U.P.B. Sci. Bull., Series C, Vol. 76, Iss. 1, 2014 ISSN 2286 – 3540

PERFORMANCE VECTORS FOR DATA NETWORKS OBTAINED THROUGH STATISTICAL MEANS

Lucian LEAHU[1], Neil DAVIES[2], Dan Alexandru STOICHESCU[3]

The recent evolution of data networks which takes place both horizontally by geographic expansion and vertically by incorporating a considerable number of technologies creates the need of a uniform performance characterization. This need emerges also from the end-user perception and expectations which don't take into account the location or the access method, but are only dependent on the network response time. This paper introduces a set of concepts, having as central one the "quality degradation" concept, followed by a case study for the data acquisition system of the ATLAS experiment at CERN [1].

Keywords: performance, quality attenuation, data networks, response time

1. Introduction

Data networks have witnessed a significant growth with respect to both the geographical expansion and the number of technologies incorporated. Internet is the main driving factor for this process as the end-user's demand for access to the Internet grew continuously in terms of "how much", "where" and "when". Along with these, a key aspect which the end-user perceives as a performance score when comparing two network access experiences is the **response time** from the communicating peer.

If the first aspects: bandwidth, time-of-the-day and location access are usually known beforehand from e.g. a contract with an ISP (Internet Service Provider), the response time is an instantaneous measure and is an *emergent* property of the network itself. Given the number of possible causes for the response time, a uniform approach in describing the network performance is thus needed.

This paper proposes a set of concepts based on mathematical statistics aimed to uniformly describe the performance of a communicating system. A key concept related to performance introduced next is the "quality attenuation" which, for the particular case of data networks can be decomposed into a three

[1] PhD student, Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, Romania, E-mail: Lucian.Leahu@gmail.com.
[2] PhD, Eng., Predictable Network Solutions, UK, E-mail: Neil.Davies@pnsol.com
[3] Prof., Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, Romania, E-mail: stoich@elia.pub.ro

dimensional vector. We will then show how these concepts can be incorporated into a methodology for establishing the "quality" of a network and, going further, to create a performance-wise comparison of different network devices. As a practical example we will use the data network from the data acquisition system (TDAQ) of the ATLAS experiment at CERN [1], which is the author's research result as part of the ATLAS TDAQ Networking team [2].

In the following chapters we will introduce the general concepts grouped under the name "observational model" then we will show how to apply them for data networks, Ethernet in particular. In the end we will present a practical use of these concepts in the case of the ATLAS TDAQ network.

2. Concepts

For a system which relies on interactive communication between its elements there is a fundamental **cycle of behavior** which characterizes it. Examples are the **communication protocols**, which have phases exhibiting cycles of behavior, e.g. request-response protocols. Furthermore, viewing a system as a set of **resources**, each cycle the system has to perform is a "walk-through" a subset of those resources.

The cycles, due to their repetitive nature, constrain the emergent performance of the communication system and in order to understand it we decompose a cycle into so-called **interactions**: tasks and/or operations, including here potential resources utilization. Using a resource involves: gaining access to it, using it, releasing it, all operations which influence the executed task[4].

Performing a complete cycle is thus equivalent to "passing through" or "experiencing" all those interactions. From the point of view of the cyclic process two things can happen when an interaction takes place:

- the process is *delayed* with a time T
- the process *looses* all or part of its elements, e.g. bits of information. We refer to this as *failure.*

Ideally, a perfect interaction is one which has $T = 0$ and the probability of loosing something inside it also 0. We define the **Quality Attenuation (ΔQ)** attached to an interaction to be the deviation from this ideal situation, i.e. a positive value for T and a binary value for failure (YES/NO). Given the repetitive nature of cycles, ΔQ becomes a statistical measure described thus by a *distribution of the execution time* and a *probability of failure*.

