Editores
João Paulo Barraca
Pedro Salgueiro
Salvador Abreu
Vasco Pedro
Vitor Beires Nogueira

CRC 2015

Actas da
14ª Conferência sobre Redes de Computadores
Universidade de Évora
19 e 20 de Novembro de 2015

UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Actas da 14ª Conferência sobre Redes de Computadores — CRC2015
Escola de Ciências e Tecnologia
Universidade de Évora
2015

ISBN: 978-989-20-6237-2

http://crc2015.di.uevora.pt

Prefácio

A 14^a Conferência sobre Redes de Computadores decorreu na Universidade de Évora a 19 e 20 de Novembro de 2015, organizada pelo Departamento de Informática.

A CRC, desde o seu início em 1998, visa reunir a comunidade nacional de investigação e prática na área das redes de computadores, proporcionando um veículo para a apresentação e discussão dos trabalhos em curso, num ambiente informal e participado. As edições anteriores decorreram em Coimbra (1998, 2011), Évora (1999), Viseu (2000), Covilhã (2001), Faro (2002), Bragança (2003), Leiria (2004, 2013), Portalegre (2005), Oeiras (2009), Braga (2010) e Aveiro (2012).

Dos 16 trabalhos submetidos nesta edição, foram selecionados 13 para apresentação, tendo todos sido objeto dum processo de revisão conforme às melhores práticas da comunidade científica internacional. Os tópicos dos artigos apresentam uma boa diversidade, representando uma cobertura significativa da investigação que se produz em redes em Portugal. Muitos destes artigos são resultado de trabalhos realizados por alunos nas universidades, o que demonstra a vitalidade da área.

Queremos aproveitar para agradecer aos membros da comissão científica da CRC, pelo trabalho de revisão rigoroso e atempado. De igual forma, queremos agradecer aos autores que submeteram o seu trabalho mais atual, contribuições sem as quais não haveria lugar à conferência. A organização reconhece ainda o apoio institucional prestado pela Universidade de Évora, nomeadamente pela cedência do espaço e disponibilização de serviço de catering. De igual forma estamos reconhecidos a todos os que contribuiram para que a CRC 2015 tenha sido um sucesso!

Évora, Novembro de 2015

Pedro Salgueiro,
João Paulo Barraca,
Vasco Pedro,
Vitor Beires Nogueira,
Salvador Abreu

Comissões

Comissão Organizadora

João Paulo Barraca (Universidade de Aveiro)
Pedro Salgueiro (Universidade de Évora)
Salvador Abreu (Universidade de Évora)
Vasco Pedro (Universidade de Évora)
Vitor Beires Nogueira (Universidade de Évora)

Comissão Coordenadora

Alexandre Santos (Universidade do Minho)
Edmundo Monteiro (Universidade de Coimbra)
Teresa Vazão (Instituto Superior Técnico)
Rui Aguiar (Universidade de Aveiro)

Comissão de Programa

Adriano Moreira (Universidade do Minho)
Alexandre Santos (Universidade do Minho)
Amaro De Sousa (Universidade de Aveiro)
António Nogueira (Universidade de Aveiro)
Antonio D. Costa (Universidade do Minho)
Augusto Casaca (Instituto Superior Técnico)
Carlos Costa (ISCTE - Instituto Universitário de Lisboa)
Carlos Rabadão (Instituto Politécnico de Leiria)
Carlos Ribeiro (Instituto Superior Técnico)
Carlos Serodio (Universidade de Trás-os-Montes e Alto Douro)
Diogo Gomes (Universidade de Aveiro)
Edmundo Monteiro (Universidade de Coimbra)
Fernando Boavida (Universidade de Coimbra)
Gabriela Schütz (Universidade do Algarve)
Janio Monteiro (Universidade do Algarve)
João Paulo Barraca (Universidade de Aveiro)
Joaquim Ferreira (Universidade de Aveiro)
Joaquim Macedo (Universidade do Minho)
Jorge Sa Silva (Universidade de Coimbra)
José Legatheaux Martins (Universidade Nova de Lisboa)
José Luis Oliveira (Universidade de Aveiro)
Luis Bernardo (Universidade Nova de Lisboa)
Luis Lino Ferreira (Instituto Politécnico do Porto)
Manuel Ricardo (Universidade do Porto)
Maria João Nicolau (Universidade do Minho)
Marilia Curado (University of Coimbra)
Noelia Correia (Universidade do Algarve)
Nuno Preguiça (Universidade Nova de Lisboa)
Osvaldo Santos (Instituto Politécnico de Castelo Branco)
Paulo Carvalho (Universidade do Minho)
Paulo Pedreiras (Universidade de Aveiro)
Paulo Pereira (Instituto Superior Técnico)

Paulo Salvador (Universidade de Aveiro)

Paulo Simoes (Universidade de Coimbra)

Pedro Gonçalves (Universidade de Aveiro)

Pedro Rodrigues (Instituto Politecnico de Braganca)

Pedro Salgueiro (Universidade de Évora)

Pedro Sousa (Universidade do Minho)

Rogério Reis (Universidade do Porto)

Rui Aguiar (Universidade de Aveiro)

Rui Lopes (Instituto Politécnico de Bragança)

Rui Miguel Silva (Instituto Politécnico de Beja)

Solange Rito Lima (Universidade do Minho)

Salvador Abreu (Universidade de Évora)

Susana Sargento (Universidade de Aveiro)

Vasco Pedro (Universidade de Évora)

Vitor Basto-Fernandes (Instituto Politécnico de Leiria)

Vitor Nogueira (Universidade de Évora)

CRC2015

5ª feira, 19 de Novembro de 2015

Sessão 1

1 *André Souto*
Traffic analysis based on compression

7 *João Cunha, Ricardo Silva, João Marco Silva and Solange Rito Lima*
Classificação multinível de tráfego de rede com base em amostragem

13 *José Carlos Martins and Luis Rato*
QoS em servidores HTTP Apache

19 *Afonso Vilaça Bastos Silva, Mário Antunes and Diogo Gomes*
TVPulse: Improvements on detecting TV highlights in Social Networks using metadata and semantic similarity

Sessão 2

25 *António Daniel Lopes, Maria João Nicolau, António D. Costa, Joaquim Macedo and Alexandre Santos*
Encaminhamento de chamadas VoIP em redes Peer-to-peer

32 *Nelson Campos and José Legatheaux Martins*
Encaminhamento Míope

38 *Filipe Azevedo, Fernando Mira Da Silva and Luís Guerra E Silva*
A Scalable Architecture for OpenFlow SDN Controllers

44 *Flávio Meneses, Daniel Corujo, Carlos Guimarães and Rui Aguiar*
Experimental Evaluation of SDN-based End-to-End Mobility Management in Wireless Environments

6ª feira, 20 de Novembro de 2015

Sessão 3

46 *André Fernandes, António Pinto and Jorge Mamede*
Automatic and secure WLAN configuration by a captive portal

52 *Emanuel Lima, Óscar Gama and Paulo Carvalho*
Reconfiguração de Redes de Sensores Sem Fios: Estratégias e Análise Comparativa

57 *João Rodrigues, Jorge Mamede and Jaime Dias*
Serviço de E-mail Marítimo Utilizando Redes Tolerantes ao Atraso

Sessão 4

63 *Hugo Costa, Cristiano Cabrita, Jânio Monteiro and Jorge Semião*
Mecanismo de Deteção de Consumos Anómalos em Redes Energéticas Inteligentes

69 *José Venâncio, José Oliveira, Jorge Mamede and Rui Campos*
Localização de Precisão Usando Redes Sem Fios Enterradas

75 **Índice de Autores**

Traffic analysis based on compression

André Souto
SQIG - Instituto de Telecomunicações and Departamento de Matemática - IST - Universidade de Lisboa
Av. Rovisco Pais, 1049-001 Lisboa Portugal
Email: a.souto@math.ist.utl.pt

***Abstract*—Internet traffic analysis is nowadays a key ingredient to detect and possibly prevent malicious activity of the web that may cause severe constrains and/or security leaks.**

In this paper, using some properties that are easy to collect from the package head we study how one can use compression to efficiently cluster similarities of different types of data and infer their (ab)normality.

In particular, we envisage to answer if, using compression, it is possible to:

1) **Identify the protocols used in an internet connection;**
2) **Identify the services that the tcp internet protocol used;**
3) **Detect the type of traffic being transmitted via internet.**

I. Introduction

Since the early use of internet its use was perverted and instead of being use only to transfer information to facilitate service providing, it has also been used to transfer improper information and/or to deny the service provision.

Hence, Internet traffic analysis is of crucial importance to numerous network activities such as security, information assurance with special emphasis in the availability. Hence the community has been investing a lot of effort to detect attackers on networks and is trying to predict possible attack (through intrusion detection systems (IDS)) and cataloging them.

An intrusion detection and traffic analysis system usually works by detecting attacks to specific machines and by analyzing the pattern observed that such attacks produces in the traffic it exchanges and by creating an alert through signature matching of an attack occurring (i.e., once a known attack pattern has been identified, a signature for it is created and it can afterwords be used to give an alert). In the literature, one can find many examples of different approaches to traffic analysis. Some of those techniques include: machine learn [8], extract matching [9] and heuristics [4]. A similar approach to the one presented in this paper is presented in [13], where a measure exploiting the compression ratio to analyze and detect anomalies in traffic of several protocols is used. The idea was tantamount to collect several samples of the traffic to establish the pattern for each entity and set what should be considered to be "normal" for each particular user or network.

We use the normalized compression distance to perform the analyzes of the traffic and use the CompLearn algorithm to extend the previously mentioned work in order to catalogue, by clusters, the different protocols used, the type of services and the type of attack. This algorithm and the measure supporting it was presented in [2] and was based on a similar method to classify mitochondrial genome phylogeny [6]. It as been used to classify data such as languages tree, genomics, optical character recognition, literature [2], music [3], computer virus and internet traffic analysis [13] and medical data [10].

The underlying measure is grounded on Kolmogorov complexity [11], [5], [1]. Given a string x, the Kolmogorov complexity measures the amount of information required for a Turing machine to describe x by the length of the shortest program that, when run in that machine, outputs that string.

The CompLearn algorithm works in two steps:

1) It determines a universal similarity distance between each pair of objects, computed from the length of compressed data files.
2) A hierarchical clustering method is applied.

In the Conference on Knowledge Discovery and data mining every year there is a challenge proposed to the community related to the field. In 1999, the challenge was to construct a predictive model capable of distinguishing between legitimate and illegitimate connections in a computer network. The data made available by the organization are referring to a wide variety of traffic of different internet protocols, namely, TCP, UDP and ICMP and included the information of several internet services running over this protocols.

The data describe several attributes that have been hand-classified (based upon flow content) to one of a number of categorized or identified using the header of the package, such as type, service used and the type of connection. Hence we used the CompLearn algorithm to envisage the distance among the data to predict their similarities when those elements of the data were removed and hence identify them later by clustering.

II. The data used

The data base is used to test the hypothesis whether one can use CompLearn to infer characteristics of online traffic. The data base was provided by KDD – Conference on Knowledge Discovery and Data Mining 1999. The data are related with the network activity with attributes such as internet protocols used (tcp, udp and icmp), internet services running over this protocols (vmnet, smtp, ntpu, shell, kshell, aol, imap4, urhi, netbiosssn, tftpu, mtp, uucp, nnsp, echo, timi, ssh, isotsap, time, netbiosns, systat, hostnames, login, efs, supdup, http8001, courier, ctf, finger, nntp, ftpdata, redi, ldap, http, ftp, pmdump, exec, klogin), types of traffic (normal, ipsweep, back, teardrop, pod, potsweep, guess password, smurf).

