself or created through observation of traffic patterns. This may not be sufficient to replicate state identically, but produces an approximation sufficient for remediation. If no metrics are available at all, it is possible to establish approximate state through symbolic execution when the middlebox configuration is known as discussed by Stoenescu et al. with SymNet [20].

Implementation: In order to evaluate the feasibility of our architecture, we have implemented a software prototype. The management layer of the MiMi architecture use a container-based micro-service architecture consisting of a liveness service, an SDN application supporting Fast-Reroute and a state serialisation middleground service. To evaluate the compatibility of our architecture with existing middlebox technologies, we focused on a load balancing service. The application schema stores a mapping between the source IP address and port number and the selected backend server IP. Due to limited access to real middlebox appliances, we use software load balancers to emulate two middlebox scenarios: an OpenFlow-based load balancer built as a Ryu application, and a NetFilter Source NAT policy. Both middlebox scenarios perform packet-level load-balancing and offer some form of state control.

Our OpenFlow application implements a reactive load balancing control application which uses a 5-tuple hash to randomise backend server selection and exact match rules to rewrite MAC and IP addresses, forwarding the packet to the correct output port. The MiMi driver is implemented as part of the application and use a Redis python library to synchronise

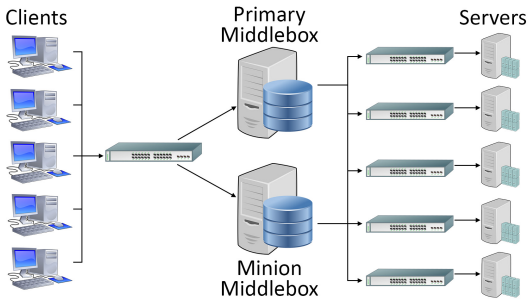

Fig. 2. Experimental topology emulating a caching service with 5 backend servers and 5 clients.

with the remote state repository. For our NetFilter-based load balancer we insert a set of rules in the NAT table of the post-routing chain that select a random backend server and forward the traffic to it. In this implementation, because the NetFilter rules are executed in the kernel, we opted to design an external MiMi driver. The driver uses the logging capabilities of the NetFilter system and implements a syslog server that can extract the relevant field from the logging information.

IV. Evaluation

In this section, we evaluate the ability of the MiMi VNF prototype to reduce TCP connection resets during a failure scenario. The experimental topology, depicted in Figure 2, emulates a caching service and consists of five clients accessing a load-balanced HTTP service supported by five backend servers. A *primary* hardware load balancer exposes an HTTP service via a virtual IP and each HTTP request is randomly redirected to a backend server, while a *minion* VNF is available to remediate service delivery, upon failures. The clients and the servers connect with the load balancers using OpenFlow switches configured with static L2 forwarding rules and a Fast Failover group table entry implementing a network failover mechanism; when the link to the primary middlebox is disabled, all traffic is rerouted to the minion VNF.

Our evaluation uses two workload models; requesting small web objects (WEB - short flows) and streaming DASH video streams (DASH - long flows). For the WEB workload, we used the WRK (v4.1.0) HTTP traffic generator running on two threads on every client and requesting a small web object (617Kb). We vary the number of parallel connections to ensure a certain level of utilisation in each experimental run. For the DASH workload, we run the scootplayer load generator, and serve the "Big Buck Bunny" video at high quality (8000kbit) segmented in 1 second chunks, with chunk sizes varying between 100Kb to 1.4 Mb. In both scenarios, the servers use the lighttpd daemon (v1.4.45). To emulate failure scenarios, we manually trigger link failures at fixed intervals, which force the OpenFlow switches to forward traffic via the minion VNF. During an experiment, we measure the number of failed connections and the overall traffic throughput. Our experiments executed on a Dell server (Dual Socket Xeon 4114, 20 Cores, 32gb RAM, Ubuntu 18.04) using the Mininet platform. Four scenarios are replicated for each experiment.

