“Quality of Service Assesment Using Machine Learning Techniques for the NETCONF Protocol”

Javier A. Ouret
Faculty of Engineering
Department of Informatics Engineering
Universidad Católica Argentina
Buenos Aires, Argentina
javierouret@uca.edu.ar

Ignacio Parravicini
Faculty of Engineering
Department of Informatics Engineering
Universidad Católica Argentina
Buenos Aires, Argentina
ignacioparravicini@uca.edu.ar

***Abstract*—Study of an unsupervised machine learning approach for the testing results defined by the RFC2544 - ITU.Y1564 standard methodologies and the use of NETCONF protocol to automatically assess traffic parameters required to comply with quality of service level agreements. By doing disruptive and non-disruptive tests for service integrity, a service provider can certify that the working parameters of a delivered Ethernet circuit complies with the end user expectations, to avoid poor application performance. This work focus in an unsupervised learning approach using Expectation Maximization based clustering algorithm. We find that the unsupervised technique used is an excellent tool for exploring and classify service parameters like frame delay, frame delay variation, packet high loss intervals, availability and throughput. A correlation of parameters with the type of service required for the network flows (real time IP for data, video and voice applications) can be applied to automatically set bandwidth profiles. The bandwidth profiles can be configured per port, VLAN and CoS based, in one or multiple EVCs (Ethernet Virtual Circuits) per UNI device port. For the setup we adopt the Yang data modeling language and XML NETCONF message encoding protocol, followed by a delayed or an optional non-delayed orchestrated activation in the network devices via multiple NETCONF transactions.**

***Keywords*—netconf, expectation maximization, performance monitoring, RFC2544, Y.1564, machine learning, service level agreement,**

I. Introduction

The exponential growth of the internet is making inefficient several protocols and methods originally designed to monitor and configure the network devices (e.g. carrier ethernet EDDs, NIDs, switches an routers). With this work is intended to improve: a) the evaluation of service level agreements in course for operative access links inside Layer 2 service provider networks (Carrier or Metro Ethernet), b) the configuration process among devices from different vendors. In this paper the feasibility of automate the process to get metrics using machine learning techniques is tested. In particular the focus is set only in 2 parameters for testing purposes: delay and delay variation. These are critical parameters during the configuration of bandwidth profiles for real time traffic.

Regarding the method to gather information from operational networks it pretends to be as automatic as possible, thus we will use machine learning for the selected parameters. Machine learning is considered a field of artificial intelligence that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) from data [1]. In order to make the learning process self sustained we need to use techniques that can process information from the network links with minimum external human intervention.

To evaluate and get the necessary data for this work the Expectation Maximization (EM) algorithm was adopted. It will help with the clusterization of the available data from operational links [2]. The data is presented for evaluation as soft clustering (clusters may overlap). The model is of mixture type, for each cluster we say to have a probabilistic distribution and the parameters (to be learn or discovered) are obviously unknown (mean μ and variance σ). For the delay d and jitter j (delay variation) a model with K distributions is acceptable for this first stage of testing.

One benefit of EM is that it naturally produces valid parameters for the mixture distribution on every iteration [3].

II. NETCONF Concepts and scope

NETCONF is an IETF network management protocol defining a simple mechanism through which a device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated [4]. NETCONF is a connection-oriented protocol, as it stablishes an authenticated session and maintains the order of the transmissions. YANG is a data modeling language to define the structure, syntax and semantics of data that can be used for NETCONF operations, including configuration, state data, RPCs and notifications. This allows the device to expose a full, formal Application Programming Interface (API). Applications can use this API to send and receive full and partial configuration data sets. NETCONF uses an Extensible Markup Language (XML) based data encoding for the configuration data as well as the protocol messages. NETCONF allows the user to perform CRUD[1] operations on the configuration of network devices, using XML data encoding as the protocol data transfer container (although since launch, JSON[2] support has been added). Besides the CRUD operations, which are implemented in a RPC-Based communications paradigm, NETCONF implements event notifications that can be setup in order to receive alerts from the Network Element, such as the ones usually sent using

[1] Create, Retrieve, Update and Delete
[2] Javascript Object Notation

SNMP. Unlike SNMP, there are no MIBs, instead the capabilities of the devices are exchanged when the connection is stablished [4].