The data include other type of information in a total of 43 possible attributes. We intended to explore the possibility of having CompLearn algorithm to classify, by clusters, the data

based on the other attributes other than the type of protocol, services used or the classification of the traffic. In order to validate these results, with real firewall data from Portuguese Navy School, we refer the reader to the recent master thesis [12].

As described in the Introduction, the data were prepared in several ways and analyzed accordingly. The pre-process of the data for each test was made by choosing from the original set some relevant entries for each test. For example, to distinguish between types of protocols, the entries were separated by "type of protocol" used, and samples of such entries were grouped into a file with the information provided by the firewall other than the type of protocol.

As mentioned above the data were split into parts in such a way that each file would contain information of an entry or a set of entries corresponding to same "internet protocols", "internet service" or "type of classification" with this information omitted. Then, we used the CompLearn algorithm to calculate the distance matrix between files. Such matrix is computed using the normalized compression distance between two files x and y, defined by:

$$NCD(x,y) = \frac{K(xy) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}},$$

where K is the Kolmogorov complexity. See Section III for details.

In order to analyze and interpret the results we clustered the information based on the matrix created in the previous step. The cluster embedded in CompLearn uses the quartet method. We refer the reader to the official website www.complearn.org and the papers[3], [2] for a fully detailed description of the algorithm.

III. Preliminaries

We now present basic definitions and results.We fix a binary alphabet $\Sigma = \{0,1\}$ and consider strings in Σ^*, usually denoted by x, y, z, etc. We use the notation $|x|$ for the length of strings, i.e., the number of bits occurring in x. In particular, the empty string ε as length 0. The $\log$ function is the shorthand for the logarithmic function of base 2. We suggest the reading of [7] for details on the topic of Kolmogorov complexity.

Definition 3.1 (Kolmogorov Complexity): Let U be a fixed prefix-free universal Turing machine. For any strings $x, y \in \Sigma^*$, the Kolmogorov complexity *of x given y is defined by* $K(x|y) = \min_p\{|p| : U(p,y) = x\}$.

The default values for y is the empty string and, typically, we drop this argument to avoid overloaded notation. The Kolmogorov complexity is machine independent, i.e., we can fix a universal Turing machine U whose program size is at most a constant additive term worse than in any other machine, and the running time is, at most, a logarithmic multiplicative factor slower than in any other machine.

Definition 3.2: The approach used is based on similarity distance, called *Normalized Compression Distance*, that for two strings x and y is defined by $NCD(x,y) = \frac{K(x \odot y) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}}$ where $\odot$ represents the concatenation of x followed by y and K is the Kolmogorov complexity, i.e., it is the length of the shortest program that given to a computer is able to print it.
Since Kolmogorov complexity is uncomputable the Normalized Compression Distance is also non-computable. It satisfies the symmetry of information property up to a small error term, i.e. it satisfies $K(x|y) \sim K(xy) - K(y)$ up to a logarithmic term depending only on x and y. Furthermore, it can be approximated by computable values. See [2] for a detailed discussed on the quality of the approximation for practical purposes by the size of compression of standard compression algorithms like gzip.

For understanding the resulting matrix, it becomes handy to hierarchically cluster the information of that matrix into groups. The clusters are computed using the *quartet method*. Consider every group of four elements and for each group $\{u,v,w,x\}$, construct a tree where each internal node has 3 neighbors. This implies that the tree consists of two subtrees of two leaves each. The possible topologies are (1) $uv|wx$, (2) $uw|vx$, and (3) $ux|vw$. The cost of each quarter is the sum of the distances between each pair of neighbors, that is, $C_{uv|wx} = d(u,v) + d(w,x)$. A larger tree is constructed by adding nodes to the underlying quartets already considered. Among all the possible trees, the best one is the one with the smallest cost (which is the sum of the cost of all quartets). For practical purposes of clustering, the CompLearn has inbuilt a method based on randomization and hill-climbing type of algorithms to find an approximation to the best tree. Notice that finding the best such tree is NP-hard and hence impracticable to find it in a reasonable time.

IV. Methodology and presentation of the results

We stress that the tests were performed taking into account the structure of each entry in the data. For each test, the selected entries were organized, filtered and then placed into files where relevant information such as "type of protocol", the "type of internet service" used and "classification of the kind of traffic" were omitted. The aim was to evaluate if the remaining fields of the entries would provide enough information to cluster similarly entries of that attribute nearby each other. It is worthwhile mentioned that, in case of positive results, this will allow, at a later stage of this research to automatically classify the information transmitted over the internet and to develop an automatic classifier based on compression for online traffic.

A. Detecting the type of protocol used

One of the attributes collected in the data is the "type of protocol" that each entry has. This first test was designed to check if the algorithm can differentiate all the protocols used. The set considered for this test consists of 4 files each of which with several entries of each type of protocols used. The matrix in Table I in page 5 shows the distances obtained with CompLearn and in Figure 1, shows the clusters constructed. Notice that the algorithm clustered all the files of different

protocols together, being the information of "tcp" more similar to "udp" than to "icmp".

B. Detecting the internet service of tcp entries

The most common entries in the KDD data base are related to the tcp protocol. Considering all the entries of the tcp protocol separated *per* files, in this test we envisage the possibility of cluster information per internet service. The results obtained are presented in Figure 2.

C. Detecting the internet service with normalized files

In the previous test, the number of entries for each internet service is different. In particular, there are files with just a few entries while, for example, http has thousands of registers. The idea of this test is to mitigate the possibility of biased results of the previous section due to file size. In order to eliminate the possibility of having the classification by the size of the files instead of its content in this test we normalized the size of the files. We restricted the test to the most abundant entries and considered services that have at least 9 entries. We created files, with the same number of entries, per type of service. The cluster can be found in Figure 3. Notice that it is similar to the results of the previous test when one considers only the significant services.

D. Detecting the internet service - single entry analysis

In order to understand deeper the structure of the entries of the tcp protocol and in particular the "services of the internet" used, in this test we considered single entries of the data base in each file in order to infer the robustness of the results presented above. We selected one entry at random from each type and applied CompLearn to obtain the cluster for comparison. The results are presented in Figure 4.

E. Detecting similar entries based on single entry analysis

The promising results obtained in the previous test lead to questioning whether the algorithm could group similar records selected from the database at random. In order to test it, we added more entries to the ones used in the previous test and run again the CompLearn. The results obtained are depicted in Figure VI.

F. Detecting types of attacks in the http service

The http service is the most common service used in the tcp protocol of the data used. Each entry of this service is classified as "normal", "ipsweep" and "back". In this test, we intend to check if the remaining information of the entries is enough to separate online traffic based on the classification given by the firewall into different clusters. The results appear in Figure 6.

G. Detecting abnormal traffic in the database

Finally, we consider all significant types of classification given by the firewall, independently of service used, and test if compression can still separate them. The idea is to see if the method used in the preceding test is robust to cluster the type of attacks occurring in the database. We filtered the entries by attack, and collect for each attack, the remaining information in a file and run the CompLearn. The results obtained are presented in Figure 7.

V. Analysis of the results obtained

In this section, we discuss the results aforementioned. We provide some insight and reasoning for the clusters presented. We divide this analysis in two parts. The former part aims the understanding the capability of CompLearn of detecting and clustering similar types of protocols and services. The latter one was designed to understand the capability of the algorithm to cluster classifications of the type of traffic.

A. Distinction between types of internet protocols used:

It is clear from the result that the data collected and their attributes are sufficiently different so that CompLearn is able to identify which kind of protocol was used. In particular, the algorithm clustered all the files of the same type in the same cluster. Moreover, clusters corresponding to udp protocol were divided in two contiguous clusters. This is explained by the individual analysis of the files, that are similar but in one of the attributes they are considerable difference (one is 32 and other 1048).

B. Distinction between services used in tcp

In the second test we used the most abundant data of tcp and, as in the first test, we checked if one could use the algorithm to split and distinguish the services that the entries were using. One factor that could, in principal, distort the results was the number of entries since the files considered were not all of same size, and for that reason the data produced might be a bit biased. In particular, the primary cluster grouped the files with services such as domain, gropher, imap4, link and rje, which correspond to the smaller files (i.e., the type of services with less entries).

C. Detecting the internet service of tcp with normalized data

In order to prevent misleading results, we normalized the data keeping only the services which would allow more than 3 registers. We fixed a certain number of registers (in this case 9) and choose the registers for each file of the services as random as possible. The results obtained are not significantly different from the previous ones with the difference that the cluster now obtained is missing the files with few registers. The remaining tree has a similar structure when the data were not normalized. The only difference is that authentication and finger are now separated by one node.

D. Distinguishing services of internet - single register analysis

As mentioned in one of the previous sections, we put the clusters to the test and see if similar services would be clustered nearly and the tree produced would keep the same shape as in the proceding test. Although not entirely equal the structure is quite similar. Some changes were observed concerning the position of the files such as authentication and ftpdata. So, the result suggests that some data (the ones placed nearly its correspondent in the previous tree), have enough regular information to identify the service by exploring that regularity. The others entries might have different or unusual properties that need a refinement to be able to have a proper classification.

E. Detecting similar registers based on single register analysis

In order to test the robustness of the previous result, we randomly had chosen another representative element of each service and perform similar analysis with this new entries and the the entries of the previous test. The results confirm that the structure of each entry is sufficiently robust to determine and to cluster the major part of the files corresponding to the same service altogether. We stress that these results need more tests, specially regarding ftpdata, authentication and http. In fact, more tests are needed since the first two services were clustered differently in the normalized and single tests. The latter one deserves more detailed analysis because looking into the information of the entries of http one can easily detect a wide variety of values for the attributes and, therefore, it would be very surprising that the algorithm would cluster all http entries nearly each other.

F. Detecting similarities/types of traffics

The last battery of tests is performed to analyze the classification presented in the data. The first intended to distinguish between the several http services and the latter one to distinguish them among all the possible protocols.

G. Distinction between http traffic

The test was prepared to analyze the most common internet service and identify the possibility of detecting differences in the several types of classifications of the original data: normal, ipsweep, portsweep and back. The clusters obtained suggests that the algorithm is able to group the information concerning the classification of "back" in a single group. The normal files were also grouped in two clusters. The "ipsweep" entries were placed in a single cluster and far from "back" entries and in the tip of the cluster. Regarding the "portsweep", it is clustered in an internal branch of the cluster of "normal" entries but all together. So, the results are encouraging to identify outliers from the data base! It is interesting to see that relatively to classification "back", the algorithm is able to group inside the same cluster very nearly all the entries considered with the "same flag" (other field attribute presented in each entry). The same happens with "normal" entries with flag "rej", which forms a cluster. On the other hand, normal registers with flags s_0, s_2 and s_f are placed together in the same cluster and flags with s_1 and s_3 are far apart.

H. Detecting abnormal traffic in the database

Finally we analyze the cluster obtained by considering entries of each protocol and for each type of attack. The results suggest that the types of attacks are separable in different clusters by the protocol used. In particular, almost all entries of the same protocol are grouped in the same cluster. The exception is pod that is tcp is placed in the same cluster as teardrop which used udp. Notice that ipsweep and portsweep, contrarily to the previous test, are very similar and are clustered together in the same group very closely. Notice also that in the previous tree there were only normal traffic between ipsweep and portsweep, suggesting that this type of files can be identified as similar.

