Localization of network service performance degradation in multi-tenant networks

Vuleti, Pavle; Bosak, Bartosz; Dimolianis, Marinos; Mérindol, Pascal; Schmitz, David; Wessing, Henrik

Published in:
Computer Networks

Link to article, DOI:
10.1016/j.comnet.2019.107050

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Localization of network service performance degradation in multi-tenant networks

Pavle Vuletić, Bartosz Bosak, Marinos Dimolianis, Pascal Mérindol, David Schmitz, Henrik Wessing

A R T I C L E   I N F O
Article history:
Received 24 March 2019
Revised 1 December 2019
Accepted 10 December 2019

Keywords:
Performance verification
Fault localization
Virtualized networks

A B S T R A C T
Modern network services are in most cases virtualized and the traffic of various users is multiplexed over the same physical links. Traditional network monitoring methods which predominantly rely on the physical interface monitoring are not sufficient because they do not provide the insight into the behaviour of the traffic in the specific network service instance or per-user service experience. This paper presents NetMon, a framework that gathers performance indicators inside the network service and at the same time allows an efficient spatial performance degradation localization capability. NetMon is technology agnostic and suitable for various network technologies ranging from legacy to the new type of services based on network function virtualization, chaining service functions or programmable network elements. The paper presents an original solution to the problem of scalable active network service monitoring with the capability to distinguish the measurements of different virtual networks. The proposed hybrid monitoring method which mixes the active probing and capturing this probe traffic provides a good trade-off between the granularity of the results and the traffic overhead created by the monitoring system itself. The system was tested in the pan-European GEANT network which connects European research and education infrastructures and showed reliable performance over long periods of time.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Monitoring and performance verification are essential groups of processes in network service operations. These processes allow providers to verify the health of the products they are offering, the operation of the underlying infrastructure and to prove the quality of the services delivered to their users. Service users might as well want to have the insight into the performance of the services they use especially in cases of strict quality of service requirements. Network services are nowadays in most cases virtualized and the traffic of various users is multiplexed over the same physical links. With this type of services it is impossible to estimate the quality of users’ experience or to verify the Service Level Agreement (SLA) by using legacy tools which monitor physical links and network element operation. One can easily imagine a situation in which the physical link operates without any flaw or congestion, while on the other side one or multiple users do not get the proper service over that link due to the virtual network misconfiguration or some problem in the network virtualization software stack. Therefore new monitoring methods for performance verification within a network service are required.

Verifying network service performance requires gathering well-known service performance indicators like packet latency, latency variation (jitter) and/or packet loss rate between the service endpoints, and comparing them to some threshold values. Straightforward solution would be to obtain these metrics leveraging standard protocols (e.g. OWAMP or TWAMP) for active probing between all the service endpoints, as used in some tools like perfSONAR [1] or RIPE ATLAS.1 However, the number of such measurements in virtual networks is proportional to the number of

1 https://atlas.ripe.net/.
network service instances (one service instance is a virtual net-
work service provided to one specific user). In multipoint virtual
networks this number roughly grows with the square of the num-
ber of endpoints for each service instance. Service monitoring us-
ing dedicated devices and service agnostic tools, like those men-
tioned above, at each service endpoint, arises scalability issues and
increases the cost of the monitoring solution.

The environment in which network services are offered is also
very complex. Single service providers are not always capable to
offer network services required to support day to day operations of
companies, public institutions or scientific projects due to the lim-
ited geographical service coverage or constraints in service portfo-
lios. In those situations other providers are needed to fill the gaps
towards all the service end points and the service is provided by a
consortium of providers. Any case where more than one provider
or institution is required to provide the service is a case of multi-
domain or federated service operations. Automating processes in
a federated environment is a challenging task due to the domain
autonomy and clear separation of authorities over the resources in
the participating domains [2]. This often leads to lengthy and in-
efficient fault-to-resolution process flows and blame games around
the responsibility for some service faults. Automated fault localiza-
tion processes would significantly improve fault-to-resolution time
and the overall user’s experience.

This paper presents NetMon, a framework that gathers network
service performance indicators inside the federated network ser-
tice towards performance degradation localization. Service avail-
ability is defined as the percentage of time during which the
service operates within the agreed performance boundaries [3].
Therefore performance degradations often lead into the reduced
availability of the service and user’s conclusion that the service is
not available (or faulty) in some periods; despite the fact that the
packets can flow end-to-end all the time. This makes the problem
of service performance degradation localization similar but not
the same as the problem of network fault localization. Network ser-
vice performance degradation localization or network service fault
localization is more complex problem than the network fault lo-
calization as the faults in the network service cannot be easily de-
tected by observing alarms raised upon the typical network out-
ages (e.g. link, interface or device outages). In the remaining part
of the text, in the description of NetMon functionalities, for brevity
and because problem statements are similar, fault localization will
sometimes be used instead of network service performance degra-
dation localization. However the reader should be aware of this
discussion and read this as network service performance degra-
dation localization.

Unlike legacy active monitoring tools [1] or RIPE Atlas, Net-
Mon is capable to automatically localize the zone inside the net-
work service where there is a degradation of the performance and
thus reduce the time to complete service fault-to-resolution pro-
cess flows, especially in federated service environments. Solution
scalability in terms of the number of monitoring agents is im-
proved using recent advances in computer and network stack vir-
tualization (e.g. using network namespaces on monitoring agents).
NetMon system is network technology agnostic and suitable for
performance verification of various network technologies ranging
from legacy (like MPLS VPNs) to the new type of services based
on network function virtualization, chaining service functions or
programmable network elements. NetMon assumes that the enti-
ties that are managing the network also manage the monitoring
system. Such assumption ensures that it can always be configured
that the probe packet path can be the same as the service delivery
path. Key contribution of the paper and NetMon system is service-
aware network service fault and performance-degradation localiza-
tion mechanism presented in Sections 4 and 5. NetMon system
was built as an attempt to automate network management pro-
cesses and to find an efficient solution for the network service per-
formance verification in the GÉANT2 network environment where
network services between the end-users are provided jointly by
the National Research and Education Networks (NREN) and GÉANT
network - an example of a federated network environment.