We introduce next a mathematical support for ΔQ in order to encompass both the execution time and the failure to execute: the **improper CDF[5].**

[4] The tasks can also be concurrently performed, involving the existence of shared resources, which consequently results in a *non-deterministic response time* from these resources

[5] Cumulative Density Function

The definition of a *proper CDF* for a random variable X is given by:

$$F_X(x) = P(X \leq x), x \in (-\infty, +\infty) \tag{1}$$

with two properties:

$$\lim_{x \to -\infty} F_X(x) = 0 \tag{2}$$

$$\lim_{x \to +\infty} F_X(x) = 1 \tag{3}$$

An improper CDF differs in that it is defined only on the interval and that , with , as depicted in **Fig. 1.** An improper CDF is thus not required to meet the property in relation 3, allowing an amount of probability of failure: . **T_f** is called the failure time and represents the time limit mentioned above, e.g. a time-out.

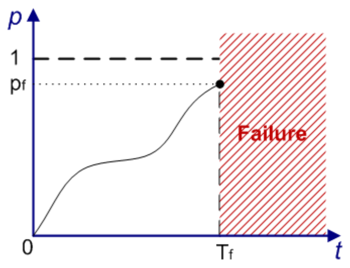

Fig. 1: Improper CDF

The performance of an entire cycle is thus given by the total accumulated ΔQ. In order to compose the ΔQ back from the individual interactions we need to make use of **distribution convolution** as we would do for the normal CDFs. For improper CDFs we can consider the rest of the probability mass to be towards +∞, thus enabling the use of convolution for these distributions in the interval of interest.

3. Observational model

Having defined ΔQ with mathematical support allows us to use it as a comparison metric. In order to do that, we need to measure it in a formalized framework, characterized by a set of concepts:

- an **outcome** is a task the system has to perform, to which performance measures are attached: time to complete and resource consumption. Outcomes can be viewed at different levels of abstraction, i.e. they can be decomposed or grouped together.

- **observables** are time-aware indicators in the system that are triggered by a particular task. We use observables to capture when a task has started or ended in order to measure its execution time. The way of choosing the observables reflects the level of abstraction the corresponding outcome is captured at.
- the **ΔQ (Quality Attenuation)**, introduced earlier.
- the **load** of a resource expresses the instantaneous demand on a resource or on a set of resources by all the tasks in the system. It influences the outcome's ΔQ by delivering a response time dependent on its load.

The formalization introduced earlier allows us to define the **goals** a system has to achieve and the method for establishing if they are met or not. We introduce two concepts for ΔQ:

- a *requirement (RΔQ)* is the desired ΔQ, i.e. the design criteria
- an e*xecution ΔQ (EΔQ*) is the ΔQ which is measured when the system is put into operation.

A system is considered to have delivered the required performance if all the outcomes have their EΔQs better than their RΔQs.

Fig. 2 exemplifies how the improper CDF for ΔQ can be used as a metric for a given task. The required execution time for the exemplified task is expressed by the RΔQ's CDF, stating that:

- 50% of the time the task has to finish earlier than T_0
- 90% of the time the task has to finish earlier than T_1

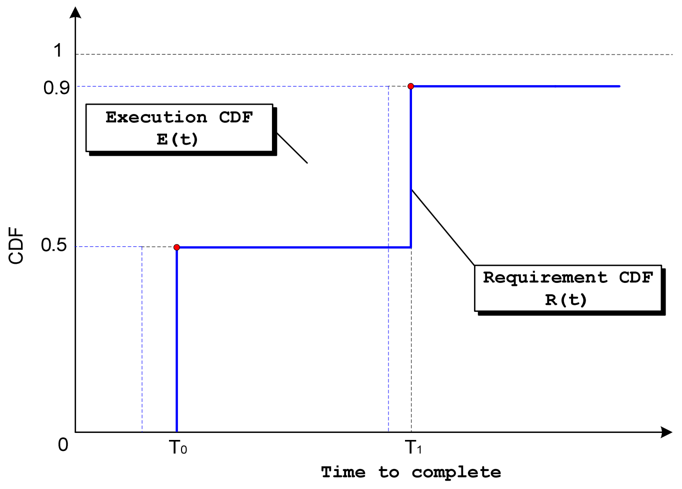

Fig. 2: Improper CDF as metric

The outcome meets the imposed requirements on the ΔQ if its execution CDF is always larger than its requirement CDF. Mathematically, the execution meets its requirement if the next relation is satisfied:

$$E(t) \geq R(t), \forall t > 0 \tag{4}$$

4. ΔQ in data networks

We will show how the concepts introduced above apply to a particular type of system, a data network. The tasks a data network has to fulfill is to transport packets between two points of the network. The associated observables are in this case the time-stamped traces captured on the network interfaces defining the end points.