VI. Conclusion

In this work, we have analyze data base of KDD relative to firewall traffic analysis and infer if some properties and attributes are able, by compression, to characterize the data as they were original characterized by the firewall. The data are composed of entries of internet packages transmitted via Internet and have several attributes, like protocol, service and classification of normal or abnormal traffic. We have used CompLearn algorithm to perform the tests designed to validate the first step of the hypothesis to design an automatic classifier of online traffic by simple algorithmic compression. In particular, we showed that the algorithm is able to distinguish, by placing the files in different clusters, different protocols (tcp, udp, icmp) and different services of tcp used. Also, in the tests we included the analysis for different threat classifications of the entries. In this case, the information was clustered in such a way that all file corresponding to "back" form a group, the "ipsweep" another and the "normal" ones were grouped into all 4 clusters with the particularity that are inner elements of the tree, close to the center.

As a future work it is important to study methods that allow to use the clusters obtained to classify new data and compare this results with other methods, namely classical techniques in data-mining. One possible approach is to use an algorithm that computes that, either by compiling the distances to representative elements or by using a new method to construct the tree of the distances between registers. It would be interesting to compare the efficiency and the results obtained with this method and the method presented in [13].

Acknowledgment

A very special thank is due to Sophie Laplante. I also thank Manuel Biscaia, Victor Lobo, Paulo Neves and Carlos Caleiro for helpful discussions. The author is partially funded by FCT grant SFRH/BDP/76231/2011 and funds from PEst-OE/EEI/LA0008/2013 and UID/EEA/50008/2013 granted to Instituto de Telecomunicaes.

References

[1] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of ACM*, 13(4):547–569, ACM Press, 1966.

[2] R. Cilibrasi and P. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

[3] R. Cilibrasi, P. Vitányi, and R. de Wolf. Algorithmic clustering of music based on string compression. *Computer Music J.*, 28(4):49–67, 2004.

[4] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport layer identification of p2p traffic. In *Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, pages 121–134, New York, NY, USA, 2004. ACM.

[5] A. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, Springer, 1965.

[6] M. Li, J. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny . *Bioinformatics*, 17(2):149–154, 2001.

[7] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.

[8] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. volume 3015 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 2004.

[9] A. Moore and K.Papagiannaki. Toward the accurate identification of network applications. In *In PAM*, pages 41–54, 2005.

[10] C. Santos, J. Bernardes, P. Vitanyi, and L. Antunes. Clustering fetal heart rate tracings by compression. In *19th IEEE CBMS 2006.* , pages 685–690, 2006.

[11] R. Solomonoff. A formal theory of inductive inference, Part I. *Information and Control*, 7(1):1–22, Academic Press Inc., 1964.

[12] T. Teles. Cibersegurança - Deteção de Outliers. Master thesis of Portuguese Navy School, 2015.

[13] S. Wehner. Analyzing worms and network traffic using compression. *J. Comput. Secur.*, 15(3):303–320, August 2007.

	icmp1	icmp2	icmp3	icmp4	tcp1	tcp2	tcp3...
icmp1	0.125984	0.496062	0.682926	0.590551	0.672043	0.638554	0.673684
icmp2	0.496062	0.116504	0.664634	0.514563	0.704301	0.668674	0.705263
icmp3	0.682926	0.664634	0.115853	0.548780	0.715053	0.626506	0.710526
icmp4	0.590551	0.514563	0.548780	0.070707	0.720430	0.668674	0.736842
tcp1	0.672043	0.704301	0.715053	0.720430	0.155913	0.602150	0.642105
tcp2	0.638554	0.668674	0.626506	0.668674	0.602150	0.126506	0.642105
tcp3	0.673684	0.705263	0.710526	0.736842	0.642105	0.642105	0.152631
tcp4	0.726851	0.740740	0.703703	0.768518	0.689814	0.689814	0.675925
udp1	0.591463	0.628048	0.609756	0.652439	0.645161	0.560240	0.673684
udp2	0.639534	0.680232	0.662790	0.680232	0.655913	0.598837	0.710526
udp3	0.600000	0.626666	0.609756	0.646666	0.645161	0.620481	0.673684
udp4	0.544871	0.596153	0.597560	0.621794	0.629032	0.512048	0.610526

TABLE I
DISTANCE MATRIX OBTAINED FROM COMPLEARN

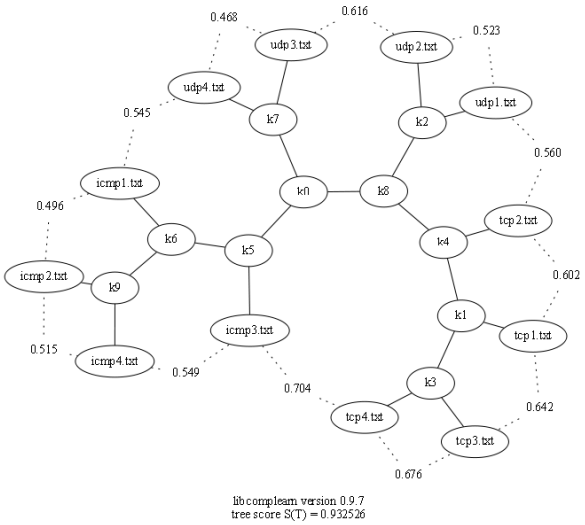


Fig. 1. Cluster of protocols.

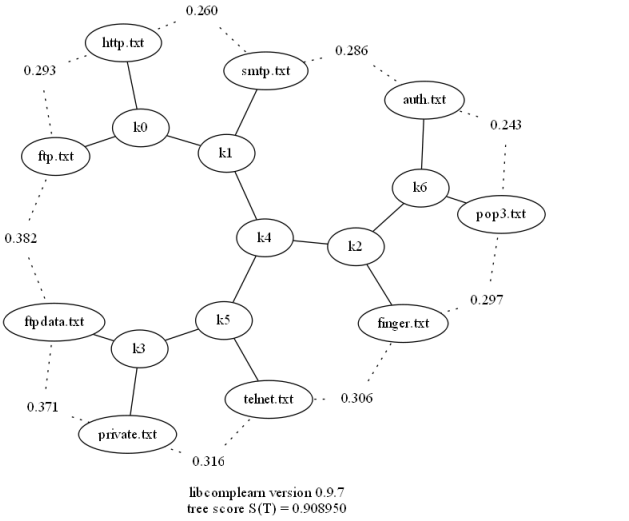


Fig. 4. Cluster of data normalized with a single register per file.

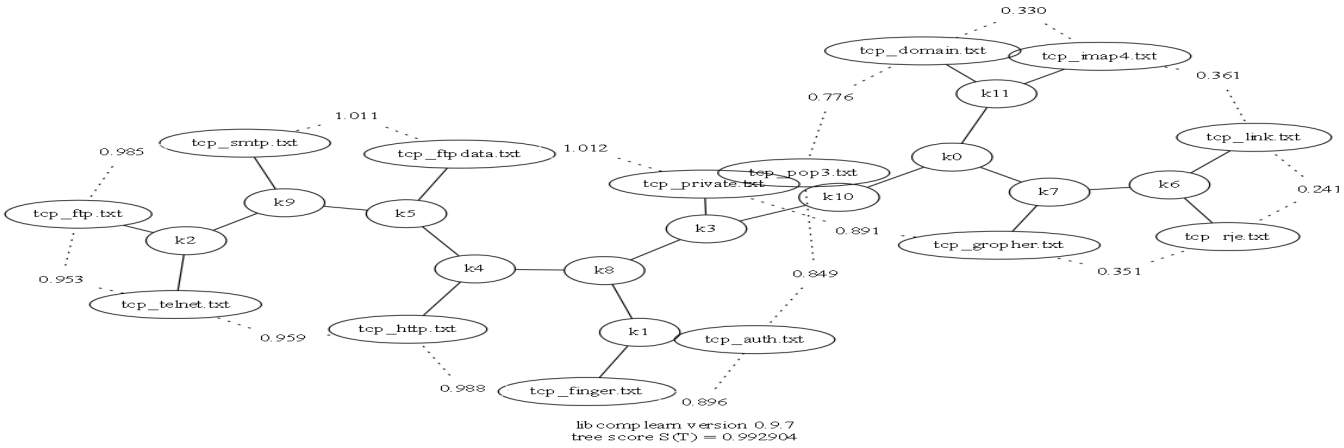


Fig. 2. Cluster of all data.

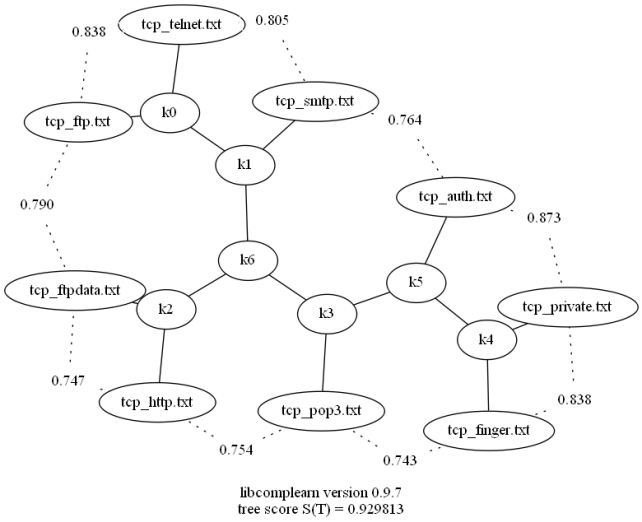


Fig. 3. Cluster of data normalized.

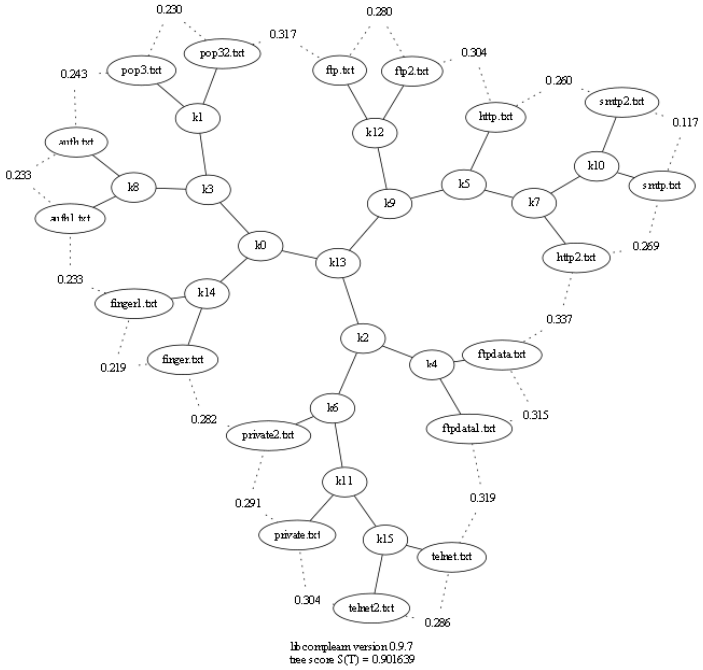


Fig. 5. Cluster of data with a single register per file and several of same kind.