NETCONF improves functionality, add more features, and increases efficiency to the network configuration process. Furthermore it provides single transactions for complex configuration data, add security embedded in the transport protocol, and is easier to develop new services than with CLI commands and SNMP.

Fig.1: a NETCONF client request can be executed in the following way using XML API:

```
<?xml version="1.0"?>
<nc:rpc message-id="16"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.net2edge.com/firmwos:6.0.1:if_manager">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <configure>
        <__XML__MODE__exec_configure>
        <interface>
          <ethernet>
            <interface>1/1</interface>
              <__XML__MODE_if-ethernet>
                <__XML__MODE_if-eth-base>
                  <description>
                    <desc_line>Netconf testing</desc_line>
                  </description>
                </__XML__MODE_if-eth-base>
              </__XML__MODE_if-ethernet>
          </ethernet>
        </interface>
        </__XML__MODE__exec_configure>
      </configure>
    </nc:config>
  </nc:edit-config>
</nc:rpc>]]>]]>
```

Fig. 1 NETCONF XML API Message

With all these functions, CRUD Operations (in addition to the Real Time Notifications), make the network elements more dynamic and flexible, making their configurations capable of being modified by any third-party software, based on any criteria, which might include configuration changes based on studies from big data and machine learning techniques.

III. Normalized metrics for QoS and SLA

Several metrics for Quality of Service assessment exist. Some of them follows recommendations from standard organizations or RFCs. In this paper we focus in the metrics from the Y.1731, the RFC2544 and the ITU Y.1564, normalized by the MEF (Metro Ethernet Forum) [5][6] [7][8].

The metrics for quality of service presented in Tables I and II are indicators that allow the characterization of links at the layer 2 level through MAC MEPs of UNI and NNI devices and evaluate the limitations thereof to determine the level of compliance of the operator with respect to user expectations.

TABLE I. MEF Normalized metrics [9].

METRIC	**UNI_A-UNI_Z**	**UNI_A-UNI_Z**	Time Interval **(T)**	**Small Time Interval Δ**	**Percentile**
FD (ms)	≤ 120	≤ 160	24h	*2m*	93th
FDR (ms)	≤ 30	≤ 30	24h	*2m*	93th
MFD (ms)	—	—	—	—	NA
IFDV (ms)	≤ 6	≤ 6	24h	*2m*	82th
FLR (% of frames)	≤ 0.12	≤ 0.12	24h	*2m*	NA
Availability (% of time)	≥ 99.95	≥ 99.95	24h	2m	NA
Resilience HLI (No. intervals in time period)	≤ 42	≤ 42	24h	2m	NA
C Resilience (CHLI) (No. intervals in time period)	≤ 4	≤ 4	24h	1m	NA

TABLE II. Type of service suggested for verification [10].

Type of Service	**CoS ID**	**Bandwidth Profile EVC & per CoS ID**	**Service Performance**
Real Time IP- IP Telephony-IP Video			
A Quality	6,7	CIR > 0 EIR = 0	Latency < 5 ms Jitter < 1 ms Packet Loss. < 0.001%
Critical burst data applications críticos with low latency and low packet loss (like Cloud storage)			
B Quality	4,5	CIR > 0 EIR <= UNI Speed	Latency < 5 ms Jitter = N/S Packet Loss. < 0.01%
Burst data applications that requires Bandwidth assurance			
C Quality	3,4	CIR > 0 EIR <= UNI Speed	Latency < 15 ms Jitter = N/S Packet Loss. < 0.1%
Best effort services			
D Quality	0,1,2	CIR = 0 EIR = UNI Speed	Latency < 30 ms Jitter = N/S Packet Loss. < 0.5%

These metrics will be detected as group ***d1*** or ***d2*** by the EM (See IV).

