Architectural Proposal for the Handling of Network Operations Data with Specific Focus on Virtualized Networks

by NGMN Alliance

Version:	**1.0**
Date:	**22 December 2017**
Document Type:	**Final Deliverable (approved)**
Confidentiality Class:	**P - Public**
Authorised Recipients: (for CR documents only)	

Project:	**Network Management & Orchestration Working Group**
Editor / Submitter:	**Jack Tottingham (Deutsche Telekom AG) & Alexander Sobania (Deutsche Telekom AG)**
Contributors:	**Marc Werner (Deutsche Telekom AG)** **Wolfgang Wölker (Deutsche Telekom AG)** **Gina Baikenycz (EE)** **Ian McArthur (EE)** **Frederic Desnoes (Orange)** **Lucila Jimenez (Telcel)** **Juan Carlos Vega Moreno (Telcel)** **Jose Luis Roman Rojas (Telcel)** **Bas Haakman (Tele2)** **Fredrik Lindgren (Tele2)** **Wolfgang Fleischer (Telekom Austria)** **Lyubomir Hadzhiivanov (Telekom Austria)**
Approved by / Date:	**NGMN Board, 22 December 2017**

Commercial Address:
ngmn Ltd.,
Großer Hasenpfad 30 • 60598 Frankfurt • Germany

Phone +49 69/9 07 49 98-04 • Fax +49 69/9 07 49 98-41

Registered Office:
ngmn Ltd.,
Reading Bridge House • George Street • Reading • Berkshire RG1 8LS • UK
Company registered in England and Wales n. 5932387, VAT Number: GB 918713901

Abstract

The solution architecture presented in this document proposes simplified principles for handling network operations data that can be applied on a large scale. In this white paper, the authors focus on obtaining access to network operations data for the purpose of network monitoring and other applications.

Document History

Date	Version	Author	Changes
16.06.2017	0.1 – 0.18	J. Tottingham	Draft versions
21.06.2017	0.19	J. Tottingham	Draft version for internal review
30.06.2017	0.20	J. Tottingham	Minor wording changes
20.07.2017	0.21	A. Sobania	Minor layout changes
14.08.2017	0.22	J. Tottingham	Incorporation of comments from NGMN NWMO Working Group
25.08.2017	0.22A	J. Tottingham	Clean version – comments and highlights removed. Materially identical to version 0.22
11.12.2017	0.23	J. Tottingham	'Security' added to introduction, section 5 and requirements
14.12.2017	0.24	J. Tottingham	Footnote 6 amended section 7.1
22.12.2017	1.0	K. Moschner	NGMN Board Approved Version

Contents

1 INTRODUCTION

Telco networks are moving more and more towards virtualized networks. While presenting enormous opportunities for efficiency and automation improvements, the adoption of virtualization[1] presents many new challenges. Amongst others, the absence of some physical network interfaces makes access to data[2] for operational purposes such as tracing and monitoring extremely challenging.

This paper proposes an architecture which offers several ways to capture and process network operations data in virtualized networks either directly from the virtual network function (VNF) or by means of an agent running on the VNF (see illustration 2). While not the explicit focus of this paper, both these approaches assume security requirements and standards are enforced during their implementation. Furthermore, by storing data in a way which makes it available for access from multiple locations and by multiple different types of application or analytics capability, the architecture overcomes another key barrier of legacy trace solutions.

Today's trace solutions impose vendor specific data formats and require mostly dedicated and proprietary hardware installation. Each specific solution tends to be designed to an operator's bespoke needs making it both expensive to develop and deploy and difficult to transfer, e.g. between applications or physical locations. The architecture[3] in this paper seeks to overcome such issues by enabling a common data capture, conversion and storage approach which, if adopted by the industry, will offer operators with more flexible and broader capabilities.

"Our aim is to realize an open architecture that will enable operators to develop and implement a future-proof and flexible tracing solution as well as analytics."

For the purposes of seeing this document implemented, the authors would recommend communication of this document to bodies such as 3GPP.

[1] While this document outlines an architecture based on virtualization, it is not the intended purpose of the authors to go into virtualization principles or benefits in detail.
[2] Throughout this document, the term ‚data' is used to refer to ‚network operations data' unless explicitly stated otherwise. In referring to 'network operations data' the authors are focusing, in the main, on control plane (CP) data.
[3] Although the focus of this paper is virtualized networks, the architectures can work equally well with legacy network solutions.

2 REFERENCES

[1] https://wiki.wireshark.org/Development/LibpcapFileFormat
[2] https://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html
[3] https://sites.google.com/site/h21lab/tools/tshark_elasticsearch
[4] https://www.elastic.co/
[5] https://loicpefferkorn.net/ipdecap/
[6] https://www.wireshark.org/
[7] https://www.elastic.co/products/x-pack/security
[8] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[9] https://github.com/Yelp/elastalert

3 TERMINOLOGY

Term	Definition
Network Operations Data	A data stream flow which passes over a digital network. Usually this data stream consists of protocols which are standardized, e.g. signalling. For the purposes of this document Network Operations Data shall cover control plane only.
Raw data	Data which has not undergone any form of transformation or aggregation and still contains the base network operations data in its purest form. This is not to say the format of the data may not have changed.
Metadata	Data which is used to describe data. It is used to define and capture characteristics and properties of the data. Here, 'time stamping' is considered metadata.