Firstly, we run the system without any failures and measure the performance when the flow replication mechanism of MiMi VNF is disabled (Clean) or enabled (Copied). This experiment is used to identify if the MiMi architecture during normal operation incurs any noticeable service degradation. Secondly, we trigger failures at fixed intervals while the flow replication is enabled (Failover) or disabled (Fail). For the WEB workload, we run the experiment for 5 minutes and trigger a failure every 30 seconds (5 state transfers). For the DASH workload, we trigger 15 evenly spaced link failures for the experimental duration. It is worth noting, that we mirror state only from the primary middlebox to the minion VNF.

Resilience Evaluation: In our experiment we use two driver designs for the MiMi VNF; state recreation via logging and direct driver insertion into the primary middlebox. In both scenarios, state is serialised to JSON and stored in a middleground (mainly Redis) separate from the two middleboxes. We initially tested our Ryu load balancing application both as the primary middlebox and the minion VNF, and we report the average and standard deviation number of failed of connections across 5 experimental runs in the upper half of Table I. The DASH workload consists of long-running TCP connections with little disruption tolerance. This is a significant volume of traffic for OpenFlow, staggered to begin in equal sets to maintain a consistent level of traffic.

From the results, we observe that for both low (50 client) and high traffic volumes (100 clients), our MiMi driver reduces the number of failed connections by 16% and 18% respectively. The results are consistent if not substantial; there is some measure of improvement, but scootplayer is intolerant of significant disruption. The WEB workload exhibits more clearly the benefits of the MiMi sytem with a reduction in connection timeouts by 41%. Timeouts inevitably spawn rebroadcasted packets, with a 36% increase in requests over fault-free operation. When transferring state, an increase of only 17% occurs. The trend can also be observed with 2000 connections, with a reduction of timeouts of 40% and 33% for traffic respectively. Applications with better robustness to failure exhibit a more distinguishable reduction in the total number of failed connections. This is not to suggest that systems with heavier traffic are inappropriate for retention of state, but that the scenario of repeated failures with transfer in one direction is not as accurate to a real world deployment. Average rates of latency and impact of this approach on fault-free operation is minimal and within normal range of deviation.

In order to evaluate our state recreation via logging driver model, we use our Iptables middlebox as the primary load balancer. There is an expected but significant performance difference between the primary and minion load balancer in this scenario, observable in the orders of magnitude difference in packet processing speeds and this affects our ability to compare the performance of the DASH workload and we restrict our analysis to the WEB workload exclusively. The lower half of Table I presents the average and standard deviation of the number of failed connections of the WEB workload for a varying number of parallel connection across five experimental runs. From the results, it is evident that the trend of timeout reduction continues with an observable change of 56 to 60% in 20-30K connections respectively. This is a significant reduction in rates of connection failure and will become more pronounced with higher volumes of traffic.

Ryu/OpenFlow middlebox

Workload	Clean	Copied	Fail	Failover
DASH (50 sessions)	12 (1)	15 (1.5)	36 (3)	29 (2)
DASH (100 sessions)	51 (2)	56 (3)	89 (5)	71 (3)
WEB (1k sessions)	2296 (73)	2057 (96)	4167 (265)	2496 (156)
WEB (2k sessions)	1177 (97)	1295 (150)	7144 (324)	4278 (212)

IPtable middlebox

Workload	Clean	Copied	Fail	Failover
WEB (20k sessions)	9497 (502)	10444 (427)	28331 (1388)	12417 (1326)
WEB (30k sessions)	13737 (461)	15512 (418)	45220 (3275)	21672 (865)

TABLE I
AVERAGE (STANDARD DEVIATION) CONNECTION FAILURES USING A MIMI VNF LOAD-BALANCER FOR DIFFERENT WORKLOADS ACROSS FIVE EXPERIMENTAL RUNS.