IV. Proposed method to evaluate delay and delay variation

The Layer 2 metrics obtained using the implementation of the Y.1731 standard exceed the accuracy of traffic measurements made with ICMP, RTT, BB or BDP values in layer 3. To get these metrics values the IEEE 802.3ah OAM is used. The OAM protocol run on Ethernet MAC sublayer using MEPs and MIPs ID to identify link endpoints. The MEP represents the endpoint of the entity to be managed (in general, a switch or router port) and the MIP is an intermediate point. For a detailed explanation see [5]. Each MEP is identified by means of an integer numeric value associated with its MAC address and defines connections between all linear MEPs and MIPs (EPS) or ring-shaped MEPs (ERPS). The configuration of the MEPs must be carried out manually on each service demarcation device based on the methodology suggested by the MEF and implemented in the firmware of demarcation devices. For

each one of the MEP / MIP the OAM frames 0x8902 circulate and it is possible to trace all the connections (Link Trace) and generate test signal patterns of different sizes (blocks of 64 to 9600 bytes) with patterns of 0s, 1s or 10101010, sequenced or not. These test signals are used to get delay and delay variation metrics. Y.1731 and Y.1564 are non-disruptive service test methodologies and can be run with traffic in service, whilst RFC2544 is disruptive.

A. *Justification for the use of machine learning techniques.*

Internet Service Providers and Carriers can manage from hundreds to millions of layer 2 links with point-to-point, point-to-multipoint or multipoint-to-multipoint configurations. For metro-Ethernet be more scalable, the service provider can use MAC encapsulation schemes for frame forwarding at the edge. The quantity of MAC address in transit require the use of pseudowire architectures (MPLS-TP, MAC-in-MAC, etc.) like IEEE 802.1ah on Provider Backbone Bridges to support large numbers of service instances in network devices [11]. Each link can have several virtual EVCs (Ethernet Virtual Channels) circuits with multiple Service VLAN (S-VLAN) or Customer VLAN (C-VLAN), each one with its own bandwidth profiles for delay sensitive traffic. Thus, is necessary to evaluate specific metrics to decide if a bandwidth correction is going to be needed. The amount of data available for evaluation justifies the implementation of non-supervised data capture and evaluation, detecting clusters of acceptable and non-acceptable metrics. Clustering methods allow learning the mixture of parameters from data. EM is adopted for this task and to get the maximum likelihood estimation in a problem where the mean and variance of clusters of metrics are hidden.

B. *The Expectation Maximization Algorithm (EM)*

EM can be used to generate the best hypothesis for the distributional parameters of the selected MEF metrics, to maximize the probability that the data we are looking comes from K distributions, each one with a mean μ and variance σ^2. The aim is to input the collected metrics into EM to do soft clustering. Each cluster of metrics corresponds to a probability distribution and want to find its parameters (mean and variance). The EM algorithm allows to derive these parameters. Some delay (and delay variation) values can fit with the required SLA (e.g.: they belong to the ***d1*** group) and others don´t (e.g.: if they belong to group ***d2***) but membership is undefined in the available data set (see D.) As (μ_{d1},σ_{d1}^2), $(\mu_{d2,}\sigma_{d2}^2)$ are needed to guess the source of data and source is needed for (μ_{d1},σ_{d1}^2), $(\mu_{d2,}\sigma_{d2}^2)$ the EM algorithm helps to resolve this problem. This is known as a maximum likelihood estimation. Given some data, we compute the value of the parameters that best explains that data.

C. *Equations samples for the EM algorithm*

In this case study the test data is built from 2 clusters of delay values (***d1***, ***d2***) and 2 clusters of jitter values (***j1, j2***). The indicated equations are written for ***d1*** and ***d2*** but can be easily extended for ***j1*** and ***j2***. We initiate the EM with 2 randomly placed distributions (μ_{d1},σ_{d1}^2), $(\mu_{d2,}\sigma_{d2}^2)$. For each $P(d1|x_i)$ we evaluate if it comes from d1 cluster or not and we adjust (μ_{d1},σ_{d1}^2), $(\mu_{jd2,}\sigma_{d2}^2)$ to fit the points assigned to them.

$$Estimated\ \mu_{d1} = \mu_{d1} \cong \frac{\sum(x_i)}{n_{d1}} \quad (1)$$

$$Estimated\ \mu_{d2} = \mu_{jd2} \cong \frac{\sum(x_i)}{n_{d2}} \quad (2)$$

$$Estimated\ \sigma_{d1} = \sigma_{d1} \cong \frac{\sum(x_i - \mu_{d1}\sim)^2}{n_{d1}} \quad (3)$$

$$Estimated\ \sigma_{d2} = \sigma_{d2} \cong \frac{\sum(x_i - \mu_{d2}\sim)^2}{n_{d2}} \quad (4)$$