4 LIST OF ABBREVIATIONS

API	Application Programming Interface
CDR	Call Detail Record
CP	Control Plane
ECMA	European Computer Manufacturers Association
IMS	IP Multimedia Subsystem
IP	Internet Protocol
JSON	JavaScript Object Notation
PCAP	Packet Capture
Rest	Representational State Transfer
RFC	Requests for Comments (see 'RFC Editor' part of Internet Society)
SCP	Secure Copy
SDM	Subscriber Data Management
SFTP	SSH File Transfer Protocol
SSH	Secure Shell
UP	User Plane
vEPC	Virtual Evolved Packet Core
VM	Virtual Machine
VNF	Virtual Network Function
VNFc	Virtual Network Function Component

5 CHALLENGES

With the advent of virtual networks and the increase of consumer data usage, operators face many challenges trying to keep their networks running in a cost-effective manner. Due to dynamic mapping between physical links and network interfaces, the traditional monitoring points become obsolete in a virtualized environment. Moreover, operators are keen to correlate network data provided by different sources to determine network analytics which are fundamental to operate their networks and services in a cost-effective manner.

Network equipment manufacturers have approached network virtualization in a siloed manner adopting tailored open source components which in turn means that operators are faced with having to have proprietary components in their network and therefore making capturing traffic in a virtual environment a challenge. In addition, virtualization allows operators to distribute their network in a targeted manner allowing them to provide additional capacity and functionality quickly and in the right place.

Furthermore, data speeds are increasing providing a variety of new and improved experiences for subscribers to use their mobile devices for activities such as video streaming, on-line shopping and banking. With increased activity, the volume of operational data is going to increase year on year and scalable solutions are required to handle this.

Operators need to be able to manage this traffic through their virtual and legacy[4] networks and provide the support teams with tools to be able to troubleshoot and maintain their network. The huge amount of data traversing telecommunication networks and the distribution of key network functions to wherever it is needed in the cloud will make monitoring and troubleshooting this new network difficult and costly for operators.

There are four areas to consider:

5.1 Capture

Different VNF solutions expose the data in a variety of formats and encrypt a number of interfaces which make it very difficult for operators to manage their network. The dynamic nature of virtualized networks means that operators need to change the way they deploy and configure traffic capturing capabilities in order to maximize the benefits of virtualization.

There needs to be consistency from equipment manufacturers to expose their interfaces in a standard format to enable operators to capture the traffic passing through their network[5].

5.2 Storage

Whether centralized or distributed, a storage layer is needed which makes raw data available to an analytics or application layer.

[4] This paper focuses only on virtualized networks and, for now, assumes legacy networks will, for the foreseeable future, continue to operate architectures along the lines of those in operation today.

[5] Being complex to deploy in a virtualized environment, the use of vTAPs or mirror ports are not considered here.

5.3 Accessibility

Openness is the key here and can be realized by both open source and standardization. Allowing multiple access points to a common source, i.e. the storage layer, enables many applications to use the same data. In making stored data available for analytics etc. operators need to take into consideration data protection, privacy and security in storage and usage and balance them with the operational needs of their network.

5.4 Security

Security is a key consideration for any telecommunications architecture. While this paper does not seek to specifically address detailed security requirements, it assumes security standards are applied across all layers of the solution outlined herein. Amongst others, requirements would include data being decrypted only when needed, e.g. during the capture phase at the NFV (see sections 5.1 and 7.1). Through the use of roles (see also chapter 10, section 4) control of access to functions and data can be maintained. Access to and retention of data should be limited to telecommunications regulations as issued by local authorities.

6 PRINCIPLES

6.1 General Principles

The key principle of the architecture is that it is independent of specific solutions. For the collection, storage and presentation of network operational data the architecture avoids siloed constructs which restrict the flexibility of the architecture, make it more complex to manage, limit cross application usage and limit the potential for automation.

The architecture maximizes opportunities offered by virtualization and cloudification, i.e. decoupled hardware and functions, scalability, efficiency of deployment and operation etc.

6.2 Layered Architecture

A layered network architecture (see illustration no. 1) overcomes silos stemming from specific solutions. Within each layer the available functions are mutually exclusive to ensure there is no duplication of functions between layers. Each layer therefore contains a uniquely segregated set of functions.

Likewise, data is not duplicated across layers. Data stored and handled in each layer is distinct from data in all other layers.

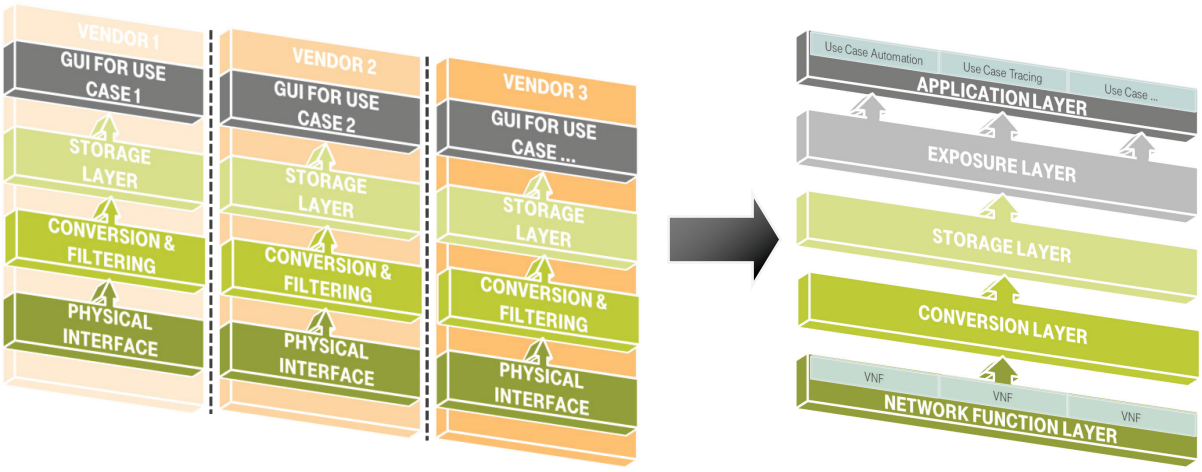

Illustration 1: From siloed to integrated architecture

6.3 Automation

Simplification of the architecture is essential in order to facilitate significant improvement in automation. Through simplification, automation potential across the scope of the entire architecture can be realized rather than being limited to specific parts of the architectural stack, as witnessed today.