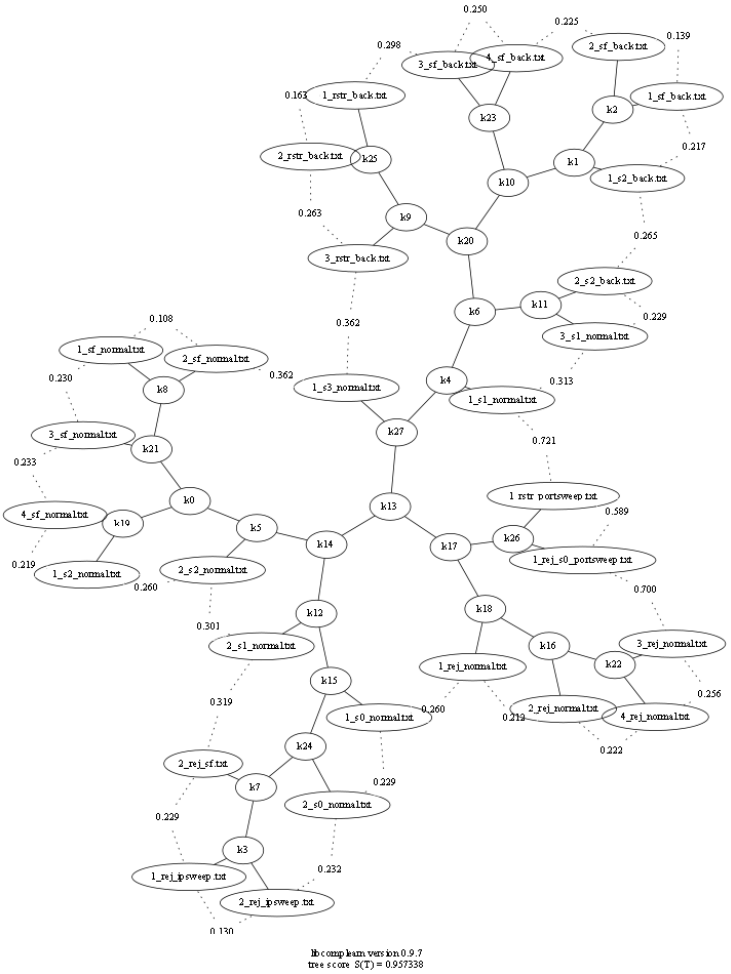


Fig. 6. Cluster of classification for http service.

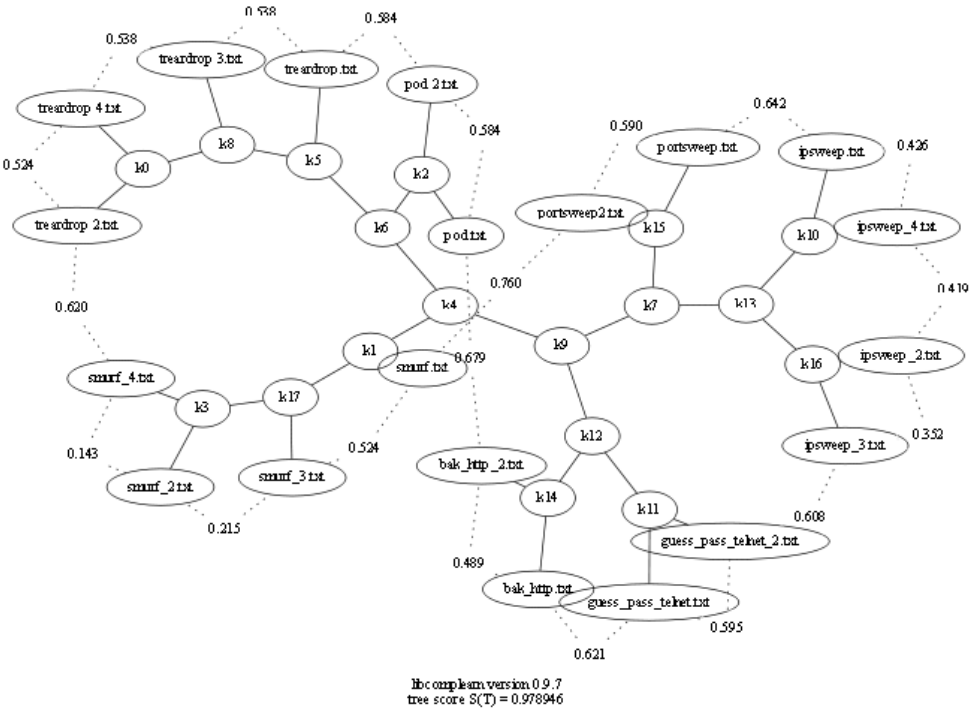


Fig. 7. Cluster of data of http with a single register per file and different classification type.

Classificação multinível de tráfego de rede com base em amostragem

João Cunha, Ricardo Silva, João M. C. Silva e Solange Rito Lima
Departamento de Informática, Universidade do Minho, Braga, Portugal
Email: pg28504@alunos.uminho.pt, pg28501@alunos.uminho.pt, joaomarco@di.uminho.pt, solange@di.uminho.pt

***Resumo*—A classificação e a caracterização do tráfego de rede são tarefas essenciais para o correto planeamento e gestão das actuais redes de comunicações. No entanto, face ao elevado volume de tráfego envolvido, essas tarefas podem beneficiar largamente do recurso a tráfego amostrado, desde que este permita obter uma visão da rede realista através de pequenas porções de tráfego.**

Neste contexto, este artigo tem como principal objetivo a exploração e comparação da acurácia de diferentes estratégias de classificação de tráfego de rede quando conjugadas com diferentes técnicas de amostragem. Recorrendo à ferramenta TIE - Traffic Identification Engine [1], que disponibiliza estratégias de classificação de tráfego operando em distintos níveis da pilha protocolar, e a uma framework de amostragem que implementa técnicas clássicas e emergentes de amostragem de tráfego [2], é analisado o impacto da estratégia de classificação e amostragem na correta identificação dos fluxos de rede. A base dos testes realizados recorre a tráfego de rede real coletado no Instituto Nacional de Estatística, posteriormente submetido a diferentes estratégias de classificação e amostragem. Desta forma, pretende-se contribuir com diretivas para a obtenção de uma classificação realista do tráfego de rede com o menor overhead em termos de tráfego coletado e analisado.

I. Introdução

A relevância e os desafios da classificação de tráfego tem crescido na última década na comunidade das redes e telecomunicações, tornado-se numa tarefa crítica para apoio a aspetos que vão desde o planeamento de redes, à auditoria de contratos de serviços e à deteção de ataques (DoS Attacks). No entanto, aspetos como o aumento da diversidade e volume de aplicações que circulam na rede, o encapsulamento de aplicações sobre distintos protocolos aplicacionais, a utilização de protocolos de segurança (IPSec, TTLS, HTTPS, etc.) na cifra dos dados, etc. colocam desafios acrescidos a uma correta classificação do tráfego. Ter a noção de que tipo de aplicações são mais utilizadas, quem as utiliza e com que fins, são razões que levam a que hoje em dia haja um maior controlo sobre o tráfego na Internet e a classificação desse mesmo tráfego de uma forma prática e ágil é necessária para obter esse controlo.

Para fazer face a estes problemas, foi criada uma ferramenta de software aberta à comunidade que permite fazer classificação de tráfego, denominada Traffic Identification Engine (TIE) [1] [3]. O TIE foi desenvolvido por uma equipa de investigadores da Universidade de Nápoles [4], que se basearam em observações dos problemas anteriormente referidos para fornecer à comunidade uma plataforma de fácil desenvolvimento e integração de técnicas de classificação de tráfego. Esta ferramenta é flexível e facilmente conciliável com outras aplicações, logo extensível a outros ramos como, por exemplo, a amostragem de tráfego. A amostragem surge, neste contexto, como a estratégia utilizada para tornar tratável o volume de dados a coletar, analisar e armazenar, já que apenas um subconjunto de tráfego da rede será usado na classificação. No entanto, idealmente, a análise desse subconjunto deverá ser capaz de permitir inferir com realismo o que se passa na rede em termos de fluxos de tráfego.

Assim, o principal objetivo deste artigo é estudar o impacto de diferentes técnicas de classificação na correta identificação dos fluxos de tráfego quando apenas parte do tráfego da rede é utilizado na análise. O subconjunto de tráfego analisado é obtido usando técnicas clássicas e recentes de amostragem de tráfego, tais como técnicas sistemáticas, aleatórias e adaptativas. A prova de conceito recorre ao auxílio da ferramenta TIE, de uma framework de amostragem desenvolvida na Universidade do Minho [2] que implementa as técnicas de amostragem referidas, e a tráfego real obtido na rede do Instituto Nacional de Estatística.

Este artigo encontra-se organizado da seguinte forma: na secção II é abordado o trabalho relacionado com o tema em estudo, nomeadamente em classificação e amostragem; na Secção III serão apresentados alguns conceitos fundamentais sobre as ferramentas e técnicas de amostragem utilizadas; na Secção IV é apresentada a metodologia de testes e algumas ferramentas desenvolvidas no seu âmbito por forma a agilizar a interação com a ferramenta TIE e auxiliar na interpretação dos resultados obtidos; na Secção V são discutidos os resultados obtidos; por fim, na Secção VI, são apresentadas as considerações finais e possíveis ideias para trabalho futuro.

II. Trabalho Relacionado

O encapsulamento de aplicações em protocolos conotados com serviços diferentes, o aumento da utilização de protocolos de segurança, o aumento de aplicações que alocam dinamicamente portas de comunicação, tornam as técnicas de classificação imprecisas quando são baseadas unicamente em protocolos de transporte e nas portas de origem/destino. Por esse motivo, existem vários métodos alternativos para a classificação de tráfego, nomeadamente: (i) análise do *payload*, e.g., por *pattern matching* [5] e análise numérica [6], que são técnicas bastante efetivas na taxa de acerto, mas pesadas e não aconselháveis para classificar tráfego encriptado; (ii) análise

do comportamento dos sistemas ou terminais (*host-behavior*) [6], em que a classificação é feita através da análise do comportamento dos sistemas terminais em vez dos fluxos de tráfego, evitando o processamento do *payload*; e (iii) análise de fluxos (*flow-behavior*), em que se assume que cada aplicação tem as suas próprias propriedades estatísticas.

Para garantir a qualidade e o bom funcionamento dos serviços de rede, é necessário que as ferramentas de análise e classificação de tráfego implementem de forma escalável e fiável as diferentes técnicas de classificação de tráfego. Porém, devido ao crescimento do volume de tráfego é cada vez mais difícil fazer uma análise ao tráfego total, por isso, a adoção de mecanismos de classificação de tráfego conjugados com mecanismos de amostragem de tráfego constituem um cenário cada vez mais útil e necessário. De facto, a utilidade da amostragem de tráfego tem sido explorada em diferentes áreas das redes de telecomunicações, nomeadamente: segurança de redes - deteção de anomalias e intrusões, botnet e identificação de DDoS [7]; determinação da conformidade dos SLA's e controlo de QoS - para estimação dos parâmetros, tais como atraso de pacotes, perda e jitter [8]; engenharia de tráfego - auxiliar a classificação e caracterização de tráfego [9].

No contexto deste trabalho, diferentes técnicas de amostragem vão ser conjugadas com distintas estratégias de classificação no sentido de avaliar o seu desempenho conjugado na correta deteção e classificação de fluxos. Desta forma, pretende-se também inferir de que forma a análise do tráfego de rede pode ser efectuada de forma mais eficiente.

III. Classificação e amostragem de tráfego

A. Ferramenta TIE

O TIE é uma ferramenta escrita em linguagem C e pensada para ambientes Unix. Pode ser executado em modo offline ou modo online. O modo offline consiste na leitura de fluxos de tráfego previamente coletados, enquanto que o modo online lê em tempo real os fluxos de tráfego que são capturados da rede à qual o dispositivo está ligado. A aplicação consegue correr dinamicamente vários plugins, que representam técnicas de classificação de tráfego implementadas em software (pode correr um ou mais plugins ao mesmo tempo). Na figura 1 está representada a arquitetura da ferramenta [3].