Each network element - a resource in the system - adds delay and potential loss to the served packets. The ΔQ introduced in Section 2 can be split into components which fall into two distinct categories: **immutable** (can't be influenced, they depend on the system structure) and **mutable** (manageable, they depend on factors like the loading of the system). We introduce three components which build up the ΔQ:

1. **G** - *immutable* - is the contribution of the physical topology, being constant for a given network path. It represents the delay introduced by the network on a “zero length” packet, consisting of: transmission medium delay and access to the medium. G comes from *Geographical* or *Given*.
2. **S** - *immutable* - is the contribution of the mechanisms for processing packets, being constant for a given logical topology and packet size. It consists of serialization and de-serialization delays, inter-frame gap and any OSI Layer 2 introduced overheads. For a given network topology and configuration, S is dependent on packet size, hence its name.
3. **V** - *mutable* - is the contribution of the network load, dependent also on other factors like the time of the day, on the total delay. V comes from *Variable*.

The decomposition was done with the purpose of obtaining statistically independent components, i.e. any two random variables from the [G,S,V] tuple are *independent*. Hence, for a given path, the ΔQ is the convolution of the distributions above[6]:

$$\Delta Q = G \otimes S(size) \otimes V(load) \tag{5}$$

[6] For simplification we use term “convolution” between two distributions to express the convolution between the *Probability Density Functions* (PDF) attached to them

We use here the result which states that for a function of two random variables with $f(x)=x+y$ the PDF is the convolution of the individual PDFs for independent random variables, as demonstrated in [3].

Furthermore, we denote the convolution $G \otimes S(size)$ in equation 8 to be the **Structural Delay** for a network path, incorporating the *immutable* components only. The Structural Delay can be seen as the ΔQ obtained when the system is idle, i.e. the load is zero:

$$SD = G \otimes S(size) = \Delta Q \,|\, (load = 0) \tag{6}$$

The composition of the two variables cannot be calculated by convolution and has to be done on a case-by-case basis. In [3] it is mentioned that the sum of two random variables ε, η produces a new random variable ζ with the PDF given by:

$$w_{\zeta} = \int_{-\infty}^{+\infty} w_{\varepsilon\eta}(z-y, y)dy \tag{7}$$

and in the non-independent case:

$$w_{\varepsilon\eta}(x, y) \neq w_{\varepsilon}(x) w_{\eta}(y) \tag{8}$$

We will denote the general composition for S as:

$$S_{AB} = S_A \bullet S_B \tag{9}$$

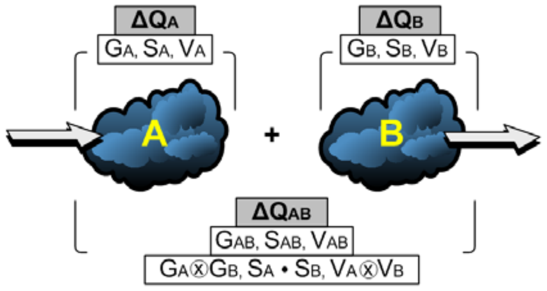

Fig. 3: Topology based convolution

Next, using the relation 7, we obtain the aggregate ΔQ:

$$\Delta Q_{AB} = G_{AB} \otimes S_{AB} \otimes V_{AB} \tag{10}$$

However, a common mistake in practice is to convolve the topological ΔQ's directly, i.e.

$$\Delta Q_{AB} = \Delta Q_A \otimes \Delta Q_B \tag{11}$$

For example, this is performed by measuring the delay of each sub-system and adding them in order to obtain the end-to-end delay. The relations 14 and 15 do not yield the same result because of the dependencies we've detailed above for S. A very important component is the one which can be managed and traded within the system: V. Using relations 7 and 8 we can obtain V by "subtracting" the structural delay from the ΔQ, denoted as:

$$V(load) = \Delta Q - SD \tag{12}$$

Mathematically, "subtracting" two known distributions involves a deterministic de-convolution.