6.4 Data Capture

Data is captured directly from the application or via means of a tracing agent running on a virtual machine. Sourcing of data at an infrastructure level is avoided. While a push mechanism to capture data is seen as most likely, the architecture does not exclude the option for operators to create a pull mechanism (see chapters 8.2.1 and 8.2.2)

For the purposes of clarity, the term 'data' within this paper and the illustration below is assumed to cover control plane data. Control plane data covers signaling, routing and such like data. The described architecture does not exclude user plane and is by its very nature more than capable of handling user plane data. Nevertheless, it is recognized that user plane data brings a number of additional challenges with it such as storage requirements to handle the volume, privacy, legal interception adherence etc.

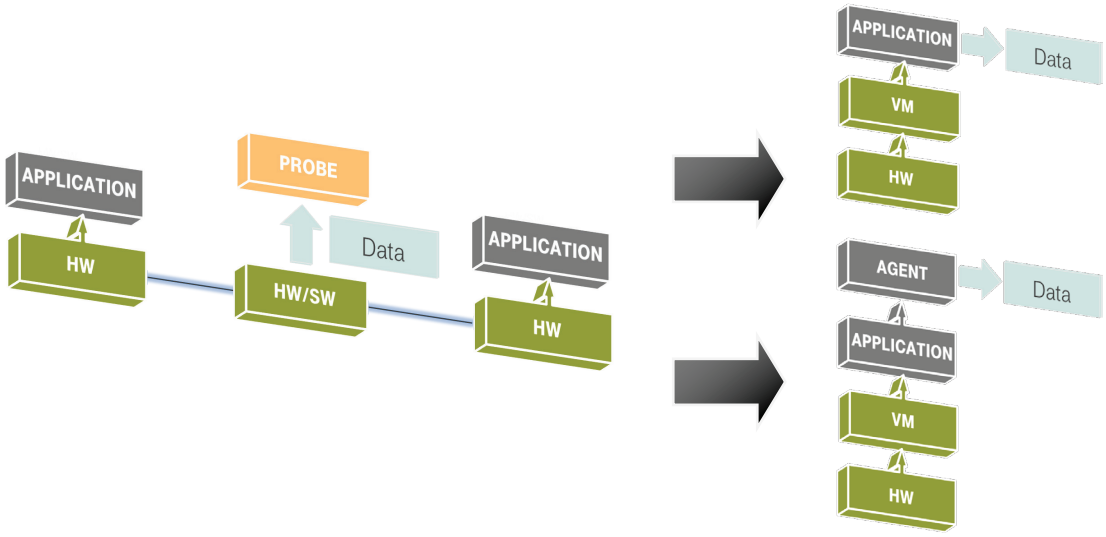

Illustration 2: Data accessibility methods for physical and virtualized network architectures

6.5 Data Integrity

The integrity of data and its 'raw' nature at the point of capture is to be preserved across the architecture. At no point is collected data modified, augmented or aggregated across the layers up to the storage layer. Metadata, e.g. time stamping can be associated with the raw network data providing they guarantee the integrity of the captured raw data.

Raw network data can be repackaged. For the purposes of storage and use by applications, data can be decrypted and reformatted. Such processes still preserve the integrity of the original raw data content.

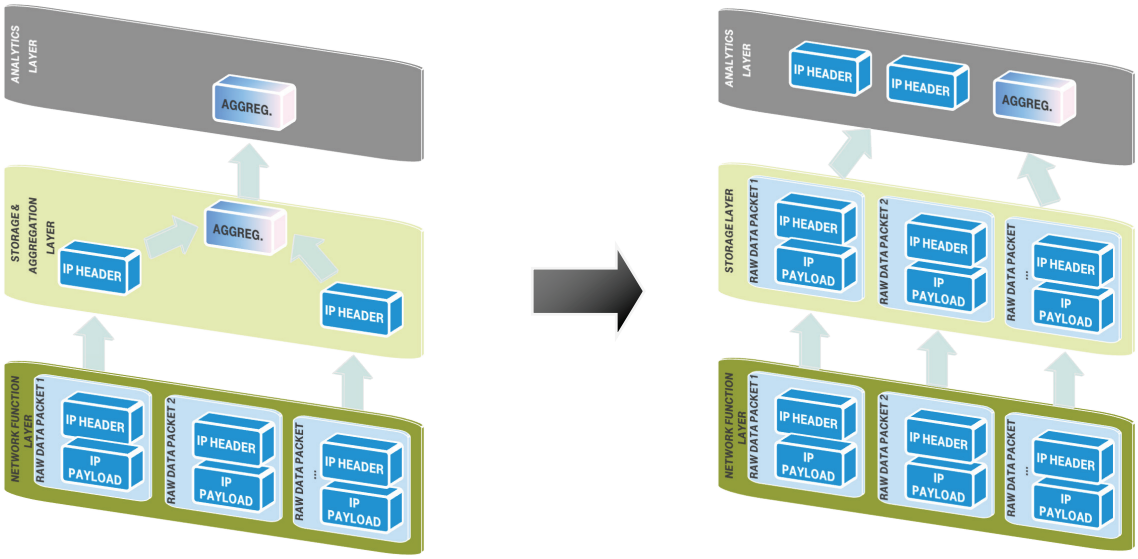

Illustration 3: Integrity of data maintained across architectural stack (no aggregation or other manipulation)

6.6 Scalability

The architecture stack is inherently scalable to handle changing demands for data capture, handling and storage. Scalability works horizontally in each layer of the stack and allows processes to run in parallel.