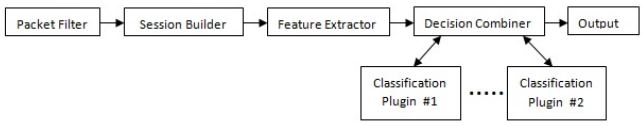


Figura 1. Componentes do TIE

O *Packet Filter* captura tráfego em tempo real ou lê *traces* que foram capturados previamente. Após o input na aplicação, os pacotes são agregados em sessões separadas do tipo flow, biflow, etc. pelo *Session Builder*. O *Feature Extractor* realiza a extração dos campos dos pacotes que são necessários para a classificação. O tipo de classificação é feita pelo *Decision Combiner* que gere a execução dos diferentes plugins ativos. O *Output* gera o output final num formato em que é possível verificar a classificação do tráfego de forma estatística, que permite a comparação de múltiplas abordagens.

B. Técnicas de amostragem

A amostragem de tráfego consiste na coleta de subconjuntos do tráfego de rede, obtidos aplicando ao tráfego total técnicas de amostragem. As técnicas de amostragem avaliadas neste trabalho correspondem às principais técnicas atualmente utilizadas em ferramentas de medição de rede, i.e., *systematic count-based*, *systematic time-based* e *random count-based* [10]. Além disso, duas técnicas adaptativas são avaliadas, i.e., *predição linear adaptativa* [11] e *amostragem multi-adaptativa*[12]. Essas técnicas são a seguir sumariamente descritas: 1) Systematic count-based (SystC): a seleção de pacotes é efetuada através de uma função deterministica e invariável baseada na posição de pacotes, usando contadores [10]. Como exemplificado na Figura 2 (a), cada quinto pacote é selecionado e capturado pelo processo de amostragem. 2) Systematic time-based (SystT): o processo de selecção de pacotes segue uma função determinista baseado no tempo de chegada ao ponto de medição [10]. Nesta técnica, o tamanho da amostra e o tempo entre amostras são definidas no início e permanecem inalteradas ao longo do processo de amostragem, como apresentado na Figura 2 (b). Como ilustrado, todos os pacotes que chegam ao ponto de medição ao longo de um período de 100 ms são selecionados para a amostra, considerando que todos os pacotes seguintes ao longo de 200ms são ignorados para fins de medição. 3) Random count-based (RandC): seleciona os pontos iniciais dos intervalos de amostragem de acordo com um processo aleatório. Como apresentado na Figura 2 (c), na aproximação aleatória n-out-of-N , n pacotes são selecionados aleatoriamente da população que consiste em N pacotes [10].

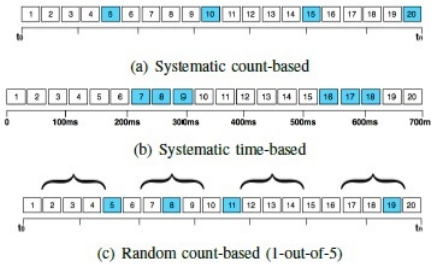


Figura 2. Exemplos de técnicas de amostragem

4) Adaptive linear prediction (LP): esta técnica baseada no tempo, usa predição linear para identificar alterações na actividade da rede, alterando a frequência de amostragem de acordo com o observado (i.e. reduzindo o intervalo entre amostras quando se observa mais tráfego que o previsto, para melhor deteção do novo padrão de tráfego) [11].

5) Multiadaptive sampling (MuST): esta técnica baseada no tempo usa mecanismos similares à anterior para identificar o nível de atividade da rede, no entanto, quer o intervalo entre

amostras quer o tamanho de cada amostra são parâmetros ajustáveis para melhorar o desempenho da amostragem [12].

Neste trabalho, as amostras obtidas usando as diversas técnicas descritas são processados pela ferramenta TIE para posterior classificação do tráfego. Comparando os resultados com os obtidos usando o tráfego global é possível avaliar *overhead* e acurácia, e selecionar qual a melhor técnica de amostragem para inferir corretamente sobre a classificação do tráfego total. A secção seguinte aborda em mais detalhe a metodologia seguida para a concretização deste objetivo.

IV. Metodologia de Testes

A Figura 3 ilustra os aspectos considerados nos testes realizados, tendo como base *traces* de tráfego real capturados na rede do Instituto Nacional de Estatística (INE).

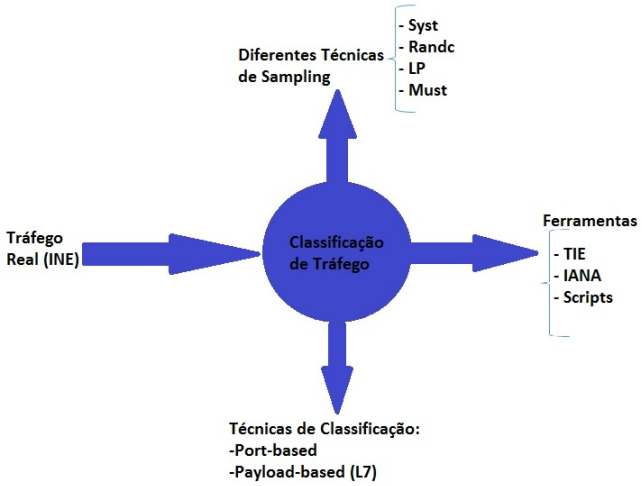


Figura 3. Arquitetura de testes

A. Classificação e amostragem

Na classificação de tráfego foi utilizada a ferramenta TIE e os seus plugins que implementam diferentes técnicas de classificação, nomeadamente port-based e *payload*-based, permitindo também a sua conjugação. Posteriormente, foi implementada uma ferramenta para interpretação dos resultados do output do TIE e, por consequência da análise do volume de tráfego não classificado, foi adicionada à ferramenta de interpretação dados fornecidos por parte da entidade reguladora da atribuição dos números de portas e endereços IP à escala mundial IANA - Internet Assigned Numbers Authority[1].

Face à necessidade de comparar o impacto da amostragem na classificação do tráfego total, inicialmente, o *trace* de tráfego do INE é injetado numa framework de amostragem cujo output são os vários *traces* com o tráfego amostrado através de cada uma das técnica de amostragem mencionadas. Neste ponto, foi desenvolvida uma script na linguagem de programação C para agilizar a interação entre os *traces* amostrados e o TIE para a classificação de tráfego, obtendo-se os resultados gerados pelo TIE para todos os *traces* de tráfego amostrado que serão, por fim, interpretados com o auxílio da ferramenta de interpretação de resultados desenvolvida. Esta sequência de passos é ilustrada na Figura 4.

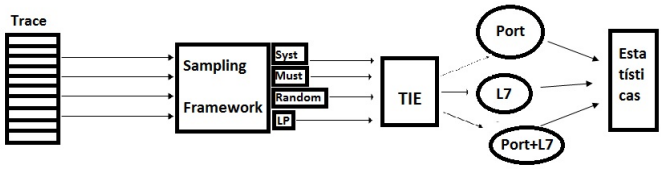


Figura 4. Esquema dos testes realizados

B. Interpretação de resultados

A ferramenta de interpretação de resultados foi desenvolvida na linguagem JAVA e tem como objetivo apresentar os resultados da classificação do tráfego numa forma mais intuitiva. Numa primeira instância, a ferramenta lê o output gerado pelo TIE e combina esse output com a lista de aplicações reconhecida pelo TIE. Dado que a taxa de tráfego desconhecido se mantinha elevada, foi criada uma segunda versão da ferramenta que, em vez de consultar a lista de aplicações do TIE, consulta uma lista fornecida pelo IANA permitindo uma classificação mais atualizada em termos de aplicações. Ambas as versões da ferramenta geram o output em forma de gráficos.

V. Discussão dos Resultados

Os testes realizados, numa primeira fase, focam a classificação multinível utilizando tráfego total e, posteriormente, o impacto das diferentes técnicas de amostragem. Este último aspecto é analisado sobre diversas perspectivas: (i) a capacidade de identificação de fluxos existentes e Heavy Hitters (i.e., os fluxos mais significativos em termos de volume de dados); (ii) a acurácia na identificação dos protocolos da camada de transporte e aplicação. São ainda apresentados resultados das técnicas de amostragem com diferentes frequências de amostragem, para avaliar o compromisso entre *overhead* e acurácia na classificação. Os parâmetros estatísticos comparativos utilizados incluem: número de fluxos processados e identificados, taxa de identificação de aplicações, grau de confiança da classificação do tráfego, erro médio absoluto (MAE) e erro quadrático (MSE).

A. Análise de tráfego Total

Nos vários resultados apresentados a seguir foi utilizado o *trace* do INE correspondente ao tráfego total[2]. Na Figura 5 apresentam-se os dados referentes à classificação do tráfego pelo plugin *port* e na Figura 6 os dados referentes à classificação pelo plugin *L7*. As diferenças da classificação feita pelos plugins *port* e *L7* em termos globais e de grau de confiança não são muito significativas, notando-se maiores diferenças em termos das aplicações identificadas de baixo volume (representadas na figura por Others). Relativamente

[1]IANA - http://www.iana.org/

[2]Este *trace* corresponde a coletas de 20 min em diferentes periodos de atividade da rede. No global, o *trace* possui 252.087 fluxos individuais unidireccionais em cerca de 3 milhões de pacotes.

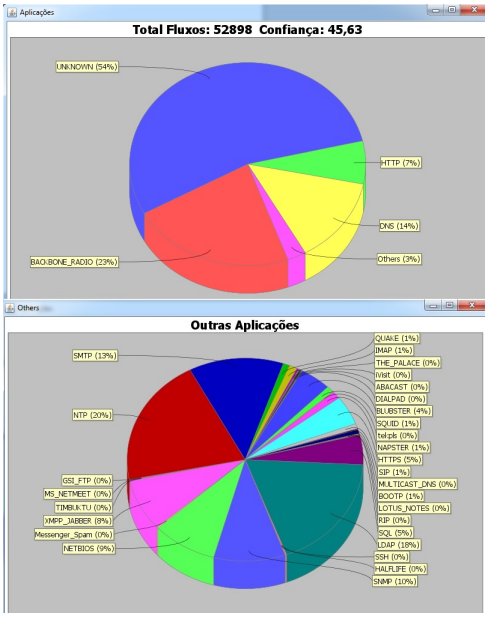


Figura 5. Classificação do tráfego *port-based*

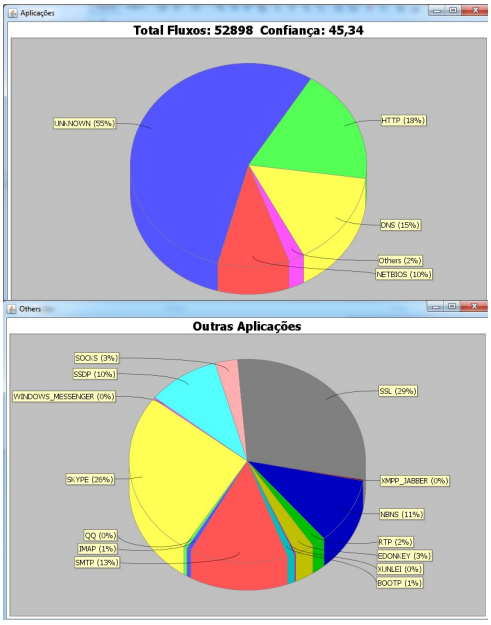


Figura 6. Classificação do tráfego *payload-based*

à aplicação conjunta dos dois plugins, a taxa de tráfego desconhecido diminuiu ligeiramente, salientando-se que as classificações *port+L7* e *L7+port* apresentam diferenças (na documentação da ferramenta TIE[1], encontra-se indicado que a ordem pela qual os plugins estão ativos tem impacto na classificação final).