5. Case study: ATLAS TDAQ Network

In this chapter we will show how the concepts above and the methodology introduced in [4] used to obtain G,S and V can be applied to a high-speed Ethernet network, implemented at CERN [1] for the Data Acquisition System of the ATLAS experiment.

The ATLAS TDAQ network was implemented gradually and its description can be found in [5] as of the stage in 2010. The architecture of the data network consists of two hierarchical layers:

- the *access/concentrator layer*, implemented using "pizza-box" switches
- the *core layer*, implemented using chassis switches

Before the implementation phase, an extensive research and testing of multi-vendor Ethernet network equipments was carried on in the ATLAS TDAQ Networking group [2]. This effort was performed in order to identify the network equipments who best meet the ATLAS TDAQ requirements mentioned in [6] by looking at performance indicators such as latency across the device and its sensitivity with the packet size. The method involved the use of dedicated network traffic generators with GPS clock synchronization for accurate time-stamping and was performed in laboratory conditions, i.e. no real traffic. See [7] for details.

Using the observational model techniques introduced earlier we will compare two switch types, added to the network in different phases of installation, from performance point of view. We will identify which of the two switches performs better.

All the involved switches are already installed in the ATLAS TDAQ Network and are running live traffic from physics experiments.

The measuring technique uses *ping* [8] Linux tool as ICMP [9] packets generator at node A with the destination as node B and *tcpdump* [10] Linux tool as capturing tool ran on both end-nodes simultaneously. The time-stamping required by the delay computation is thus offered by tcpdump itself.

The timing information in tcpdump comes from the *ioctl* SIOCGSTAMP system call[7] which allows us to get the time of the packet when it was received by the network interface card (and not by the application layer). The obtained captures are then processed by a tool written in Haskell [11] and provided by [12]. This tool obtains the *one-way delay* by employing a clock correction algorithm based on the big number of samples offered to it and assuming the linear model mentioned in [4], section 4.2.1.

For the first exercise we chose two pairs of observation points (named A and B) in order to measure the paths depicted in **Fig. 4**[8].

The paths separation was possible due to the dual network configuration of the SFI nodes and due to separate physical networks the Core, “Pizza-box” and XPU are belonging to: EF1 and EF2[8].

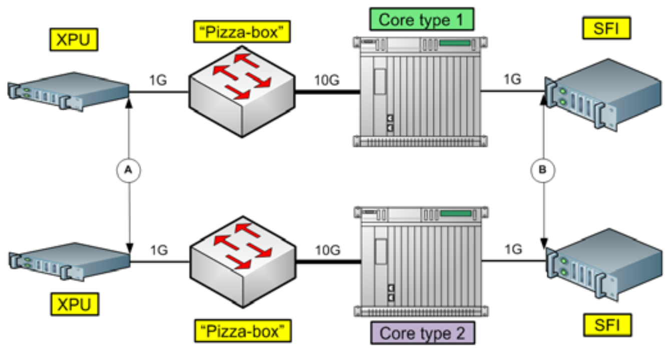

Fig. 4: Network paths for two different core switch types

The performance vector for a network is given by the [G,S,V] tuple. However, as V is dependent on the load existing in the network the relevant elements for comparison remain G and S, i.e. the Structural Delay[9]:

[7] It obtains the timestamp when the packet was received in the interrupt handler. The error margin here is how long it takes for the interrupt handler to occur, which is a few microseconds.

[8] The naming is done according to the network description in [5] and respects the ATLAS TDAQ elements’ roles in the system.