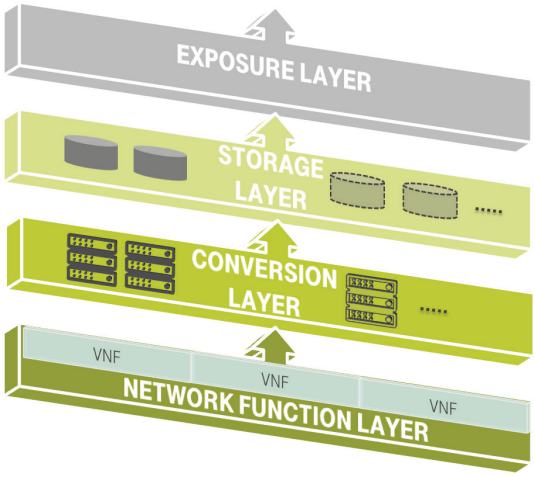

Illustration 4: Architecture benefitting from horizontally scalable architecture through virtualization/cloudification

6.7 Leveraging Data to Support Multiple Applications

The capture of data takes place through 'passive' means which should not affect network traffic and performance. Data capture occurs only once and is made available to different applications, e.g. analytics. All stored data can be used multiple times either simultaneously or at different times by the same or different applications.

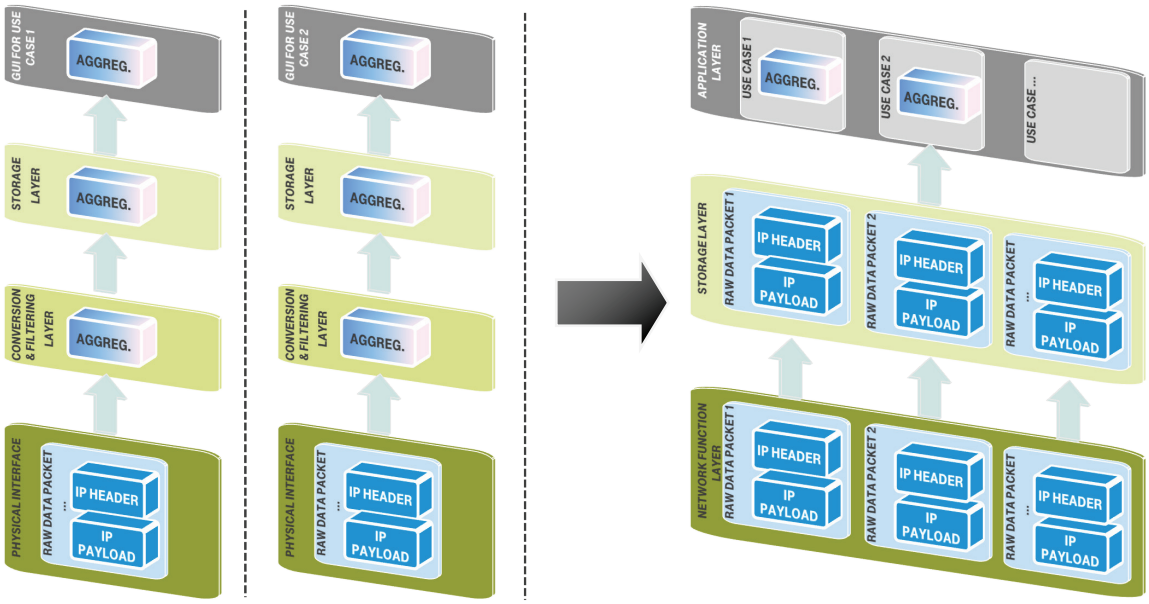

Illustration 5: Analytics running from a single storage layer instead of independent parallel silos

6.8 Data Analytics and Other Uses

Through the use of standard APIs, data is offered to analytics and other applications. APIs ensure analytical and other uses of the data can work across the entire storage layer.

7 ARCHITECTURE

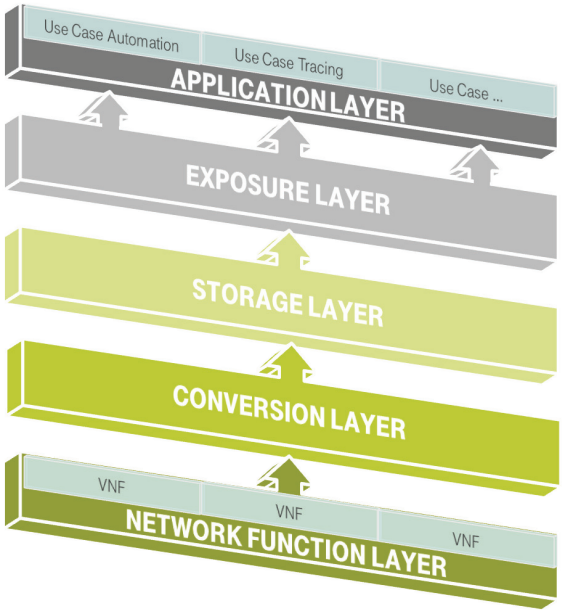

Illustration 6: Layered Architecture

7.1 Network Function Layer

The network function layer (see illustration 1) is where all network functions are performed. Any network function or other form of software running in this layer on a virtual machine (VM) or in containers makes its data available to a conversion layer (see below). Data to/from the virtual machine is provided either directly by a network function via an interface or by permitting the installation of agents to collect the data from the VM/container.

All network packets which leave or enter the virtual machine are provided up to the conversion layer in a raw format. Where data is encrypted[6], the network function decrypts it as part of the process. The focus is control plane data but other forms of data, e.g. user plane should not be excluded. On transfer and storage, data can still be in an encrypted state[7].

In order to ensure data is easily transferred, no modification or aggregation takes place. The original data remains in its unmodified state.

The capture of data is 'passive' via duplication. Any malfunction of the duplication process does not affect the production environment.

[6] Decryption of data and where it takes place is for each operator to decide based on its operational and legal requirements concerning data security and privacy. For decryption to occur, the key needs to be made available at the point of decryption. All keys should be stored and transmitted securely.