1) Análise com IANA: Como referido anteriormente, no sentido de aumentar o volume de aplicações identificadas por porta, a base de dados de aplicações fornecida pelo TIE foi complementada com a lista de aplicações fornecida pela IANA. Pela análise da Figura 7, verifica-se que na classificação

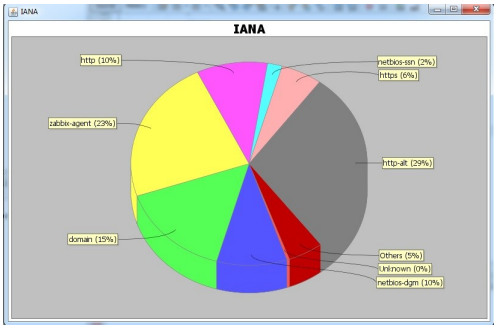


Figura 7. Classificação do tráfego com informação da IANA

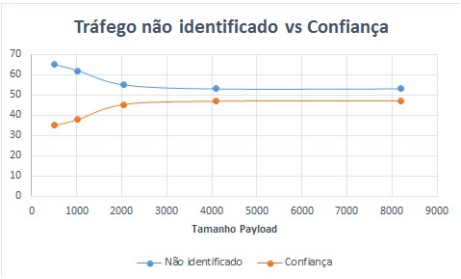


Figura 8. Impacto do tamanho do *payload*

por portas da IANA se obtém 0% de tráfego não identificado, sendo reconhecidas um muito maior número de aplicações.

2) Impacto do tamanho do payload na classificação: Um dos parâmetros exigidos na execução do plugin L7 é o tamanho (em bytes) do *payload* que irá ser analisado para a classificação de tráfego, sendo o tamanho padrão 2048 bytes. No sentido de avaliar o impacto desse valor no volume de tráfego identificado, realizaram-se experiências com 5 tamanhos alternativos de *payload*, mais concretamente 512, 1024, 2048, 4096, 8192 bytes. A Figura 8 compara o tamanho do *payload* com a percentagem de aplicações identificadas no tráfego e o grau de confiança obtido. Como ilustrado, o aumento do tamanho do *payload* analisado reduz o número de fluxos de tráfego com aplicações desconhecidas e aumenta o grau de confiança dos resultados obtidos. Estas variáveis tendem a estabilizar após um *payload* de cerca de 3000 bytes, em que o aumento de dados analisados se torna infrutífero.

B. Análise de tráfego Amostrado

A amostragem de tráfego tem como principal objetivo reduzir o impacto da monitorização no funcionamento da rede, mantendo a acurácia na estimação de parâmetros referentes ao seu comportamento estatístico. Nas secções seguintes são apresentados os resultados alcançados através da comparação dessas técnicas no contexto da classificação, nomeadamente na análise aos fluxos identificados e *Heavy Hitters*, na identificação de protocolos de transporte e aplicações. Numa primeira fase é utilizada a técnica port-based e, posteriormente, é analisado o impacto da alteração da técnica de classificação, considerando-se classificação port-based e *payload*-based.

1) Identificação de fluxos existentes e Heavy Hitters: Previsivelmente, a frequência de amostragem é diretamente proporcional à acurácia das estimativas, dado que o aumento do tráfego coletado resulta num maior conjunto de dados para análise estatística. Enquanto isto pode ser verdade na análise de uma técnica de amostragem, quando se altera o método de amostragem, o compromisso entre o aumento do tráfego amostrado e a acurácia das estimativas pode ter diferenças significativas. Esta secção foca a análise efectuada com a técnica SystC com várias frequências de amostragem e, posteriormente, compara as diferentes técnicas entre si. São usados dois parâmetros comparativos, nomeadamente: (i) a quantidade de fluxos identificados; e (ii) a percentagem de fluxos *heavy hitters* (HH) identificados, em que a noção de *heavy hitter* se refere aos fluxos mais significativos em termos de volume de dados (neste artigo considerados como os fluxos maiores que representam 20% do volume de tráfego).

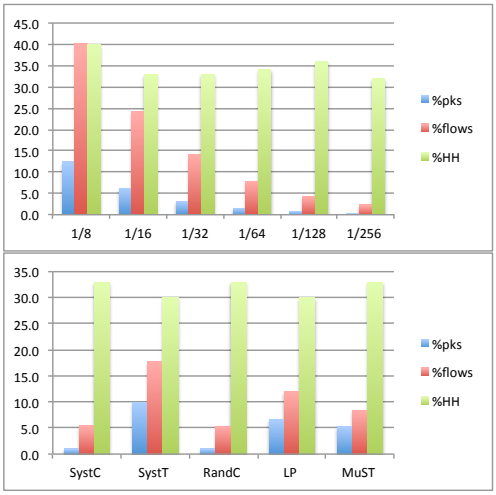
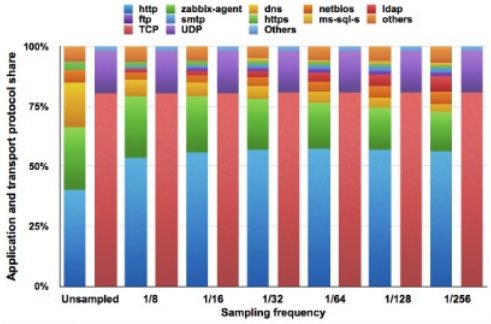


Figura 9. Identificação de fluxos - análise comparativa

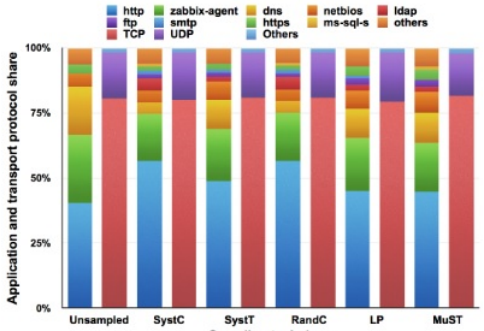
Analisando a Figura 9 verifica-se que com a técnica de amostragem SystC diminuindo o número de pacotes coletados, diminui também o número de fluxos identificados. Como exemplo, a frequência SystC 1/8 representa que é selecionado 1 pacote em cada 8. Numa análise mais pormenorizada do parâmetro dos fluxos *heavy hitters* conclui-se que a redução da frequência de amostragem não implica uma diferença significativa na identificação destes fluxos.

Relativamente às diferentes técnicas de amostragem, denota-se que um maior número de pacotes implica um maior número de fluxos identificados. No entanto, as técnicas *count-based* são mais eficientes, pois para o mesmo número de pacotes amostrados a percentagem dos fluxos identificados é significativamente superior. Isto acontece devido às políticas de seleção de pacotes distintos em uso, ou seja, o processo de seleção de pacotes baseado nas técnicas *count-based* aumenta a probabilidade de captura de fluxos distintos, ao contrário das técnicas *time-based* em que os pacotes são selecionados sequencialmente.



	1/8	1/16	1/32	1/64	1/28	1/56
MAE	0.0099	0.0464	0.1738	0.1337	0.1232	0.3045
MSE	0.0017	0.0026	0.0383	0.0262	0.0189	0.1383

Figura 10. Análise na camada de transporte e aplicação - SystC



	SystC	SystT	RandC	LP	MuST
MAE	0.2464	0.0999	0.0462	0.7521	0.7634
MSE	0.0817	0.0112	0.0330	0.7992	0.7671

Figura 11. Análise na camada de transporte e aplicação - Comparação entre técnicas

2) Identificação de protocolos de transporte e aplicação: Considerando uma análise ao nível da camada de transporte, a redução do número de pacotes amostrados promovido pela diminuição da taxa de frequência de amostragem da técnica SystC, não afeta a acurácia da classificação de tráfego, como apresentado na Figura 10 e confirmado pelos baixos erro médio absoluto (MAE) e erro quadrático (MSE) em todas as frequências. Porém, considerando a camada de aplicação, a distribuição das aplicações é bastante afetada, resultando numa elevada quantidade de tráfego *HTTP* identificado.

Embora a alteração da técnica de amostragem não tenha impacto na acurácia na classificação da camada de transporte (ver Figura 11), em termos da camada aplicacional isso já não se verifica. Como apresentado também na Figura 11, as técnicas baseadas em tempo conseguem ter um desempenho mais realista na distribuição da taxa de utilização da camada aplicacional, em que a técnica *MuST* obtém os melhores resultados. Globalmente, os resultados evidenciam que pequenas frações do tráfego de rede permitem obter uma visão bastante útil e realista dos fluxos de rede na pilha protocolar.

3) Impacto da técnica de classificação: De seguida, encontram-se os resultados da classificação do tráfego com as técnicas de amostragem acima referidas e as diferentes

Técnicas de Amostragem	Numero de Fluxos	Unknown (%)	Grau de Confiança (%)
Systc1-8	26382	84	15,64
Systc1-16	16767	87	13,21
Systc1-32	10207	88	11,66
Systc1-64	5871	89	11,12
Systc1-128	3252	89	10,58
Systc1-256	1768	89	11,09
Systc1-512	960	88	12,29
Systc1-1000	487	91	9,45

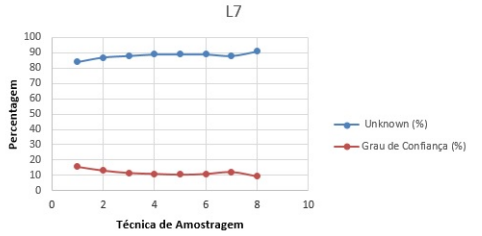


Figura 12. Amostragem sistemática com classificação *payload-based*

Técnicas de Amostragem	Numero de Fluxos	Unknown (%)	Grau de Confiança (%)
Systc1-100	4032	89	10,99
Systt	9784	68	31,67
Randc	3975	89	11,37
LP	6918	67	32,58
MUST	4088	65	34,89

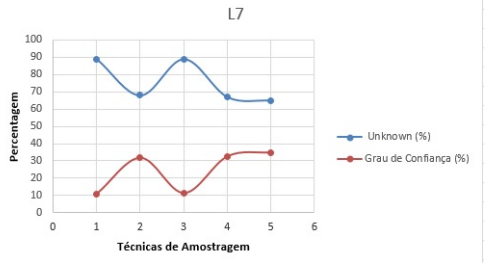


Figura 13. Técnicas de amostragem com classificação *payload-based*

estratégias de classificação port-based e L7. A técnica SystC é aplicada para as frequências *Systc1-8, Systc1-16, Systc1-32, Systc1-64, Systc1-128, Systc1-256, Systc1-512 e Systc1-1000.* num dos *traces* mais significativos do INE.

Os resultados obtidos com a classificação port-based de *payload*-based foram bastante similares. Pela análise da Figura 12 concluí-se que qualquer uma das técnicas de classificação de tráfego tem uma elevada taxa de tráfego desconhecido, o que já acontecia considerando o tráfego total. Ao considerar um estudo mais minucioso dos gráficos e das curvas de tráfego desconhecido e grau de confiança observa-se que estas duas métricas estatísticas são inversamente proporcionais.

Quando comparando as várias técnicas de amostragem (ver Figura 13), mais uma vez, é notório que as curvas dos parâmetros estatísticos são inversamente proporcionais. Numa análise mais pormenorizada denota-se que a técnica de amostragem com melhores resultados é a amostragem multi-adaptativa (*MUST*) pois obtém sempre taxas de tráfego desconhecido inferiores e graus de confiança maiores que as outras técnicas. Também se salienta que a classificação de tráfego *port+L7* obtém os melhores resultados, independentemente da técnica de amostragem utilizada. A combinação da técnica de amostragem (*MUST*) com a técnica de classificação de tráfego *port+L7* tem os melhores resultados, obtendo um grau de aplicações identificadas superior a 50%.