The paper is organized as follows: Section 2 gives an overview
of the related research and standardization work in the area of
fault localization; Section 3 analyses different approaches to
network service performance monitoring and verification; Section
4 describes chosen strategies to overcome key challenges in
multi-tenant network performance verification and key compo-
nents of the NetMon system; the next section gives the fault lo-
calization algorithm used to spatially localize the faults in the net-
work and the last section gives an example of the use of the sys-
tem in the real network environment.

2. Related work – fault localization

Fault localization in networks is understood in the literature in
two different ways. Vendors and standardization bodies typically
define a process of fault localization as a problem of finding a spa-
tial location in the network where the problem has occurred. The
process consists typically of the service sectionalisation and per-
section testing which allows the faulty segment detection [4]. In
the research community the majority of the papers explore fault
localization as detecting the root cause of the problem from the
set of observed metrics and symptoms rather than the spatial lo-
calization of the problem in the network. A comprehensive recent
overview of research efforts on fault localization methodologies in
various network setups including overlay and virtual networks is
given in [5].

As it can be seen, there is a significant body of work in the area
of fault localization, but considerably less attention is given to the
virtual network fault localization which should probably change in
the future due to the advances and widespread use of virtualized
infrastructures. One of the research topics on the fault localization
in virtualized overlay networks is the event correlation in over-
lay and substrate networks which can lead to the detection of the
root cause of the problem [7]. In the other related papers, authors
aim to solve one of the key monitoring issues in overlay networks
which was mentioned in the Introduction section: scalability of the
measurement methodology due to the explosion in the number of
service endpoints with the increase in the number of users. In
[8,9] authors proposed methods to lower the number of measure-
ments from n2, where n is the number of endpoints in the virtual
network. Although such methodologies can be used for root cause
analysis, the fact that their result is an estimation of the real met-
ic between the service endpoints makes them hardly usable for
the verification of the contractual obligations and the relationship
between the provider and service users. In [6] authors proposed
the method to spatially localize the problem in the underlying sub-
strate network based on the active end-to-end measurements on
the overlay from the strategically chosen set of measurement end-
points. Similarly to the previously mentioned papers, the result of
the measurements is an estimation of the metrics in the overlay
with a limited accuracy. Unlike these previous research attempts
which give an estimation of the real performance metrics Net-
Mon is a system which provides deterministically measured per-
formance metrics and compares them to the SLA targets and is
able to spatially localize the degradation of network service per-
formance in a way which does not allow the disputes of the inter-
ested parties.
As mentioned above, spatial localization of the fault consists of sectionisation and per-section testing. Xia et al. [10] defined the concept of the monitoring zones as subsets of network elements where the performance of the zone can be estimated by capturing packets ingress and egress to the zone and analysing key performance indicators from these captures and their timestamps. Sectionalisation of the network into monitoring zones can also be a strategy for monitoring and localizing faults in all those situations where the service consists of multiple service elements (e.g., chained service functions where a monitoring zone corresponds to each service element separately). The same problem of spatial performance degradation detection was addressed recently by the IETF. Packets which flow through the network are alternatively marked in order to allow latency and loss measurements on intermediate monitoring zones [11]. In this approach network elements implement the detection of the blocks of packets with the same mark and thus estimate the performance parameters. Both methods mentioned in this paragraph require the out-of-band transport of the packet time information recorded at different network elements towards the central station for the analysis and have similar issues as other out-of-band methods (e.g. capturing packets using network taps and packet brokers) with this traffic transport overhead which is further discussed in Section 4.3. Similarly to these methods, NetMon uses sectionalisation of the network and gets the performance information at the zone boundaries, but instead of using alternatively marked packets as in [11] it does not have a place for storing a service information, it uses a modified version of OWAMP [13] protocol which enables deterministic per-network service and per segment monitoring as it will be described in the paper.

The emergence of programmable network elements and In-band Network Telemetry using P4-programmable network interface cards [12] allows new methods for fault localization. In a fully P4-enabled network it would be possible to add into the packet the time it entered the device at each hop as it traverses the network, and from the list of the timestamps received at the destination to deduce per-segment latency and latency variation. Still for the complete view of the service operation between all the service end-points, the telemetry data has to be transported from the packet destinations to the central result repository through some out-of-band channel. Strategy for per-network service and per-zone performance verification described in this paper can be reused for in-band network telemetry in a P4 enabled environment.

3. Strategies for network service performance verification in sectionalised network

Production environments usually use three performance metrics to describe QoS: packet latency, latency variation and packet loss rate (PLR). If defined in the SLA, these parameters are monitored during the whole service operations phase. Other well-known parameters like the offered capacity or throughput can be specified in the SLA, but are typically not measured continuously during the service operation. Bandwidth tests (e.g. available bandwidth or TCP throughput) are often intrusive and may disrupt the service by creating temporary congestion that can produce packet loss in users' traffic or decrease available throughput. Therefore those parameters are typically measured before the service is put into production or as a part of the troubleshooting procedures [15].

IETF proposes two modes of measuring SLA parameters in frame mode MPLS networks from the routers [16] which are applicable to other network services: direct and inferred. Inferred mode is an active measurement methodology for measuring PLR, latency and latency variation which assumes sending probe packets into the network and inferring the performance parameters from the probe packet arrival times. On the other hand, direct mode can be used only for PLR estimation. Direct mode counts regular user’s packets flowing through the network and derives metrics from that data by correlating the data about the lost packets with the network flow or similar traffic statistics data. Direct mode which passively reads network element variables is not suitable for delay and jitter metrics, unless there are some features in devices which allow the measurement of SLA parameters by again using active probing (e.g. Cisco SLA or Juniper RPM) and which write measurement results in specific SNMP MIBs. However such features are typically not interoperable between different vendors’ equipment, and as such are of limited value in multi-domain environments which are likely multi-vendor as well.