V. CONCLUSION

In this paper we have presented MiMi, a prototype and initial exploration of the problem of building resilient systems around fixed hardware middleboxes. Middleboxes present a difficult goal for maintaining state across failures, but NFV offers considerable potential in its adaptability, especially when placed in an auxiliary role where its comparative disadvantages are diminished. The MiMi VNF architecture offers significant improvement in connection retention both for long and short flows, with state reconstructed from logging acting as a first step to more generic state preservation. This paper lays the groundwork for an idea and argument that to our knowledge has not been investigated elsewhere. Current technology trends, especially in regard to hybrid models of softwarised networks and underlying hardware, demonstrate that there is an appropriate and growing domain for this sort of concept. We believe it will grow more relevant as SDN further enters more traditional networking. For future work, investigate alternative mechanisms for state reconstruction, including symbolic execution and logging, to integrate efficient packet processing platforms, including the Click router, and to explore VNF management techniques, such as scaling and clustering.

Acknowledgement The authors gratefully acknowledge the support of the Next Generation Converged Digital Infrastructure (NG-CDI) Prosperity Partnership project funded by UK's EPSRC and British Telecom plc.

References

[1] R. Potharaju and N. Jain, "Demystifying the dark side of the middle: A field study of middlebox failures in datacenters," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 9–22.

[2] R. Oudin, G. Antichi, C. Rotsos, A. W. Moore, and S. Uhlig, "OFLOPS-SUME and the art of switch characterization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2612–2620, 2018.

[3] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Comput. Netw.*, vol. 54, no. 8, p. 1245–1265, Jun. 2010.

[4] G. Reali, M. Femminella, and L. Monacelli, "Probabilistic codebook-based fault localization in data networks," *IEEE Transactions on Network and Service Management*, vol. 15, pp. 567–581, 2018.

[5] R. Abhishek, S. Zhao, S. Song, B.-Y. Choi, H. Zhu, and D. Medhi, "BuDDI: Bug detection, debugging, and isolation middlebox for software-defined network controllers," in *2016 12th International Conference on Network and Service Management (CNSM)*, 2016, pp. 307–311.

[6] J. Sherry and S. Ratnasamy, "A survey of enterprise middlebox deployments," 2012. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-24.pdf

[7] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13. New York, NY, USA: Association for Computing Machinery, 2013.

[8] L. Lamport, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, December 2001. [Online]. Available: https://www.microsoft.com/en-us/research/publication/paxos-made-simple/

[9] S. Huang, F. Cuadrado, and S. Uhlig, "Middleboxes in the internet: A HTTP perspective," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, 2017, pp. 1–9.

[10] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*. San Francisco, CA: USENIX Association, Apr. 2008.

[11] J. Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. a. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-recovery for middleboxes," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, p. 227–240, Aug. 2015.

[12] S. Li, M. Y. Saidi, and K. Chen, "Survivable virtual network embedding with resource sharing and optimization," in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, 2015, pp. 1–6.

[13] P. Das and P. M. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing," in *2013 IEEE Conference on Information Communication Technologies*, 2013, pp. 473–478.

[14] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 487–507, 2019.

[15] R. Stoenescu, V. Olteanu, M. Popovici, M. Ahmed, J. Martins, R. Bifulco, F. Manco, F. Huici, G. Smaragdakis, M. Handley, and C. Raiciu, "In-net: In-network processing for the masses," in *Proceedings of the Tenth European Conference on Computer Systems*, ser. EuroSys '15. New York, NY, USA: Association for Computing Machinery, 2015.

[16] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 13–24.

[17] IDC, "The business value of nvidia ethernet switch solutions for managing and optimising network performance," Tech. Rep., April 2021. [Online]. Available: https://resource.nvidia.com/en-us-ethernet-switching/idc-biz-value-ethernet-swtich-wp

[18] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to rsvp-te for lsp tunnels," Internet Requests for Comments, RFC 4090, 2005.

[19] "Engineered elephant flows for boosting application performance in large-scale clos networks," Broadcom Corporation, Tech. Rep., March 2014. [Online]. Available: https://docs.broadcom.com/doc/1211168569445

[20] R. Stoenescu, M. Popovici, L. Negreanu, and C. Raiciu, "SymNet: Scalable symbolic execution for modern networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 314–327.