[9] The values are obtained from one “experiment” which means that the distributions become scalars

$$SD = G + S * framesize \tag{13}$$

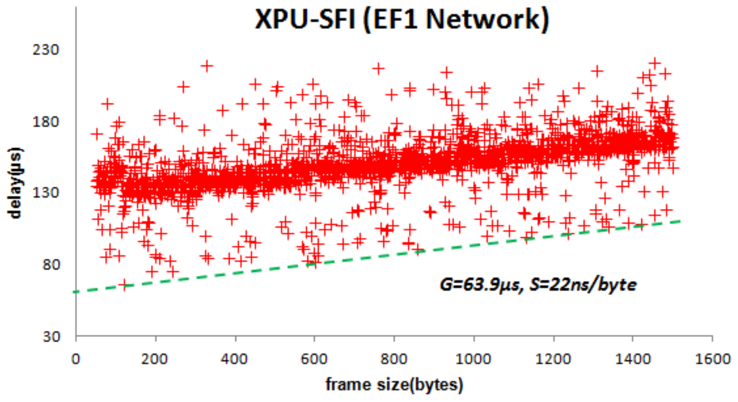

Fig. 5: Structural Delay for path 1 (XPU-SFI)

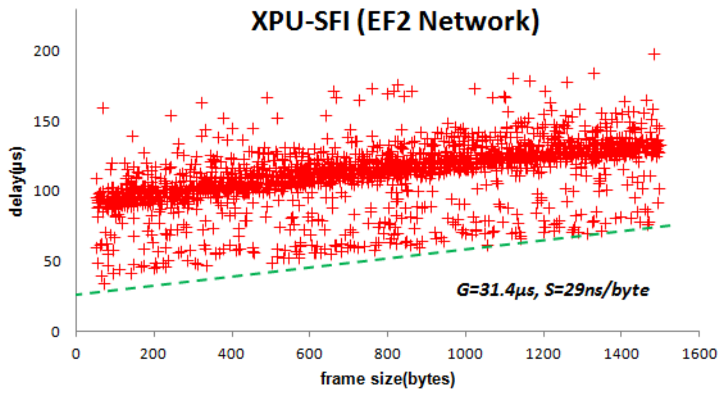

Fig. 6: Structural delay for path 2 (XPU-SFI)

The difference row shows:

- Core type 1 introduces a fixed 32µs additional delay compared to Core type 2
- Core type 2 introduces an additional 7ns/byte compared to Core type 1

If we compute the Structural Delay using relation 18 for both paths against Ethernet frame size and also the difference between the two we obtain the plots in **Fig. 7**.

The conclusion is that, although S is slightly better for Core type 1 the big difference in G is not cancelled even for large frames, hence for the entire

Ethernet frame size interval the second switch is better from the performance point of view. For the maximum frame size the Core type 2 is still faster with 20μs than Core type 1.

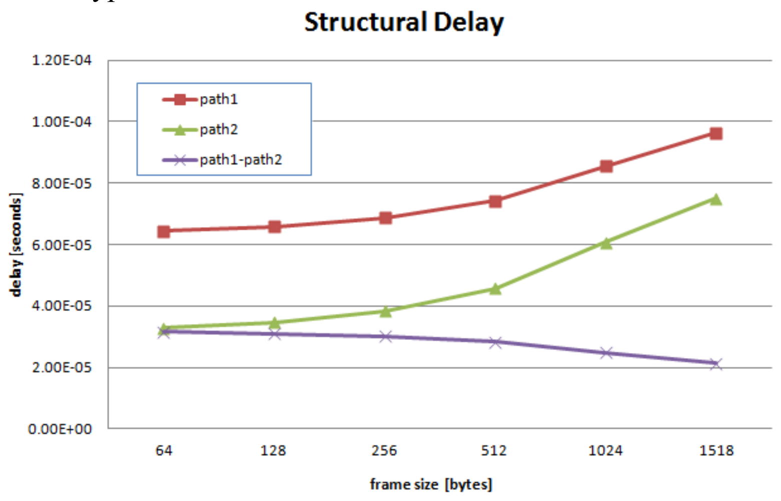

Fig. 7: Differential Structural Delay

The G and S obtained for both paths are presented in **Fig. 5** and **Fig. 6** and are extracted in the following table:

Table 1

Structural Delay for path 1 and path 2 (XPU-SFI)

Network / Difference	G(μs)	S(ns/byte)
EF1	63.9	22
EF2	31.4	29
EF1-EF2	32.5	-7

6. Conclusions

In this work we introduced an approach for describing networks with the purpose of understanding their end-to-end performance. We do this by adopting a top-down view of a communicating system which exhibits one important property: cyclic behavior. We continue with defining a framework of concepts: outcomes, observables from which a central concept called **quality attenuation (ΔQ)** is derived.