VI. Conclusões

Neste artigo foi apresentada uma análise comparativa do impacto da utilização de diferentes técnicas de classificação e amostragem na classificação de tráfego de rede. Nesse contexto, avaliou-se a aplicabilidade real das atuais e emergentes técnicas de amostragem para a identificação e classificação de fluxos, tendo em conta que este é um importante passo para tornar essas tarefas mais tratáveis evitando estimativas menos realistas da utilização da rede. Neste aspeto, as técnicas *count-based* revelaram ser mais eficientes uma vez que permitem identificar mais fluxos, com a mesma quantidade de tráfego amostrado. Adicionalmente, é notório que uma maior frequência de amostragem não conduz necessariamente a uma maior acurácia nas estimativas e, dependendo dos objetivos, pode ser vantajoso o recurso a uma amostragem de baixa frequência ou uma técnica de amostragem específica quando se pretende melhorar o compromisso entre o excesso de dados a processar e armazenar, e a acurácia dos resultados quando se estuda o comportamento dos fluxos de rede.

AGRADECIMENTOS - Este trabalho é suportado pela FCT - Fundação para a Ciência e Tecnologia no âmbito do projeto UID/CEC/00319/2013.

Referências

[1] W. de Donato, A. Pescapé, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *Network, IEEE*, vol. 28, no. 2, pp. 56–64, 2014.

[2] J. M. C. Silva, P. Carvalho, and S. R. Lima, "A modular sampling framework for flexible traffic analysis," in *SoftCOM 2015 - 23rd International Conference on Software, Telecommunications and Computer Networks*, 2015.

[3] A. Dainotti, W. De Donato, and A. Pescapé, "Tie: A community-oriented traffic classification platform," in *Traffic Monitoring and Analysis*. Springer, 2009, pp. 64–74.

[4] A. Dainotti, W. De Donato, A. Pescapé, A. Botta, G. Aceto, and G. Ventre, "Tie - traffic identification engine," http://tie.comics.unina.it/doku.php?id=topmenu:home.

[5] R. Ramaswamy, L. Kencl, and G. Iannaccone, "Approximate fingerprinting to accelerate pattern matching," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, pp. 301–306.

[6] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[7] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *Network, IEEE*, vol. 23, no. 1, pp. 6 –12, 2009.

[8] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On passive one-way loss measurements using sampled flow statistics," in *INFOCOM 2009, IEEE*, april 2009, pp. 2946 –2950.

[9] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapé, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, pp. 451–476, 2012.

[10] T. Zseby, M. Molina, and N. Duffield, "Sampling and Filtering Techniques for IP Packet Selection," RFC 5475, Internet Engineering Task Force, Mar. 2009. [Online]. Available: http://datatracker.ietf.org/doc/rfc5475/

[11] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive sampling for network management," *Journal of Network and Systems Management*, vol. 9, no. 4, pp. 409–434, 2001.

[12] J. M. C. Silva, P. Carvalho, and S. R. Lima, "A multiadaptive sampling technique for cost-effective network measurements," *Computer Networks*, vol. 57, no. 17, pp. 3357–3369, 2013.

QoS em servidores HTTP Apache

José Carlos Martins, Luis Miguel Rato

***Resumo*—Os serviços baseados na Internet têm registado um crescimento contínuo e acelerado nos últimos anos, dependendo o seu sucesso, em larga medida, da qualidade de serviço (QoS) prestada. A sociedade moderna tornou-se fortemente dependente da Internet e dos vários serviços nela disponibilizados.**

Neste artigo é apresentado um sistema original de controlo em cadeia fechada de QoS, aplicado ao servidor HTTP Apache. O objetivo principal do sistema é implementar um mecanismo capaz de realizar a diferenciação de serviço (*DiffServ*) entre duas classes de sítios: *Premium* e *Outros*, através da introdução de um atraso variável de modo a regular a carga do servidor para cada uma das classes.

***Palavras chave*—Servidor HTTP, Apache, Teoria de controlo, QoS, Erro nulo, Rejeição de perturbações, Seguimento de referência.**

I. Introdução

COM o advento da "computação em núvem", os serviços baseados na Internet desempenham um papel cada vez mais importante. O seu sucesso depende, em larga medida, da qualidade de serviço (QoS) prestada[1], o que enfatiza a necessidade de monitorizar a QoS de modo a impor um "Acordo de Nível de Serviço"(ANS)[2] e de implementar mecanismos capazes de realizar a diferenciação de serviço (*DiffServ*).

O aumento do uso de servidores HTTP representa um constante desafio ao seu desenho e desenvolvimento, no sentido de lhes permitir melhorar o desempenho, a disponibilidade e a escalabilidade. Embora na maior parte dos casos não seja técnica e economicamente possível alocar recursos dimensionados para os picos de utilização é, ainda assim, possível fornecer uma melhor QoS a sítios privilegiados (*Premium*), mesmo quando os servidores estão sobrecarregados com pedidos[3].

O recurso a sistemas de QoS é, cada vez mais, uma exigência/tendência no desenho e desenvolvimento de servidores HTTP.

A teoria de controlo é uma ferramenta poderosa no desenho de sistemas de QoS, permitindo alocar de forma dinâmica diferentes recursos a diferentes classes de pedidos baseados em amostras de *feedback*. Pese embora a teoria de controlo seja utilizada em várias áreas, no caso concreto de sistemas computacionais, o objetivo principal é torná-los mais robustos e estáveis. As áreas mais populares de pesquisa têm sido as redes informáticas, sistemas operativos, servidores HTTP, sistemas de gestão de bases de dados (SGBD), multimédia e gestão de energia[4].

Administradores de sistemas experientes sabem que deixar a gestão de sistemas dinâmicos a cargo de operadores não é aceitável, já que as alterações ocorrem tão rapidamente que não é humanamente possível responder em tempo útil[4].

Nas últimas duas décadas foram realizados vários estudos sobre QoS em serviços baseados na Internet, alguns dos quais com especial ênfase na diferenciação de serviço [4], [5], [6], [7], [8]. Linhas mais recentes de investigação abordam os controladores *fuzzy*, os quais são controladores dinâmicos, não-lineares[9], [10], [11].

Universidade de Évora

Universidade de Évora, Escola de Ciências e Tecnologia/ LISP (Laboratório de Informática, Sistemas e Paralelismo)

II. Equipamentos e Metodologia

A. Equipamentos

No sentido de implementar um sistema de QoS aplicado ao servidor HTTP Apache, baseado em teoria de controlo, é implementada uma infraestrutura informática dedicada. Esta infraestrutura é constituída por vários componentes, designadamente: um servidor HTTP Apache modificado (de agora em diante referido como servidor Apache), dois clientes HTTP de agora em diante referidos como clientes), uma *Gateway* (onde reside todo o mecanismo de QoS, baseado em teoria de controlo: controlador proporcional-integral-derivativo (PID), transdutor, *etc.*), dois switch e respectiva cablagem.

No Quadro I apresentam-se as principais características dos clientes, da *Gateway* e do servidor Apache. Os dois *switch* utilizados são do padrão 10/100/1000Base-T, com 2Gb/s, por porta, em modo *full duplex*.

Característica	Clientes	*Gateway*	Servidor Apache
Processador	Intel P8700 2.53GHz	Intel Atom330 1.60GHz	Intel T3400 2.16GHz
Memoria (Gb)	2	2	2
Interface de Rede	10/100/1000Base-T	2 x 10/100/1000Base-T	10/100/1000Base-T
Sistema Operativo	GNU Linux (Fedora 19)	GNU Linux (Fedora 19)	GNU Linux (Fedora 19)
Kernel	3.11.9-200 64bit	3.11.9-200 64bit	2.6.18-371.1.2 64bit

Quadro I: Principais características dos clientes, *Gateway* e servidor Apache.

Os vários equipamentos são interligados da seguinte forma:

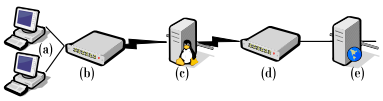


Fig. 1: Esquema simplificado da infraestrutura: (a) Clientes *Premium* e *Outros*, (b) *Switch A*, (c) *Gateway*, (d) *Switch B*, (e) Servidor Apache.

Na Figura 2 apresenta-se um esquema simplificado da "ligação física ao sistema de controlo em cadeia fechada". São representados, entre outros aspetos, o fluxo de pacotes, a sua classificação, as filas de espera e o controlador PID.

Os clientes HTTP efetuam os pedidos ao sistema, sendo os mesmos enviados para a *Gateway*. Os pacotes com os pedidos são colocados na fila de espera única da interface `eth1`, sendo posteriormente encaminhados (`ip_forwarding`) para a interface `eth0`. Quando os pacotes chegam à interface `eth0`, são classificados de acordo com o sítio a que se destinam, isto é: `Premium` ou `Outros`, existindo uma fila de espera associada a cada tipo de sítio. Caso os pacotes

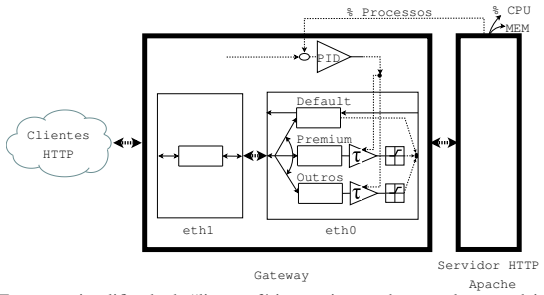


Fig. 2: Esquema simplificado da "ligação física ao sistema de controlo em cadeia fechada aplicado ao servidor HTTP Apache".

não sejam destinados a nenhum destes sítios, são colocados na fila de espera `Default`. Aos pacotes colocados nas filas `Premium` e `Outros` é, eventualmente, aplicado um atraso no seu envio (τ). O valor deste atraso é determinado pelo controlador PID e transdutor do sistema. O controlador recebe a informação enviada pelo servidor HTTP Apache, nomeadamente, a percentagem de processos `Ativos` e `Premium`, CPU e memória, em intervalos de cerca de 5 segundos. Os pacotes com os pedidos são finalmente enviados pela *Gateway* para o servidor HTTP Apache, para processamento e resposta.

B. Otimizações

No sentido de otimizar os recursos disponíveis dos equipamentos e devido ao facto de se poder registar um elevado volume de tráfego de rede e de pedidos HTTP, são realizadas algumas otimizações ao nível dos clientes, da *Gateway* e do servidor Apache.

1) Clientes HTTP: Após o arranque dos clientes é executado um *script*, cujo objetivo é optimizar os recursos disponíveis. Em seguida apresenta-se o mesmo.

Listagem 1: Clientes HTTP (`tune.sh`)

```
#!/bin/bash
ulimit -n 65535
ulimit -u 2048
ulimit -c unlimited
modprobe tcp_htcp
sysctl -w net.ipv4.tcp_congestion_control=htcp
ifconfig eth0 txqueuelen 10000
```

2) Servidor Apache: Neste trabalho é utilizada e modificada a versão 2.2.3 do servidor Apache. O código fonte do servidor Apache é alterado de modo a ser possível recolher informação do seu *Scoreboard*. Essa informação é posteriormente transmitida para a *Gateway*.

É executado um *script* igual ao dos clientes visando os mesmo objetivos.