3.1. End-to-end measurements vs. metric composition

Monitoring multi-domain network service and looking into the end-to-end and per-domain performance parameters is technically the same as monitoring network service in a single network which is sectionalized into separate zones. Research literature distinguishes two general strategies for SLA verification and service monitoring in multi-domain networks, both based on active probing: end-to-end measurements and metric composition.

End-to-end measurements mainly assume the use of active methods to measure latency, latency variation and loss between service endpoints (e.g. OWAMP or TWAMP protocol). Active measurements are performed by measurement agents (MA) who inject synthetic probe traffic into the network and from its statistical properties at the receiving end infer measured parameters. Active measurements are performed end-to-end (between the customer edge (CE) or provider edge (PE) devices, depending on the contracted service type), thus avoiding the problem of accessing network elements in the remote domains. Such approach ensures accurate depiction of network impact on SLA parameters as synthetic traffic can be forced to traverse the same path as customer’s traffic between the key service points using the same processing. Scalability presents the main drawback of the end-to-end measurement approach which becomes evident issue for the multi-point multi-instance services. For the service that has M instances where each service instance $x \in \{1, 2, ..., M\}$ has nx end points, the total number of MAs is $N_{max} = \sum_{x=1}^{M} b_x$ which can quickly grow in large networks with a significant number of service instances. This estimation of the number of MAs assumes that a separate measurement agent is needed at each end-point in each service instance.

Metric composition [14,17] proposes a solution to address the scalability of the end-to-end measurements. SLA parameters are measured per domain and then end-to-end performance is estimated by using metrics’ nature, i.e. latency is additive, latency variation is described using mean, variance and skew of the contributing variations along the path and PLR is indirectly multiplicative. Through metric composition, number of MAs is significantly reduced in comparison to end-to-end measurement case as particular measurements may be reused for different services and instances. Total number of MAs in this case is $N_{max} = \sum_{x=1}^{M} b_x$, where $b_x$ is the number of cross-border connections of the domain $x$, and $D$ is the number of domains. The number of MAs may be even higher depending on the intradomain and interdomain topologies, but still significantly less than the end-to-end scenario. Several caveats are associated with metric composition, mainly tied to the level of accuracy especially for latency variation estimation [18]. Furthermore, there are other issues that negatively impact the estimation accuracy such as: the responsibility for the border link measurements, double measurements on MA links, time synchronization of the measurements in different domains, etc. Additionally, unlike end-to-end approach where the measurement results should just be transferred from the MAs to the result repository, metric composition requires also a central place to process all the
gathered data from the domains and calculate the performance metrics. Also the exposure of per-domain data towards the central measurement gathering entity, in-instance measurements deployment and service instance topology changes further complicate metric composition implementation and system design.

Despite the fact that metric composition has inherent fault localization capability because metrics are measured separately per domain, due to the higher reliability and accuracy of the results of end-to-end active performance measurements and lower total cost, it was chosen that the NetMon system uses hybrid approach based on active probing method. Section 4 will describe chosen NetMon strategy and what in addition to the active probing was needed for fault localization and how the scalability problem was resolved.

3.2. Analysing users traffic

Recent study [19] argued that due to the higher reliability of modern network services in uncongested networks packet loss can be inaccurately estimated if only active probes are being used. Active packet probes either make a snapshot of a small fraction of total transmission time which is not a representative sample of the population or could take different paths from the user’s traffic in case of network traffic load balancing. Authors suggest that user’s packets and their performance parameters should be analyzed instead. Nowadays this can be done easily without any impact to the performance of service delivery using optical network taps or packet mirroring on network elements, so that the copy of the traffic is diverted to the analysis stations. However such an analysis of the users traffic does not come without a new set of issues: from technical like detecting that particular packets belong to specific user or network flow, to potential privacy issues and the total cost of the monitoring solution which can be significantly increased due to the higher required processing capability at the monitoring sites and higher transport overhead which is needed when moving user packet performance data to the performance analysis stations. Some of these issues are addressed in the next section where further justification for the chosen NetMon strategy is given. Still, [19] is important because it summarizes key performance verification considerations, such as flow identification, which are the same in both user traffic performance analysis and general fault localization.

4. Localising faults in virtualized network environments - NetMon approach

As discussed in the previous section, more accurate and reliable end-to-end performance verification results can be achieved using some active monitoring protocol between the service end-points than through the metric composition. Active probing in multi-service environment requires injecting probe packets into the specific target service instance. This can easily be done at the provider edge devices by bridging or routing these packets from the probing device (Monitoring agent) into the target service instance. However, end-to-end probing alone does not provide insight into the performance on specific segments on the paths between the end-points and thus does not have the fault localization capabilities.

Spatial performance degradation localization requires the information about the packets (e.g. packet arrival timestamp) along their path through the network. By comparing the timestamps of the arrival of the same packet from various points along its path in the network it is possible to calculate the time it took for a packet to traverse each of the monitoring zones (segments between the two points where the timestamp was taken) which further enables the detection of the performance degradation on a specific zone. Fig. 4.1 shows an example of such strategy. In the figure the network is divided into three monitoring zones, where packet capturing elements (PC) are at the boundaries of the zones. PCs capture packets using mirroring from the network elements or tapping on the links, detect chosen packets and send the timestamp information with the specific ID of the packet to the central correlating element (Cor) which can accurately compare the timestamps and calculate the performance parameters for each monitoring zone and the links between the zones. In case of a packet loss in some of the zones, packet will be observed at the ingress boundary of the zone, but not at the egress points during the timeout period when the arrival of the packet is expected at the egress boundaries (discussed further in Section 5). Naturally all PC elements have to be time synchronized.