[7] For analytics to take place decryption would be required if the data is in an encrypted format.

7.2 Conversion Layer

In the conversion layer, data received from the 'network layer' is converted into a format suitable for storage. This process is done without any modification or aggregation of the original data. Metadata can be optionally added at this point.

The conversion layer is able to parse (and therefore understand) all relevant protocols including protocols which have been specifically developed for individual operators' tracing requirements.

Conversion is made possible using appropriate tools.

7.3 Storage Layer

The storage layer is where converted data is stored for subsequent analysis. An API is used to move data from the conversion layer to storage layer. Logically the data storage layer appears like a centralized system and supports parallel processing and horizontal scaling. Physically the data can be stored in a distributed or centralized manner.

For efficiency reasons, stored data is indexed.

7.4 Exposure Layer

The exposure layer makes stored data available to the application layer through dedicated APIs. Search and count mechanisms based on the indexed data are available for numerous applications.

7.5 Application Layer

Any application can make use of the available 'exposure layer' API for different purposes, e.g. analytics.

8 DETAILED EXAMPLES

8.1 Overview

This chapter gives some practical example implementations how the new architecture can be implemented by an operator given the principles described previously. The different examples do not extensively cover all requirements but serve to explain the architecture in further detail[8].

8.2 vEPC

The vEPC example assumes a virtualized EPC is implemented by an operator with both a push and pull model being illustrated. Other virtualized use cases like IMS, SDM would be implemented in a similar manner.

8.2.1 Push Model

a. The vEPC duplicates the traffic within the VNF instance, e.g. MME or tracing agent. The VNF takes a raw copy of the incoming IP packet destined for the function which the VNF is providing.
b. The vEPC encapsulates the duplicated traffic in IPinIP (RFC 2003) towards a capturing VNF instance. The capturing VNF instance is typically located close to the originator to keep time stamping as close to the source as possible.
c. The IPinIP encapsulation gets stripped of (for example by open source tool ipdecap, see reference [5]) and dissected and converted by Tshark into JSON format. (See reference [3]).
d. The duplicated dissected traffic in JSON format is injected via curl into Elasticsearch.
e. The traffic is exposed via a REST API to the applications.

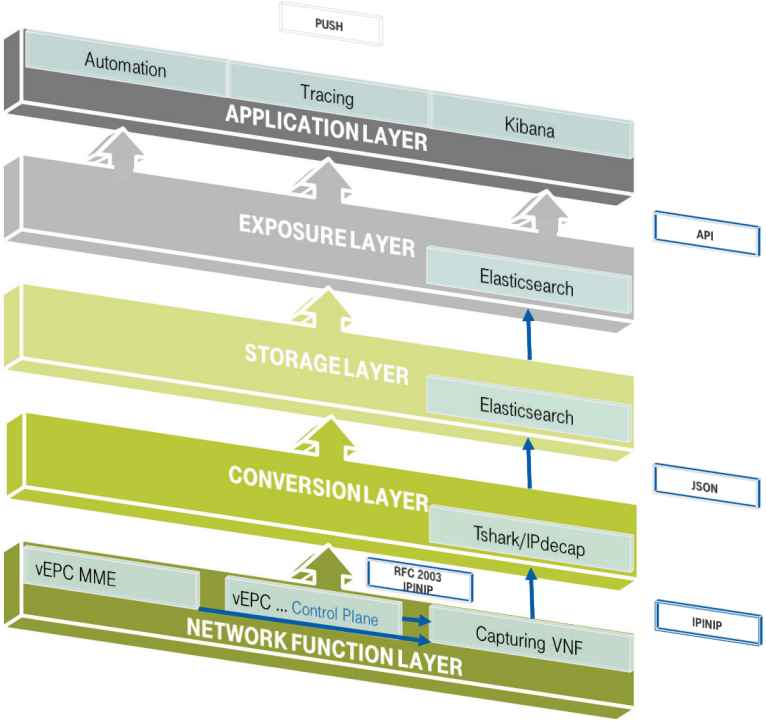

Illustration 7: vEPC data being 'pushed' to the conversion layer

[8] Listed tools and formats are merely exemplary and are not intended to suggest alternatives are not possible.

8.2.2 Pull Model

a. The vEPC duplicates the traffic within the VNF instance. The VNF takes a raw copy of the incoming IP packet destined for the function which the VNF is providing.
b. The capturing VNF collects the traffic from all different VNF via SCP or SFTP. The traffic is pulled by the conversion layer.
c. The conversion layer dissects the traffic via Tshark and converts it into JSON format (see reference [3]).
d. The duplicated dissected traffic in JSON format is injected via curl into Elasticsearch.
e. The traffic is exposed via a REST API to the applications.