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \ .e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (5)$$

Probability that x_i belongs to delay group d1 (matching expected SLA delay metrics).

$$P(x_i|d1) = \frac{1}{\sqrt{2\pi\sigma_{d1}^2}} e^{-\frac{(x_i-\mu_{d1})^2}{2\sigma_{d1}^2}} \quad (6)$$

For each x_i try to figure out from which set of parameters (μ,σ^2) it came from: $((\mu_{d1},\sigma_{d1}^2)\ or\ (\mu_{d2,}\sigma_{d2}^2))$?.

A soft assignment is done (based in the probability and not in the value itself):

$$d1_i = P(d1|x_i) = \frac{P(x_i|d1)P(d1)}{P(x_i|d1)P(d1) + P(x_i|d2)P(d2)} \quad (7)$$

$$d2_i = P(d2|x_i) = 1 - d1_i \quad (8)$$

Then the (μ,σ^2) are re-estimated to fit the value assignments a bit better:

$$\mu_{d1} = \frac{d1_1x_1 + d1_2x_2 + \cdots + d1_nx_n}{d1_1 + d1_2 + \cdots + d1_n} \quad (9)$$

$$\sigma_{d1}^2 = \frac{d1(x_1 - \mu_{d1})^2 + d1(x_2 - \mu_{d1})^2 + \ldots + d1(x_n - \mu_{d1})^2}{d1_1 + d1_2 + \cdots + d1_n} \quad (10)$$

These are the points for the delay d2 cluster:

$$\mu_{d2} = \frac{d2_1x_1 + d2_2x_2 + \cdots + d2_nx_n}{d2_1 + d2 + \cdots + d2_n} \quad (11)$$

$$\sigma_{d2}^2 = \frac{d2(x_1 - \mu_{d2})^2 + d2(x_2 - \mu_{d2})^2 + \ldots + d2(x_n - \mu_{d2})^2}{d2_1 + d2_2 + \cdots + d2_n} \quad (12)$$

Priors:

$$P(d1) = \frac{d1_1 + d1_2 + \cdots + d1_n}{n} \quad (13)$$

$$P(d2) = 1 - P(d1) \quad (14)$$

Iteration continues until convergence and extract the mean and variance for each group of metrics. From here decide to which group need to be re-configured to fit better with the required SLA.

Sample results:

```
d1/d2 set 1: [μ=[ 2.036 54.479], σ=[ 0.069  0.435]
d1/d2 set 2: [μ=[ 4.29  79.968], σ=[[ 0.17   0.941]
```

Fig. 2 Sample result from EM algorithm

NETCONF is used for the recollection of data to feed the EM and for the device´s bandwidth profile reconfiguration.

D. *Configure and get delays and delay variations. MEP DM (Delay Measurement)*

Delay Measurement is done between MEPs. It can exist many MEPs in one group, but DM is done between two MEPs only as it is a flow point to multi-point function.

Both the one-way and the two-way delay + delay variation can be calculated based on the exchanged information between the MEPs.

```
(config)# mep 1 dm 1 dual flow multi interval 10
last-n
mep = mep command.
1 = mep ID.
dm = delay measurement.
1 = Priority in case of tagged OAM. In the MPLS
and EVC domain this is the COS-ID.
dual = Delay Measurement based on 1DM PDU
transmission.
flow =  The two way delay is calculated as round
trip symmetrical flow delay. The far end residence
time is subtracted.
multi = OAM PDU is transmitted with multicast MAC.
interval 10 = between PDU transmission in ms.
last-n = delay for average last N calculation min
value is 10.
```

Fig. 3 Device configuration command sample [12]