The density of the packet capturing elements (or the size of the monitoring zone) depends on the desired granularity of the fault localisation and the total cost of the solution. The cost of the solution consist of the cost to establish each PC (a physical or virtual machine), to connect it to the respective network element (a port might be required) and the cost of the network overhead required to transport performance related information to the Cor, which further discussed in detail in chapter 4.3. The price of the PC element depends on the packet capturing technology in use. PC could be an expensive server requiring dedicated packet capturing NICs or network analysis accelerators if all the traffic is captured on a high speed link and if probe packets have to be filtered from it. On the other side if mirror filtering on network elements is used, then it is possible to mirror only probe traffic towards the PC devices in which case inexpensive devices or even virtual machines, as in our testbed, can be used. The size of the monitoring zone can range from very small monitoring zones, which consist of a single network segment, where packets are captured at each network element along the packet path and performance parameters are calculated per network segment, to the monitoring zones, which correspond to the whole network (e.g. single provider). In the latter case, if multi-domain network

![Fig. 4.1. Fault localisation: principle of operation.](image-url)
services are being used, it is possible to detect which domain contributed the most to the performance degradation of the service. Such a feature is particularly useful for debugging multi-domain services, as appropriate corrective action can be requested from the domain responsible for the performance degradation.

Although the principle of spatial performance degradation localization described above seems to be simple, its implementation is not trivial in multi-service multi-tenant networks as there are several issues that have to be resolved in order to enable accurate capturing of the per-segment performance metrics:

- Detecting that the packet belongs to the specific service instance,
- Detecting exactly the same packet along its path through the network,
- Time synchronization issues - Accurately inferring what happened with any specific packet,
- Operating in federated environments issues.

Sections 4.2 - 4.5 describe the way previously mentioned issues are solved in NetMon. However, before that NetMon system architecture is briefly described as the roles of its components are important for understanding how these mechanisms are implemented.

4.1. NetMon architecture

The architecture of the NetMon system (Fig. 4.2) has the same logical components found in other similar monitoring systems [20-22]:

- Monitoring Controller (MC)
- Monitoring Agents (MA)
- Monitoring Result Repository (MRR)

In addition to these components, NetMon architecture also includes Packet Capturing elements (PC) and Correlator (Cor) for performance degradation localization. The roles of all components are described in more detail in the subsequent sections.

4.1.1. Monitoring controller (MC)

The top-level component in the NetMon architecture is Monitoring Controller (MC). MC controls the operation of the other NetMon components and stores all the relevant information about the services which are being monitored. It can be used as a stand-alone tool through the MC web portal, but also can be integrated with other OSS/BSS management components via REST TMF3 Service Test Management API interface for invoking new measurements from the service provisioning tools. NetMon supports both options. In the former case, the user would have to manually add all the relevant service information (e.g. service end-points, configuration parameters, the set of measurements that should be performed - service test specification, etc.) into it and invoke the measurements. In the latter case NetMon obtains the required information from relevant service and resource inventories through the standard TMF Service Inventory API interface. Each service instance has a unique Service ID. For one service instance it is possible to choose one or more service test specifications from the templates with the description of performance tests. MC creates a Service Test instance with the unique Service Test ID. Then it provides configuration information to the MRR component, needed for proper performance result presentation, and configures MA devices, initiating the monitoring. Since every Service Test instance has separate Service Test ID, depending on certain needs and according to different test specifications, it is possible to process multiple different measurements of the same service instance at the same time.

4.1.2. Monitoring agents (MA)

Monitoring Agents (MAs) are Linux virtual or physical machines used as sources and sinks of the active probe traffic. MA is being used to measure key performance indicators (latency, latency variation, loss) end to end. Each MA device resides in multiple VPNs simultaneously, using separate Linux network namespace for each VPN on the connection towards the network element it is connected to. This way it is possible to inject the probe traffic into multiple user VPNs and monitor the performance of the user service instances from a single MA device thus reducing the scalability problem mentioned in Section 3.1. MAs use modified one-way ping OWAMP [14] code, capable to store Service identifier and other relevant information in the probe packets, as explained in the Section 4.2. This approach enables per-service instance fault localization by capturing probe packets encoded with the specific Service Test ID.

4.1.3. Monitoring Result Repository (MRR)

The Monitoring Result Repository (MRR) is the main component for gathering, storing and displaying monitoring data flowing

---

3 TeleManagement Forum Open APIs: https://www.tmforum.org/open-apis/.
from MAs, MC and Cor. It is based on InfluxDB and Grafana visualization platform. End-to-end network metrics (latency, latency variation, loss) obtained from MAs are collected and stored with their corresponding Service ID. Per-zone monitoring results needed for fault localization are gathered from the Cor. In both cases Advanced Message Queuing Protocol (AMQP) messaging is used as a transport method for sending these results. MC provides the MRR with SLA thresholds for all active service tests. These are required for the monitoring dashboards that depict all the relevant performance metrics and RAG indicators for threshold violation. In case of threshold violation, MRR can send notifications to the service operations centre.

4.1.4. Packet Capturing Elements (PC)

PCs are implemented as Linux virtual or physical machines. In the implementation given in Section 6, PC interfaces are connected to the mirror ports of the network elements (in this case vMX routers). PCs filter probe packets sent by MAs from the traffic tapped at these interfaces. The capturing is based on extended Berkeley Packet Filter (eBPF)\(^4\) using a recent Linux kernel (at least v4.10). PC detects the probe packets and gets the timestamp at the moment of capturing. The summary information about the packets (service ID, packet identifier and timestamp at the moment of capturing) is sent to the Cor for further processing.

PCs currently support the following encapsulations: pure IP, QinQ, IP over MPLS, implicit MPLS pseudowire (without control word), explicit MPLS pseudowire (with control word), some combinations of the two (if kernel 4.10 is used). With the newer kernels (4.18) it also supports more general combinations of packet tags, up to the three levels, where each level allowing 5 consecutive MPLS labels or VLAN labels. GRE and VXLAN encapsulations are under development.