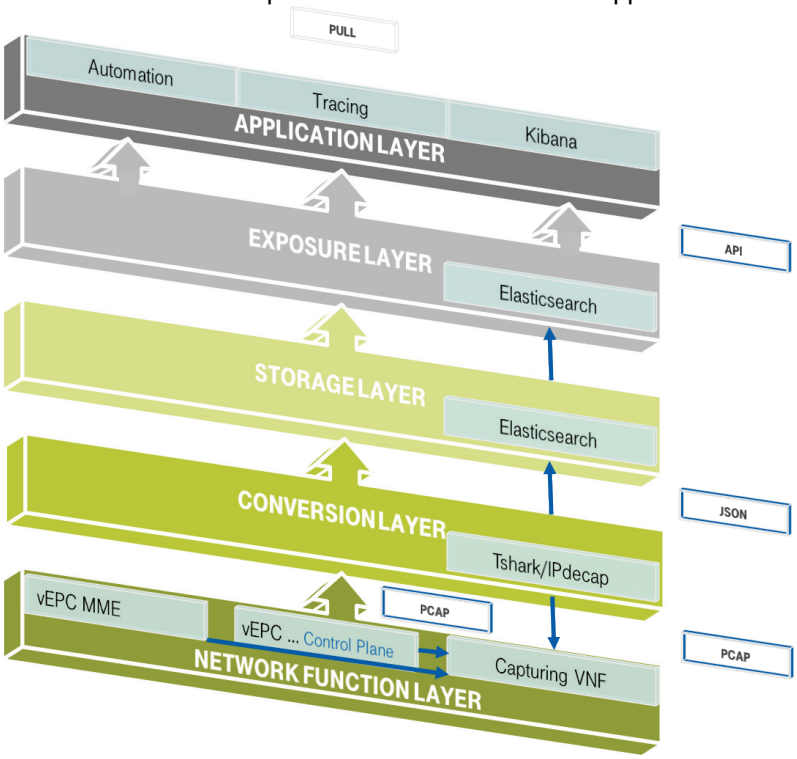

Illustration 8: vEPC data being 'pulled' by the conversion layer

8.3 Scalability

Scalability in the above examples is possible. Processing can be distributed across data centers and/or multiple VNF's. The Elasticsearch example is already distributed and very scalable by itself. Complete architectures to optimize the processing are out of scope for this paper but today's cloud architectures can provide all the scalability paradigms required.

8.4 Open Source Formats

PCAP or PCAPNG is used by the libpcap library developed by the tcpdump (a common packet capturing tool) team.

The PCAP format is an open source format which consists of:

a. Global header defining the file

b. Record (per packet) headers defining the packets including timestamps.
c. Raw packet data

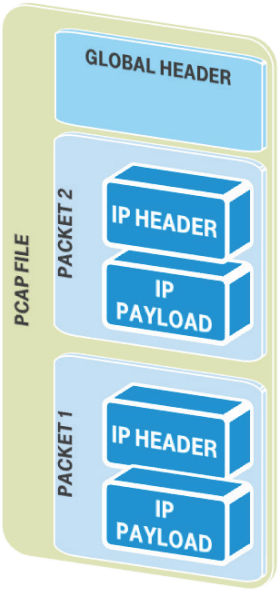

Illustration 9: Example of raw packet data in PCAP

8.5 IPinIP

The IPinIP protocol is very basic and only adds an outer IP header to the traffic. The outer IP header will make the encapsulated duplicated packet transportable to any IP based destination. See RFC2003 from IETF for details about IPinIP encapsulation.

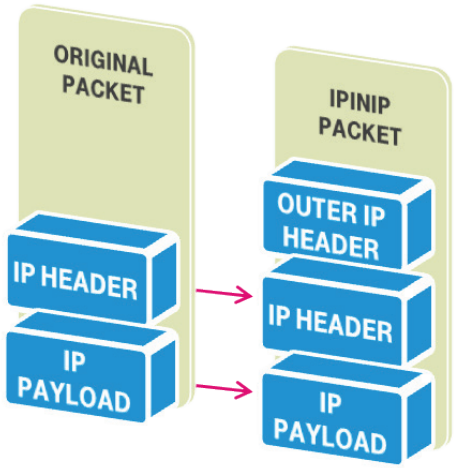

Illustration 10: IPinIP conversion process

8.6 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write and for machines to parse and generate. It is based on a subset of the JavaScript programming language. JSON is an ideal data-interchange language. ECMA publishes the JSON format (see reference [8]) and appendix.

8.7 Kibana

With the data stored in Elasticsearch, it can be searched and visualized according to specific requirements of the user. A common tool for this is Kibana; a web application available from Elasticsearch that enables searching and visualization of Elasticsearch data. While it ships with several visualizations, it supports plugins enabling extensions of the user interface and its capabilities.

Its main function is to graph data stored in Elasticsearch in the form of dashboards and reports. However, it supports displaying data in a tabular form as well which is a better fit in an interactive fault-tracing scenario. An example of this can be seen in illustration 11.

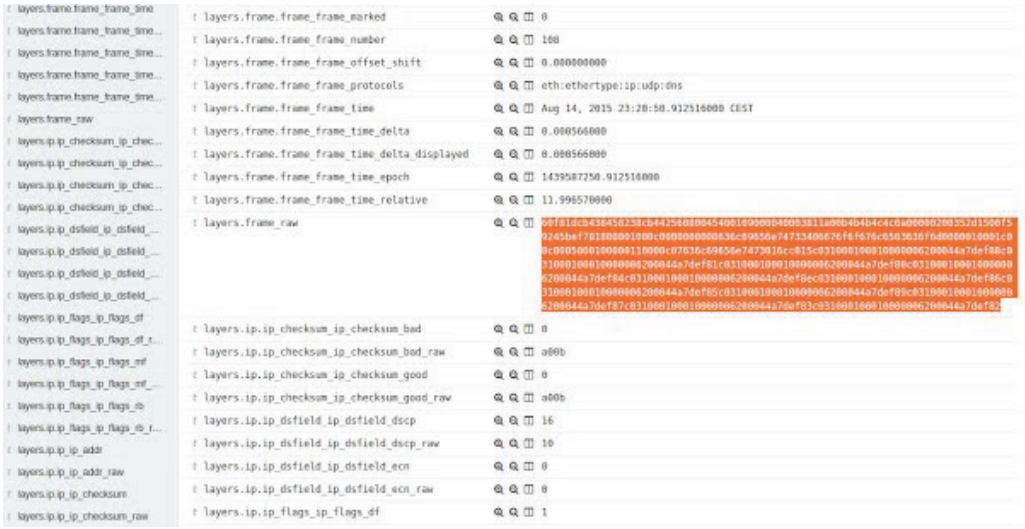

Illustration 11: Viewing decoded and filtered packet data. See reference [3]

A more “at a glance” style dashboard example can be seen in illustration 12. Dashboards will automatically refresh the data at a configurable interval.