We showed that ΔQ can be used as a metric in order to be able to compare two systems from performance point of view, allowing us to decide which one performs better. The quality attenuation can also be used as design criteria by introducing the *aspiration and expectation ΔQ*. Furthermore, for the data networks we decompose ΔQ into three components statistically independent forming a vector [G,S,V]. We defined the basic operators and properties for them in order to re-compose the end-to-end ΔQ.

Using these concepts we exemplified how they can be applied in the case of a real network for a practical purpose: obtain G and S for two types of central switches. We confirmed that the newer generation type behaves better, as expected. We quantified the "difference in quality" for two generations of network devices using commodity PCs, open source tools and more importantly using the equipment in production environment.

As ideas as future research steps we identified:

- perform the same type of analysis for all the network devices in a network, potentially identifying those who cause bottlenecks
- infer on the loading of a network from the remaining component: V. Preliminary results show a good correlation between V and the loading factor of the network used as case study in section 5

In the context of the future upgrade of the ATLAS TDAQ system this methodology can be used to assist network devices upgrade decisions while the concepts introduced here can be employed for network re-design.

R E F E R E N C E S

[1] CERN, "The European Organization for Nuclear Research," CERN, [Online]. Available: http://www.cern.ch.

[2] S. Stancu, M. Ciobotaru, C. Meirosu, L. Leahu and B. Martin, "Networks for the ATLAS Trigger and Data Acquisition," in *Computing in High Energy and Nuclear Physics*, Mumbai, India, 2006.

[3] M. Ciuc and C. Vertan, "Perechi de variabile aleatoare," in *Prelucrarea Statistica a Semnalelor*, Bucharest, Editura MatrixRom, 2005, pp. 37-39.

[4] L. Leahu and N. Davies, "Observational model in ATLAS TDAQ Network," UPB PhD Report, Bucharest, 2010.

[5] L. Leahu and N. Davies, "Performance metrics in ATLAS TDAQ Network," UPB PhD Report, Bucharest, 2010.

[6] S. Stancu, M. Ciobotaru and D. Francis, "Relevant features for data-flow switches," CERN, Geneva, 2005.

[7] M. Ciobotaru, "Characterizing, managing and monitoring the networks for the ATLAS Data Acquisition System," PhD with "Politehnica" University, Bucharest, 2007.

[8] "Ping Linux Manual," [Online]. Available: http://linux.die.net/man/8/ping.

[9] J. Postel, "RFC 792," September 1981. [Online]. Available: http://tools.ietf.org/html/rfc792.

[10] "Tcpdump," [Online]. Available: http://www.tcpdump.org/.

[11] "Haskell," [Online]. Available: http://www.haskell.org/haskellwiki/Haskell.

[12] "Predictable Network Solutions Ltd.," [Online]. Available: http://www.pnsol.com/.

[13] R. Rajkumar, C. Lee, J. Lehoczky and D. Siewiorek, "A Resource Allocation Model for QoS

Management," in *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, 1997.

[14] C. W. Mercer, S. Savage and H. Tokuda, "Processor Capacity Reserves for Multimedia Operating Systems," in *Proceedings of the IEEE International Conference on Multimedia Computing*, 1994.

[15] R. Rajkumar, "Synchronization in Real-Time Systems: A Priority Inheritance Approach," Kluwer Academic Publishers, 1991.

[16] J. W. Layland and C. L. Liu, "Scheduling algorithms for multiprogramming in a hard real time environment," *Journal of ACM,* vol. 20, no. 1, pp. 46-61, 1973.

[17] "Ethereal," [Online]. Available: http://www.ethereal.com/.

[18] "Wireshark," [Online]. Available: http://www.wireshark.org/.

[19] R. W. Wolff, "Poisson Arrivals See Time Averages," *Operations Research,* vol. 30, no. 2, pp. 223-231, 1982.

[20] "802.1q - Virtual LANs," IEEE, [Online]. Available: http://www.ieee802.org/1/pages/802.1Q.html.