4.1.5. Correlator (Cor)

The Correlator is the central element of the monitoring system; it gathers information about the captured packets and calculates per-zone performance parameters. It requires only the knowledge of the set of PC with their identification. This information is provided by the MC. As input it receives the raw reports from the PCs and, by processing them at a given time granularity, it produces time and spatial refined statistics as output (pushed towards the MRR). Cor can work as a stand-alone tool or as a distributed Correlator in order to balance the load coming from multiple network flows or to work in a separate domain if additional data protection is required, as described in 4.5. The detailed description of the operations performed by the Cor is given in Section 5.

4.2. Packet to service identification

Virtualized network services can be implemented using various technologies. One of the most common approaches is to insert dedicated headers into the packets which are used to differentiate the packets belonging to specific virtual service instances (e.g. MPLS labels or VLAN tags). However, even with the Legacy technologies there are differences in the protocol stacks of the packets that belong to the specific network service instance. In some cases (e.g. MPLS L2VPN) there is a pair of labels where the inner label does not change in the packet along its path between the service endpoints. This label could be used to detect packets belonging to a particular service instance along their path. But there are as well technologies which use a single identifier and do the per-hop identifier swapping (e.g. circuit-cross-connect MPLS L2VPN). Such changeable identifiers make it impossible to map the packet which is captured on different locations along its path through the network to the specific service instance using only identifiers and without the interaction with network elements and getting the protocol related data from them. Also, there are some recent network virtualization technologies which do just the IP address translation and do not add anything into the packet protocol stack [23]. Any technology agnostic, but service-aware monitoring technology for fault localization has to overcome this problem and provide a solution which allows easy reconstruction to which service instance packets belong to, preferably without the interaction with network elements.

We propose the addition of the dedicated service-related fields into the packet padding field of the OWAMP probe (öwapping) packets (Fig. 4.3). Such change of the probe packet is not against the OWAMP specification which does not define the mandatory content of the padding field. Adding at least Service ID, as shown in Fig. 4.3 enables the association of the probe packet with the specific network service instance regardless of the underlying network technology. The minimum amount of information needed to encode in the öwapping packet is as follows:

- A 128-bit magic string known to the MAs and PCs which ease the packet parsing. In the testing phase the following string in ASCII was used: “NETMONviaOWAMPv1”, however, any other string can be used. PCs are parsing the packets and looking for this string to detect the position of the location of the relevant data. The length of the string guarantees low collision probability and low probability of false detection of the relevant data.
- Service ID - 32 bit string which uniquely identifies the network service being monitored
- Cor ID - IPv4 or v6 address of the Cor - the address to which the packet reports are being sent. Putting this IP address into the probe packets enables minimal (or no) configuration of the PC software, and no interaction with the Controller. It also enables simple Cor load balancing schemes which are being controlled from a central controller.

\[^4\]https://www.iovisor.org/technology/ebpf.
4.3. Detecting the same packet along its path through the networks

In order to calculate the transmission time of each packet between the two points in the network it is necessary to detect exactly the same packet in different places along packet path, at the boundaries of the monitoring zones. Service ID which is encoded into the packet indicates to which service instance the packet belongs to. Packets belonging to a single service instance can have different IP addresses (e.g. packets flowing in the opposite directions in case of a point-to-point service), but this information is not sufficient to detect a specific packet as there are multiple packets in the same flow. Since there are no fields in internet or transport packet headers which can be used for unique packet identification [24], if the users traffic is being analysed, one would have to compute some non-cryptographic hash (e.g. jenkins hash) of the headers and/or data payload of the packet in order to attempt the unique identification. Cases when two different packets have the same hash (collisions) are still possible and their probability depends on the chosen hash function. However in the case of the NetMon system which is capturing the OWAMP probe packets, the additional information that is needed is already contained in the probe packets: sequence number and timestamp recorded at the owning probe node (Fig. 4.3). This information uniquely identifies each packet in a flow.

Therefore the minimum set of data that is needed to uniquely identify the packet from a specific network service and which is being sent from the PC to the Cor contains:

- Service identifier - 32 bit Service ID
- Direction/flow identifier within the service instance: source and destination IP addresses
- 32 bit Packet identifier: packet sequence number from the probe packet or hash
- 64 bit timestamp of the packet recorded at each PC (used in OWAMP)

Using this approach the amount of data recorded per packet can be approximately as low as 96 bits (12 bytes): the size of the packet identifier and the local timestamp. The other data can be recorded once for multiple packets if the packets are being gathered in batches. If the information about the packets is being sent for each packet separately, then the amount of data recorded per packet is between 20 and 48 bytes for IPv4 and IPv6 respectively. It is important that the amount of data recorded for each packet is as low as possible because this information creates an overhead which has to be transported to the Cor. For a monitoring system which monitors n service instances which are being monitored at p zone borders where c packets are being analysed per second in each service instance, total minimal amount of application level packet information traffic towards the Cor is: \( T_{\text{Monitor}} = \frac{96 \text{ n} \cdot \text{pc bits/s}}{\text{s}}. \)

To illustrate the amount of traffic generated by the PC devices we will use an example. In case of the GEANT network if each NREN is a separate monitoring zone, there is 38 monitoring zones with \( p = 37 \) zone borders (all the NRENs are connected to the central GEANT domain in a star topology). Further, we assume that there are \( n = 10 \) instances of network services (VPNs) which are delivered to the GEANT users and a very demanding condition that all the service instances are multipoint having endpoints in all 37 NRENs. This means that each VPN has 666 pairs of endpoints. If each pair of endpoints in each service instance is probed with 100 packets every minute, then there are \( c = 1110 \) probe packets per second. The total amount of application level packet information traffic from all the PCs towards the Cor is 39.4 Mbit/s or around 1.06 Mbit/s per PC which is a small fraction of the total capacity of the links in the network.