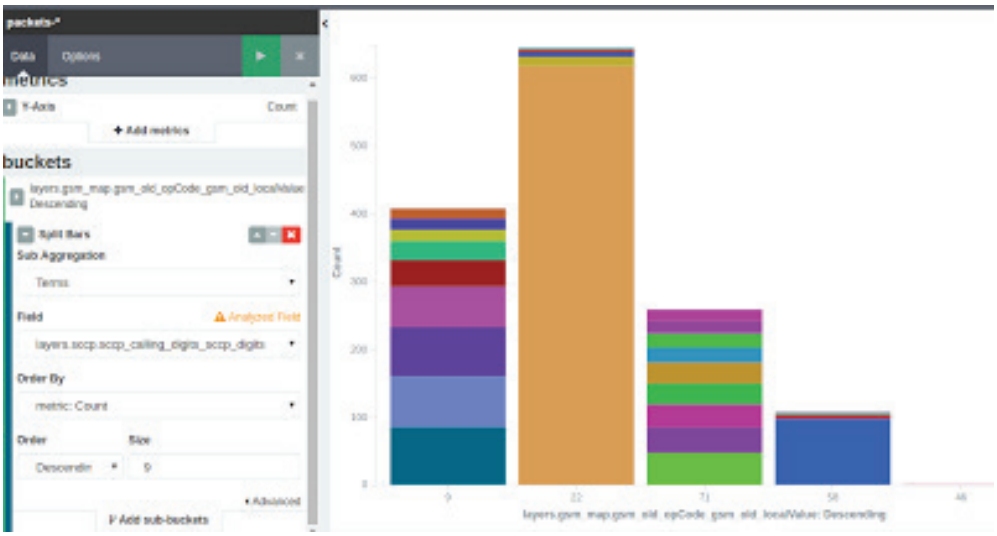

Illustration 12: Kibana visualization showing GSM MAP split by Operation Codes and Calling GT. The data was obtained by decoding the packets using Tshark as described in this document. See reference [3]

Depending on the use-case, creating custom visualization-plugins for Kibana could be useful. These could be created in-house or provided from a vendor. Plugins can be made to adapt the Kibana user interface to better fit a specific need or display search results in a use-case specific manner.

In addition, to user facing interfaces like Kibana there are alternatives for automated alerting based on data obtained by querying Elasticsearch. There is the alerting module in the X-Pack available from Elastic as well as Open Source alternatives such as Elastalert. See reference [9]

9 COMPARATIVE OVERVIEW

Using 'Integration and Testing' and 'Tracing and Root Cause Analysis' as example use cases, the following section demonstrates the differences of the architecture when compared to today's architecture. Note: the examples have been drawn from mobile network processes but apply equally for fixed line networks. The benefits of 'tracing and root' cause analysis apply equally to 'integration and testing'.

Current Architecture	***Proposed New Architecture***
Access to data is granted using dedicated interfaces.	Open APIs across the stack enable access to data using different tools and methods.
Data is stored selectively for each individual tracing solution, thus preventing its use by other applications/analytics tools in support of different use cases.	Data is stored in a single, logical storage layer supporting access to data via multiple applications/analytics tools thereby supporting a wide range of use cases.
The structure of the stored data in the storage layer is determined by each solution thus resulting in many different structures even for the same use case.	Data can be structured generically and independently of vendors thereby enabling it to support multiple use cases.
A multitude of different data storage solutions makes integration between platforms complex, costly and difficult to automate.	A single architecture solution for storage provides single point of entry and source for all applications/analytics tools thus supporting simplification and automation.
Numerous logical and physical activities prevent operators from conducting the desired full level of testing when new nodes are integrated into the network. The silo effect generated by dedicated solutions limits the extent to which full end-to-end testing can be performed.	End-to-end testing is facilitated by an architecture which by definition eliminates silos and thereby allows a greater range of logical and physical activities, e.g. testing, to be performed.
Existing solutions offer limited opportunity for automation and machine learning beyond the scope of their own systems. For example, today's solutions offer limited capabilities to detect new nodes and determine their impact on the network.	In a virtualized world the ability to identify new nodes (virtual machines) in an automated way becomes critical in order to reduce reliance on human intervention. The integrated nature of the new architecture allows more use of machine learning, e.g. through the automatic detection of new nodes.
Visualization of data is limited to vendor specific solutions.	Analytics and visualization of data can be performed by any tool thus reducing dependency on vendors and enabling a broader range of analytical and visualization tools to be used.

10 REQUIREMENTS[9]

Reference Number	Layer	Requirements
1.01	Network Layer	Network traffic data shall be duplicated and made available to a conversion layer
1.02	Network Layer	Network signaling/control plane data shall be made available from network sources
1.03	Network Layer	Network sources shall include virtual network functions (VNFs) and other sources
1.04	Network Layer	User plane data could be provided from network sources
1.05	Network Layer	An intermediate network storage facility shall support the push of data to analytics tools (push mode)
1.06	Network Layer	An intermediate network storage facility shall support the pull of data by analytics tools (pull mode)
1.07	Network Layer	In push mode, duplicated network data shall be made available instantaneously
1.08	Network Layer	In push or pull mode, duplicated network data shall be continuous, i.e. it shall not be interrupted or delayed prior to being made available to other layers
1.09	Network Layer	During the duplication of network data, the network layer shall remove all forms of encryption
1.1	Network Layer	The network layer shall deliver duplicated data in a clearly defined and readable format
1.11	Network Layer	All signaling data shall be exposed to the conversion layer
1.12	Network Layer	All traffic data (signaling etc.) shall be presented in its raw state such that no transformation has been conducted
1.13	Network Layer	All data shall be presented in an open, non proprietary and license free format, e.g. PCAP, PCAPNG
1.14	Network Layer	The network layer shall provide the ability to disable/enable push/pull modes in order to stop/start data provision by the VNF or agent running on the VNF
1.15	Network Layer	The network layer shall provide an API to configure how push/pull modes are handled

[9] Requirements outlined here are assumed to adhere to local, EU and other telecommunications laws.