Fig. 3 shows the CLI command version for the delay measurement that will be converted into a NETCONF XML *configure* session. Fig. 4 shows the command to execute the test and the command to recall he results. All the delay and delay variation information are sent to a data storage for ulterior analysis with the EM algorithm, along with the identification of each device (IP), port, VLAN, EVC and bandwidth profile. Fig. 5 present the CLI version from Fig. 3 converted to XML input to NETCONF.

```
(config)# perf-mon interval dm 15
(config)# do show mep dm

MEP DM state is:
RxTime : Rx Timeout
RxErr : Rx Error
AvTot : Average delay Total
AvN : Average delay last N
Min : Min Delay value
Max : Max Delay value
AvVarT : Average delay Variation Total
AvVarN : Average delay Variation last N
MinVar : Min Delay Variation value
MaxVar : Max Delay Variation value
```

Fig. 4 Device configuration command sample [12]

Configuration Request

```
<?xml version="1.0"?>
<nc:rpc message-id="16"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.net2edge.com/firmwos:6.0.1:if_manager
">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <configure>
        <__XML__MODE__exec_configure>
        <mep>
          <mep-id> 1 </mep-id>
          <delay measurement> dm </delay measurement>
          <priority> 1 </priority>
          <flow> dual flow </flow>
          <OAM PDU> multi </OAM PDU>
          <interval> interval 10 </interval>
          <last n delay > 10  </last n delay>
        </mep>
        </__XML__MODE__exec_configure>
      </configure>
    </nc:config>
  </nc:edit-config>
</nc:rpc>]]>]]>
```

Fig. 5 NETCONF dm Configuration

E. *Testing platform*

For testing purposes the Testtool platform was used. The tool allows to simulate up to 1000 NETCONF devices and do scale testing. It is based on a core implementation of Netconf server from OpenDayLight ODL controller [13].

The EM algorithm was programmed using Python and tested with data obtained using the explained methodology. Data sources were Lab layer 2 links with traffic generated by RFC2544 and Y.1564 tools. To validate the parameters in a real environment we used Liberator 304, 306 and 4424 Carrier Ethernet NIDs from Net2Edge.

V. Conclusion

The methodology proposed in this paper is used for learning clusters of metrics values during the operation of Ethernet layer 2 links. The learned values are then used to decide if adjustment of the bandwidth profiles in the devices are needed and we use the NETCONF protocol to set the configuration. As result the human intervention to collect, analyze and select metric data is reduced. The reconfiguration process of bandwidth profiles receives the input from the EM model and an historical database for the study of the link layer traffic performance is generated. On-going and future work aim to increase the amount of data to learn, add more metrics to the EM and NETCONF protocol and continue with the development of the software configurator.

Acknowledgment

This paper is part of a 2 year research work under development that is partially supported by the Faculty of Engineering, Universidad Católica Argentina, Buenos Aires.

REFERENCES

[1] Samuel, Arthur. "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research and Development, 1959.

[2] Dempster, A.P.; Laird, N.M.; Rubin, D.B. "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society, Series B. 39, 1977, pp 1–38.

[3] Padhraic Smyth, “Mixture Models and the EM Algorithm”, Department of Computer Science, University of California, Irvine, 2017

[4] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, RFC 6241 “Network Configuration Protocol (NETCONF)” , IETF, 2011.

[5] Y.1731. “Operation, Administration And Maintenance OAM”, Functions and mechanisms for Ethernet based networks, ITU, 2006.

[6] S. Bradner,J. McQuaid, RFC 2544 “Benchmarking Methodology for Network Interconnect Devices”, IETF, 1999.
Available: https://tools.ietf.org/html/rfc2544.

[7] ITU Y.1564. Ethernet service activation test methodology. ITU 2016.

[8] Christopher Cullan, Understanding Carrier Ethernet Service Assurance. Part I & II. MEF. 2016.

[9] MEF Performance Monitoring Metrics.
Available:
https://wiki.mef.net/pages/viewpage.action?pageId=24710765

[10] MEF Technical Specification 6.2 “EVC Ethernet Services Definitions”, MEF, 2011.

[11] IEEE 802.1ah - Provider Backbone Bridges
Available: http://www.ieee802.org/1/pages/802.1ah.html

[12] Liberator 444 CLI Configuration Guide, Net2Edge Ltd, Rev.D, 2017.

[13] OpenDayLight
Available: https://www.opendaylight.org/