With the very small monitoring zones (large \( p \)) or if the users traffic is being captured (large \( c \)) the amount of gathered data can be significant and the transfer of that data can present a challenge for the network and can make the price of the monitoring solution impractically high.

4.4. Time synchronization issues

Spatial performance degradation localization requires the detection of the same set of packets at the boundaries of the monitoring zones along their traversal through the network. The time it takes for a single packet to cross the network between the service endpoints consists at least of the time to cross all the network elements along the path and links between them (packet propagation time). Packet propagation time can be variable due to various congestions that might occur in network elements. Also packets can be dropped for various reasons (e.g. errors, full buffers). The goal of the per-zone performance verification is to accurately calculate propagation times and the number of lost packets in each zone and to detect the zone which contributed to the overall performance degradation. This requires time synchronization of packet capturing processes which associate a timestamp of the arrival with each packet and thus enable per-zone performance estimations. It also requires considerations about the length and synchronization of the time slots in which the packets are expected at each monitoring zone boundary and proper inferring when the packets are declared as lost.

In [11] authors proposed two methods for the detection of the packets along their traversal: based on a fixed number of packets that are detected, and the other is based on a fixed time slot in which the packets are observed. The latter was chosen as more deterministic, but on the other side it raise the problem of accurate detection of lost packets. This is illustrated by considering the network path depicted in Fig. 4.1, with 6 PC elements with finite propagation time between them, perfect time synchronization on PC elements and no other cross traffic which could affect packet interarrival times. Fig. 4.4 shows the time packets spent on each of the links monitored by PC elements, with the rightmost edge of each packet representing the arrival time of the whole packet which is timestamped.

We assume that all PC elements start capturing process at the same time (e.g. invoked by cron process), and that the capturing process lasts the same fixed amount of time (\( T \)). Due to the propagation time between the nodes only the first three PC elements (connected to R1, R3 and R4) will see all 6 packets which arrived in the same analysis period T. PC elements connected to R7 and R8 will miss the arrival of packet P6, and it can be falsely concluded this packet was lost inside the Zone 2. Also, PC element connected to the R10 will miss P5 as well, which could again lead to the conclusion that another packet was lost, this time in Zone 3. Furthermore, if the next analysis period comes immediately after the beginning of the previous one, PC elements connected to R7, R8 and R10 can see packets which didn't originate at the service end-point. The problem of synchronizing packet capturing elements and inferring what happened with the packets is even more complex in the realistic case with the imperfect time synchronization in the PC elements in the network, and the additional latency that packets have on some of the links due to the cross traffic impact. The analysis of the packets should be a during an extended time interval \( T + t \), where \( t \) is estimated as a time which is sufficient to include the maximal clock skews on different PC machines and maximum expected propagation and queuing latency. NetMon solution for packet analysis timing issues is given in Section 5.

4.5. Performance verification in federated environments

Packet capturing and deep packet inspection even if done in a single domain can raise suspicion of the network service users
that the content is eavesdropped and the user’s privacy compromised. If the system is used in a federated environment, and if as described in the previous sections single central Cor is used, this would mean that some domains would have to export packet information to a foreign domain which hosts the Cor. Such an approach could make some domains reluctant to deploy the system if they do not fully control the packet capturing and analysis processes. Although capturing only probe packets which is fully under the control of each domain (filters can be set up on the network elements) does not compromise user’s traffic, and the data sent to the Cor (Section 4.3) does not contain any personal information, NetMon could be configured to use one Cor element per domain. Cor in each domain would gather the packet information from its own domain, and only the measured performance data (and no raw timestamps) from the domain would be sent to the central MRR and shared with the other domains. If monitoring zones correspond to the domains in federated environment, such an approach allows easy detection of the domain which contributed to the performance degradation.

5. Fault localization algorithm

The NetMon Correlator continuously receives the reports about the captured packets from all PCs as they are captured along the paths in use. However, the performance information is not calculated per packet, but periodically with period T in order to provide the performance metric calculation at the desired time granularity. The choice of the period T impacts the amount of overhead traffic that the system creates and has to handle, as discussed in Section 4.3. Small periods T provide finer-grained measurements, but more numerous sampling sets, and thus create a larger networking and computational overhead as more information is exchanged, processed and displayed. The choice of this sampling period determines the overall accuracy of the system and can be adapted to teach user specific needs.

Per zone performance measurement and fault detection algorithm operates as follows:

- For each endpoint pair (or flow) in a service instance a separate packet matrix M is created. M stores the packet timestamps and identifiers from the reports received from the PCs in the period [T_{current}, T_{current} + T + t]. T_{current} is the time at which the monitoring process starts for a given flow. T_{current} and the next period are updated following an event-driven model described later, to reflect the current monitoring period departure bound. Timestamps are based on the system clock of the PCs which are time-synchronized with the Cor. Regular NTP synchronization turned out to be reliable enough for the latencies that were observed in the network environment where the system was tested (Europe-wide network as described in Section 6); the maximum possible error was strictly lower than the minimal per segment real latency.