2.01	Conversion Layer	The conversion layer shall be able to collect data by means of 'pull' from presented data sources
2.02	Conversion Layer	The conversion layer shall be able to receive data by means of 'push' from presented data sources
2.03	Conversion Layer	The conversion layer shall support all common data formats
2.04	Conversion Layer	The conversion layer shall be scalable (upwards & downwards) to meet changes in volumes
2.05	Conversion Layer	The conversion layer shall support the addition of metadata to enhance and/or supplement stored data
2.06	Conversion Layer	The conversion layer shall not modify the content of the data it sends or receives
2.07	Conversion Layer	The conversion layer shall support conversion of data from one format to another
2.08	Conversion Layer	By default, the conversion layer shall not filter, i.e. exclude some data content
2.09	Conversion Layer	The conversion layer shall have the ability to parse/decode data (incl. use of own tools)
2.1	Conversion Layer	The conversion layer shall be able to differentiate the granularity of parsed and decoded data
2.11	Conversion Layer	The conversion layer shall deliver parsed data in an intermediate, open and non-proprietary format, e.g. JSON, XML, CSV, AVRO
2.12	Conversion Layer	The conversion layer shall support the definition of boundaries or thresholds in order to limit the amount of data ingested by the conversion layer and thereby avoid overload of systems
3.01	Storage Layer	The storage layer shall support full horizontal scalability in order to increase or decrease storage and performance
3.02	Storage Layer	The storage layer shall support redundancy and robustness
3.03	Storage Layer	It shall be possible to access data from more than one point of entry.
3.04	Storage Layer	It shall be possible for multiple simultaneous accesses to data to be made
3.05	Storage Layer	The data shall be stored in an efficient way

3.06	Storage Layer	Data shall be accessible instantaneously
3.07	Storage Layer	It shall be possible to manage and housekeep data, e.g. delete obsolete data
3.08	Storage Layer	Shall provide the option to store data in encrypted format where regulatory or security stipulations deem it necessary
4.01	Exposure Layer	The exposure layer shall provide the single point of access to the storage layer
4.02	Exposure Layer	In the exposure layer, it shall be possible to anonymize or mask (such that it cannot be viewed by certain users) data
4.03	Exposure Layer	The exposure layer shall provide multiple access towards the application layer
4.04	Exposure Layer	The exposure layer shall provide authentication and authorization for access to data
4.05	Exposure Layer	The exposure layer shall support output of data in multiple formats
4.06	Exposure Layer	The exposure layer shall support the push or pull of data to the application layer
4.07	Exposure Layer	The exposure layer shall support the filtering of data
4.08	Exposure Layer	The exposure layer shall not be able to delete or modify stored data
4.09	Exposure Layer	The exposure layer shall be able to manage traffic/bandwidth across the network
4.10	Exposure Layer	The exposure layer shall support the setting of boundaries which determine what data is output or how queries run

11 CONCLUSION

The architecture presented here offers a new opportunity to access network operations data in virtualized environments. Through the provision of a duplication function, operators are able to efficiently capture data from virtualized interfaces without negatively impacting operational systems. The layered approach to capture, conversion, storage and exposure of data provides flexibility and scalability. The interface at the exposure layer ensures operators can apply numerous applications and analytics capabilities.

Through this approach, operators can expect a simplification of monitoring, tracing and other tasks. The impact on human resources performing manual monitoring activities is expected to reduce as the architecture will support a greater application of automation from the storage/exposure layers and a broader spectrum of analytical capabilities.

12 APPENDIX

Example TSHARK JSON output – DISSECTED GTP-C in JSON format:

```
    # json top-level protocol filter
    ./tshark -T json -J ip -r siqtran.pcap
    [
 {
  "_index": "packets-2017-02-14",
  "_type": "pcap_file",
  "_score": null,
  "_source": {
   "layers": {
    "frame": {
     "filtered": "frame"
    },
    "sll": {
     "filtered": "sll"
    },
    "ip": {
     "ip.version": "4",
     "ip.hdr_len": "20",
     "ip.dsfield": "0x00000000",
     "ip.dsfield_tree": {
      "ip.dsfield.dscp": "0",
      "ip.dsfield.ecn": "0"
     },
     "ip.len": "276",
     "ip.id": "0x00000000",
     "ip.flags": "0x00000000",
     "ip.flags_tree": {
      "ip.flags.rb": "0",
      "ip.flags.df": "0",
      "ip.flags.mf": "0"
     },
     "ip.frag_offset": "0",
     "ip.ttl": "64",
     "ip.proto": "132",
     "ip.checksum": "0x00007ad6",
     "ip.checksum.status": "2",
     "ip.src": "127.0.0.1",
     "ip.addr": "127.0.0.1",
     "ip.src_host": "127.0.0.1",
     "ip.host": "127.0.0.1",
     "ip.dst": "127.0.0.1",
     "ip.addr": "127.0.0.1",
     "ip.dst_host": "127.0.0.1",
     "ip.host": "127.0.0.1",
     "Source GeoIP: Unknown": "",
     "Destination GeoIP: Unknown": ""
    },
    "sctp": {
     "filtered": "sctp"
    },
    "m3ua": {
     "filtered": "m3ua"
    },
    "sccp": {
     "filtered": "sccp"
    },
    "tcap": {
     "filtered": "tcap"
    },
    "gsm_map": {
     "filtered": "gsm_map"
    },
    "gsm_sms": {
     "filtered": "gsm_sms"
    }
   }
  }
 }
]
```

Illustration 13: JSON format output. See reference [3]