M has p rows, where p is the number of PCs in the network. The time t denotes an additional watchdog period required to provide accurate per-zone performance metric calculations in the situation of transmission and inter-component (AMQP) queuing delays. The number of columns c, c > 1, is a parameter that is indirectly configurable by the monitoring system administrator by choosing t. A packet which is sent during a given T period is not necessarily seen by all PC in the same period due to the packet transmission delays as discussed in Section 4.4. These delays cannot be neglected for the packets sent close to the end of each period T. c also depends on the time required to pass the reports from PC to the Cor through the NetMon AMQP message system which is used for inter-component communication. The time to pass the reports to the Cor depend amongst other parameters on the distance between each PC and Cor. Reports with identifiers and timestamps are not necessarily received by the Cor in the incrementing timestamp order even for a single given PC. Therefore the timestamps carried in reports cannot be used solely for the decision whether the group of reports should be processed and correctly shift the matrix (updating T_{current}). An out of order sequence of delivered reports could cause the Matrix being shifted too early without receiving a delayed report. Hence, the latencies of both packets that are analysed and reports with timestamps should be estimated and taken into account. These two latencies: packet transmission and inter-component queuing, are used as an additional period, denoted t, such that c - (T + t)/T. To resume, the choice of the watchdog period t determines c and mostly depends on the total transmission time of the packets end-to-end and should be at least as large as this maximum latency increased by the worst expected queuing additional delay introduced by our system. As a rule of thumb, with T = 1 s and t = 2 s (and therefore c = 3), the system gave reliable and accurate results for the virtual networks that were set-up Europe-wide as it will be described in Section 6.

- Upon the event of receiving a report with a packet whose timestamp is greater than T_{current} + T + t, Cor searches the whole matrix for all the packet identifiers which were sent in [T_{current}, T_{current} + T]. If the packet has passed through the network properly, the set of rows (PCs) in the ascending timestamp order for a given packet identifier i is the path of the
The processing function is driven by the event of receiving the packet which is outside the matrix and the algorithm uses the parameter next_chunk to specify the time Tcurrent of the next period: this parameter provides the integer expression (with T as radix) of Tcurrent to possibly handle the next group of reports. The group of reports which consists of all packet reports all over the matrix which are indexed by an identifier present in the current sender bin (i.e., matrix[report.sender][next_chunk]) is called a chunk. If the timestamp of the report, report.ts, falls within the matrix overall period ([Tcurrent, Tcurrent + T + 1] or, in its logical matrix expression, slot_diff < nbslot, lines 16–19), then the Cor simply stores the report (both in time, packet_bin, and in space, report.pc) and continue. Otherwise, the timestamp is outside the margins of the matrix (slot_diff >= nbslot) and, before recording the report (its identifier and timestamp in particular), more processing is necessary for extracting and calculating the performance metrics and then slide the matrix.

Four key steps are processed potentially multiple times, in the loop given at line 19:

1. The current Chunk Extraction (line 20): all packets identified in the bin of the sender depicted by next_chunk are extracted from the matrix (to form a chunk of packets reports);
2. The per-flow Path Election (line 21): comparing the path order of each stored packet report belonging to the chunk extracted previously;
3. The Path and per-zone performance metric calculation (line 22);
4. Transmit performance statistics in a structured JSON format to the MRR (line 23).

6. NetMon implementation and testing

NetMon implementation was successfully tested in the GÉANT GTS [25] testbed located in 6 European PODs which are located in London, Bratislava, Paris, Milan, Hamburg and Prague respectively, as shown in Fig. 6.1. The testbed consists of 12 virtual Ju-
Fig. 6.1. Testbed for NetMon verification.

Fig. 6.2. End-to-end service performance monitoring.
niper vMX routers with JUNOS 17, two in each city, which allowed the creation of multiple L2 and L3 VPNs between the customers (CPE) devices in an environment with real inter-device latency. The topology has redundant links between the routers which enabled testing the operation of the system in cases of various topology changes. Also, the underlying testbed architecture allows setting the traffic parameters like latency, latency variation or loss on the links between the vMX routers using the tc\(^5\) tool which is again useful for testing the operation in cases when some of the links in the network are faulty or experiencing various issues. Other components in the topology (MA, PC and CPE) were virtual machines with the Ubuntu operating system with 1 virtual CPU and 2GB of RAM. Other components of the system (MC, MRR, Cor) are virtual machines located in Prague POD with the same configuration. The propagation delay of the PC reports towards the Cor were approximately equal to the time it took those packets to traverse the path between the respective POD and Prague POD in an uncongested network, as the out-of-band connectivity between the PCs and Prague POD was uncongested.

Each domain in the testbed is a separate autonomous system. The testbed had three different virtual networks configured: one multipoint L3VPN between all 4 CPE devices, one multipoint L2VPN between CPE1, CPE2 and CPE3 and one point-to-point L2VPN between the CPE2 and CPE4. All the VPNs were configured as carrier-supporting-carrier VPNs with the central domain acting as a supporting carrier. MAs were connected to the same vMX routers in the same set of VPNs as respective CPE devices. PC devices were connected to the vMX routers, where mirroring of the traffic on the inter-router ports towards the PCs was configured. Total PC-to-Cor out-of-band overhead traffic bandwidth recorded on Cor at layer 2 from all the PC machines for a point to point L2VPN between the CPE2 and CPE4 (crossing all 6 PCs in both directions) was 264.96 Kbps. Multipoint VPNs required larger bandwidth as there were multiple endpoint pairs involved in the measurement. Overall all three VPNs had around 2Mpbs of overhead traffic. The processing load on the Cor was negligible in these circumstances.

6.1. Performance degradation detection

NetMon was used to monitor packet loss, latency and latency variation. All the performance measurements were done periodically with the 1 min period. The results are displayed in the dashboard (a matrix of a full mesh of measurements in a VPN), which shows the red-amber-green indicators of the status of the measured parameter and in per-parameter temporal graphs and tables which are described in the text below.

Figs. 6.2. and 6.3. show an example of how NetMon system makes spatial performance degradation and fault localization easy. Fig. 6.2. gives the end to end latency and latency variation graphs for a path in an MPLS L3VPN. From the graph it can be seen that the performance degradation (latency increased by about 20 ms)

\(^5\) tc (traffic control) - https://linux.die.net/man/8/tc.
happened around 16:51:30. Below the graph is the table with the per-span averages of the key performance parameters along the path of the packets between MA2 and MA4 devices. The values in the table are calculated as average values of all the measurements made in the time window shown in the graph above (from approx. 16:47:30 till 16:52) which can be set by the user by zooming into the specific time window of interest. By looking at the per-span data in the table, it can be seen that the latency parameter on a specific span which is affected by the degradation is going to start to increase. However, because the average value will slowly increase if the degradation persists for a longer period of time, it is probably more efficient for the operator to detect the change by observing the per-span temporal graphs. Fig. 6.3, shows how the same performance degradation was localized on the link between VMX2 and VMX3.

Besides the change of the latency using tc tool, NetMon was also successfully tested in cases of path changes. By shutting down the appropriate set of links in the testbed: VMX5-VMX7, VMX5-VMX6, VMX7-VMX8 and VMX8-VMX10, the service delivery and probe traffic between CPE2 and CPE3 were forced to take the sub-optimal path VMX5-VMX8-VMX6-VMX7 in domain 2 instead of the direct path VMX5-VMX7. With this path change the traffic was forced to cross twice more the Paris-Milan distance which is creating an obvious end-to-end increase in latency. The end-to-end graphs for latency and latency variation in this case are shown in Fig. 6.4. At the same time, Grafana displays per-segment results which include a different set of spans at different time windows before and after the path change.

NetMon was also implemented in the production GEANT network where several L2 and L3 MPLS VPNs were installed and monitored between GEANT PoPs in Milan, Hamburg, Vienna and London. In this setup NetMon has shown long term stability - periods longer than a month without a single lost measurement.

7. Conclusion

This paper presents the NetMon system which was created as an attempt to solve network service monitoring and performance verification problems in GEANT environment. It has a unique capability to do the scalable spatial fault and performance degradation localization in multi-tenant, multi-vendor and multi-network technology environment. By using a hybrid monitoring method which mixes the active probing and capturing this probe traffic it provides a good trade-off between the granularity of the results and the traffic overhead created by the monitoring system itself. Adopting the concept of monitoring zones the same approach can also be useful for some novel types of network services like those composed of chained service components where each service component can be monitored separately.

Next steps in the development of the NetMon system include adding support for new types of services and network technologies to the PCs and the support for the analysis of dedicated user flows which will enable even finer grained analysis of specific flows. The latter has to be followed with the additional analyses including resolving the privacy issues and a demand analysis for such a system because the cost of the monitoring system and the amount of generated overhead can potentially grow very fast.

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.

CRediT authorship contribution statement

Pavle Vuletić: Conceptualization, Methodology, Software, Supervision, Project administration, Writing - review & editing. Bartosz Bosak: Software, Validation, Writing - review & editing. Marinos Dimolianis: Investigation, Software, Validation, Visualization, Writing - review & editing. Pascal Méridol: Methodology, Investigation, Formal analysis, Data curation, Software, Validation, Writing - review & editing. David Schmitz: Methodology, Investigation, Software, Validation, Writing - review & editing. Henrik Wessing: Methodology, Investigation, Formal analysis, Writing - review & editing.

Acknowledgements

This work was supported by the GN4 - phase 2 project under GEANT 2020 Framework Partnership Agreement (FPA). The project is co-funded by the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2) and National Research and Education Networks in Europe, members of the GN4-2 consortium. Any opinions, findings, conclusions or recommendations expressed in this material are those of the
authors and do not necessarily reflect the views of the GN4 – phase 2 consortium or the European Commission.

Authors want to thank to all the members of the JRA2T1 and JRA2T3 tasks of the GN4 – phase 2 project who provided us a huge help and support with setting up the laboratory and GTS tested environments and especially Jerry Sobieski, JRA2 activity leader, for all the support, helpful discussions and valuable comments during our research.

References


Pavel Vuletic obtained his BSc, MSc and PhD in Computer Systems and Network Architecture from University of Belgrade, School of Electrical Engineering. He used to work on all layers from applications to network engine to the deputy director of AMRES, national research and education network, being responsible mainly for network development. He is currently associate professor at the University of Belgrade, School of Electrical Engineering at the Department of Computer Engineering and Information Technology, teaching Advanced Computer Networks, Data Security and Software Defined Networks courses. His-research interests span from network management principles and software defined networks to network and service performance and security.

Bartosz Rosak received his master’s degree in computer science in 2007 from Poznań University of Technology in Poland (Laboratory of IT Systems in Management). Since 2006 he has been working at the Application Department of Poznan Supercomputing and Networking centre as a systems analyst and developer. His-research interests include widely understood support for large scale computations on Grid, Cloud and HPC infrastructures, multiscale computing as well as system integration. He was a participant of a variety of European and national projects including BREIN, MAPPER, ComPat, GEANT and PLGrid.

Pascal Mérinod received his Ph.D. degree in computer science from the University of Strasbourg, France, in 2008. He then spent 2 years with the Université catholique de Louvain in Belgium, as a Post-Doctoral Researcher working in the TROILOY Project. He is now an Assistant Professor with the Network Research Group, iCube Laboratory, Strasbourg. He is involved in several european and national projects. His-main research topics are routing and Internet measurements.

David Schmitz received his Diploma in computer sciences from the Ludwig Maximilians University (LMU) Munich in 2002. Since that time he works at Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities as a researcher and software developer in different areas, e.g. networks and systems management, security, SDN, IT service management, service modelling, fault management, multi-layer network monitoring and visualization. He received his Ph.D from LMU in 2008 regarding service-oriented impact and recovery analysis.

Henrik Wessex completed his PhD study on electronic control of optical infrastructures at the Technical University of Denmark in 2006. While pursuing the PhD study Henrik Wessex participated in numerous European and national research projects. Since 2013 he has been involved in the European Infrastructure GÉANT projects (GN3, GN3plus, GN4-1, GN4-2) as taskmember or task leader. His research is focused on network performance from network technologies to service monitoring. In addition he is currently Head of Studies for B.Sc. Network Technology and it at DTU, in which position the gap between the students and the network performance monitoring research can be minimised.