O servidor Apache tem um endereço IP fixo principal e dois endereços fixos virtuais/*alias* (associados a *virtual host*). São, igualmente, alterados alguns dos parâmetros do servidor Apache, sendo os mais importantes os seguintes:

Listagem 2: Servidor Apache (`httpd.conf` - excerto)

```
Timeout 5
KeepAlive On
MaxKeepAliveRequests 0
KeepAliveTimeout 3
<IfModule prefork.c>
StartServers 256
MinSpareServers 5
MaxSpareServers 10
ServerLimit 256
MaxClients 256
MaxRequestsPerChild 0
</IfModule>
```

3) Gateway: O mecanismo de QoS implementado na *Gateway* utiliza como principais ferramentas o `tc`, disponibilizado pelo pacote `iproute`[12], o `HTB`[13] e o `NetEM`[14]. A *Gateway* depois de receber a informação do servidor Apache executa um *script*, cujo objetivo é implementar o mecanismo de QoS, nomeadamente, a classificação do tráfego, definição e aplicação das classes de serviço.

Tal como sucedeu nos clientes e no servidor Apache, após o arranque da *Gateway* é executado o *script* com o objetivo de optimizar os recursos disponíveis. O ficheiro `/etc/sysctl.conf` foi igualmente alterado, tendo sido introduzidas as seguintes linhas:

Listagem 3: *Gateway* (`/etc/sysctl.conf` - excerto)

```
net.ipv4.ip_forward = 1
net.ipv4.netfilter.ip_conntrack_max = 32768
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_orphan_retries = 1
net.ipv4.tcp_fin_timeout = 25
net.ipv4.tcp_max_orphans = 8192
net.ipv4.ip_local_port_range = 32768 61000
```

C. Execução e recolha de informação

Os clientes solicitam, através do programa `httperf`[15], ao servidor Apache a sua página de teste da seguinte forma: `httperf --hog --server <sitio> --uri /index.html --wsess=300000,50,2 --burst-len 5 --rate 50 --timeout 5`, em que <sitio> é o endereço IP dos sítios *Premium*/*Outros*.

É escolhida esta página HTML estática, com 5043 bytes, devido ao facto de se pretender que não existam conteúdos dinâmicos, ou acesso a sistemas de gestão de bases de dados (SGBD), os quais poderiam colocar limites/atrasos ao sistema, ou até formar um ponto de congestionamento. Também se pretendeu que a página escolhida não tivesse dimensões muito reduzidas, já que não permitiria gerar tráfego suficiente, nem ser muito grande, o que criaria tráfego em excesso.

III. Apresentação e discussão de resultados

No anexo A são apresentados os quadro resumo dos resultados obtidos pelos clientes HTTP (Quadro II e Quadro III), no sistema sem controlador e com o controlador PID.

A. Sistema sem controlador

O número total de processos Ativos é geralmente 100% (Figura 3), o que revela a saturação do servidor Apache. Analisando as percentagens dos processos-filho HTTP (de agora em diante referidos como processos), constata-se que não existe diferença significativa entre o sítio *Premium* e o sítio *Outros*. O sistema não faz qualquer diferenciação entre os pedidos do cliente do sítio *Premium* (de agora em diante referido como cliente *Premium*) e os pedidos do cliente do sítio *Outros* (de agora em diante referido como cliente *Outros*). Tendo em consideração a natureza estocática dos pedidos, existem instantes em que a percentagem de processos do sítio *Outros* é superior à percentagem de processos do sítio *Premium* e vice-versa. Este é o comportamento esperado do sistema, já que o servidor tem capacidade para receber/processar os pedidos dos clientes e está configurado para satisfazer os mesmos conforme vão ocorrendo (*First In First Out* - FIFO).

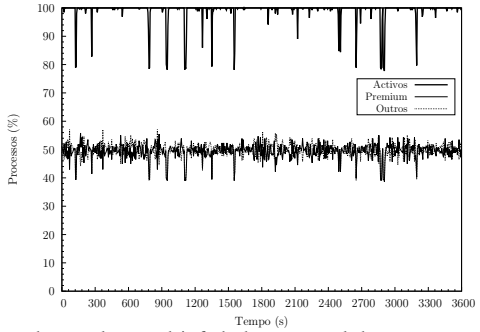


Fig. 3: Sistema de controlo em cadeia fechada sem controlador e com saturação. Servidor Apache: Percentagem absoluta de processos.

B. Sistema com controlador PID

Em seguida, vamos analisar o comportamento do sistema de controlo em cadeia fechada com um controlador PID discreto[4]. As constantes estão definidas do seguinte modo: $k_p = 5, k_i = 3, k_d = 8$, com um tempo de amostragem de 5s. A taxa de sessões é de 50 por segundo (50ss) e cada sessão efectua 50 pedidos. É de salientar que a definição das constantes indicadas resulta de um processo de afinação em que, por exemplo, para valores baixos de k_p (< 2) o sistema não consegue fazer a diferenciação de serviço, já que o cliente *Outros* exibe percentagens relativas de processos ativos superiores ao objetivo e, para valores elevados de k_p (> 10) os dois clientes são penalizados e o sistema oscila permanentemente.

As percentagens relativas (PR) de processos são calculadas da seguinte forma:

$$\text{PR}_{\text{Proc. } \textit{<Premium\|Outros>}}\ (\%) = \frac{\text{Proc. } \textit{<Premium\|Outros>} \text{ (n)}}{\text{Proc. } \textit{Ativos} \text{ (n)}} \cdot 100$$

1) Erro nulo: O objetivo (*setpoint*) definido neste caso é: 80% da capacidade do servidor, expressa em processos *Ativos*, a processar pedidos do sítio *Premium* e os remanescentes 20% a processar pedidos do sítio *Outros*.

A percentagem de processos *Ativos* é muito elevado, estando geralmente compreendida entre os 90% e os 100% (Figura 4). Regista-se uma clara diferenciação entre o cliente *Premium* e o cliente *Outros*.

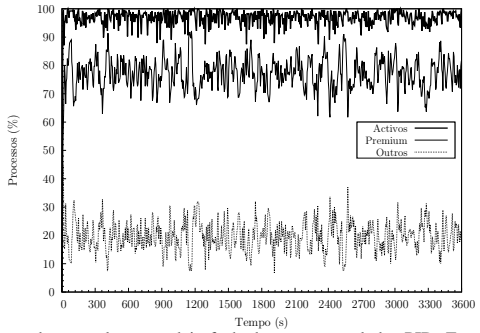


Fig. 4: Sistema de controlo em cadeia fechada com controlador PID. Erro nulo e 50ss. Servidor Apache: Percentagem absoluta de processos.

Na Figura 5 constatamos, tal como já era possível observar na Figura 4, que o objetivo definido foi atingido. A percentagem relativa de processos a executar pedidos do sítio *Premium* é tendencialmente 80% e do sítio *Outros* é tendencialmente 20%. A natureza estocástica dos pedidos HTTP não permite atingir, na perfeição, o objetivo definido, pelo que este comportamento do sistema não está relacionado com o seu incorreto funcionamento.

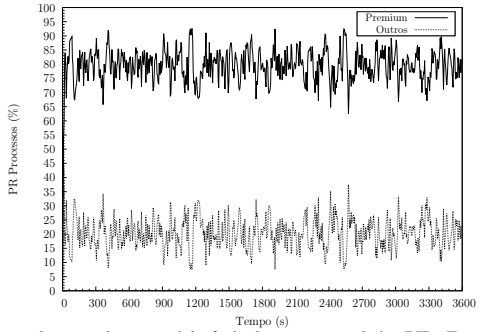


Fig. 5: Sistema de controlo em cadeia fechada com controlador PID. Erro nulo e 50ss. Servidor Apache: Percentagem relativa de processos Ativos.

No que diz respeito ao erro, o mesmo está geralmente compreendido entre os -10% e os +10%, o que é bom se tivermos em consideração a carga imposta ao servidor e a variabilidade dos pedidos.

O sistema consegue ajustar dinamicamente o atraso nos pacotes de modo a atingir o objetivo definido. Tal como esperado, é quase sempre imposto exclusivamente atraso ao pacotes do cliente *Outros* (Figura 6). Existem dois momentos (perto dos instantes t_{1200} e t_{2400}) em que também é imposto atraso aos pacotes do cliente *Premium*, mas tal facto não está associado a uma efetiva necessidade de atraso nesses pacotes. Convém mencionar que no código-fonte do sistema foram implementadas algumas funcionalidades para testar o sistema face ao *Seguimento de referência* e à *Rejeição de perturbações*. Nestas metodologias, o período de observação de 3600s é dividido em três intervalos de 1200s. No final destes intervalos é realizado um *reset* do controlador PID de modo a termos um sistema sem memória/histórico. Esta situação, normalmente indesejada, pode ser encarada como uma perturbação do sistema, mas que atesta a robustez do mesmo, já que ele reage de forma rápida e correta, impondo novamente os atrasos necessários com vista a obtenção dos objetivos traçados.

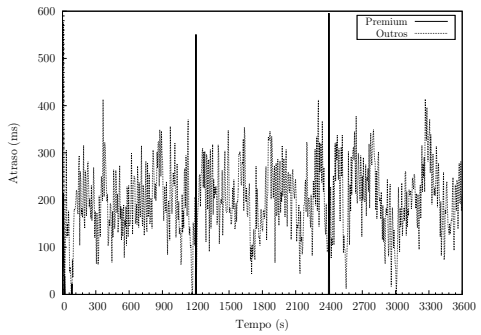


Fig. 6: Sistema de controlo em cadeia fechada com controlador PID. Erro nulo e 50ss. Gateway: Atraso.

A taxa de ligações (lig./s), nos dois clientes, foi semelhante ao sistema sem controlador com saturação. Contudo, no que diz respeito à taxa de pedidos e à taxa média de respostas o sistema exibe um comportamento que permite atestar a ação

do controlador, já que nos clientes *Premium* e *Outros* a taxa de pedidos e respostas é inferior ao sistema sem controlador com saturação. É assim percetível a ação do sistema, em particular sobre o cliente *Outros*, mais concretamente através da imposição de atrasos aos seus pacotes, conforme constatado na Figura 6.

Não houve diferença significativa na taxa de ligações entre o cliente *Premium* e o cliente *Outros*: 50,8 e 53,4, respectivamente. No entanto, tendo em consideração o que é referido anteriormente, já se registou uma diferença expressiva na taxa de pedidos (ped./s): 439,9 no cliente *Premium* e 84,4 no cliente *Outros*. Esta diferença justifica-se se tivermos em consideração os valores obtidos na relação repostas por ligação. No caso do cliente *Premium* este valor foi de 46,963, enquanto no cliente *Outros* foi apenas 16,840. Assim sendo o cliente *Premium* fez, em média, mais pedidos e obteve mais respostas.

Como resultado de uma maior taxa de pedidos do cliente *Premium* e deste ser privilegiado relativamente ao cliente *Outros*, também obteve uma maior taxa média de respostas (resp./s): 426,4 contra 61,8 no cliente *Outros*.

2) Rejeição de perturbações: Na Figura 7 está representada a percentagem de processos quando existe uma perturbação no instante t_{1200} e a reposição das condições iniciais no instante t_{2400}. A perturbação consiste na introdução de pedidos do cliente *Outros*. O objetivo é verificar como reage o sistema ao aparecimento súbito de uma elevada carga de pedidos do cliente *Outros*, quando este só estava a processar pedidos do cliente *Premium* e subitamente os pedidos do cliente *Outros* desaparecem e o sistema volta a ter exclusivamente pedidos do cliente *Premium*.

Como é natural, nos intervalos t_0 até t_{1200} e perto t_{2400} até t_{3600} existe uma coincidência total entre a percentagem de processos *Ativos* e processos *Premium*, já que o servidor Apache está exclusivamente a processar pedidos do cliente *Premium*. Deve ser salientado que nestes intervalos o sistema tem o controlador ativo, pelo que tenta manter a percentagem relativa de processos *Premium* perto dos 80% (objetivo), mesmo que não existam pedidos do cliente *Outros*. Essa é a razão pela qual o número de processos *Ativos* não se aproxima dos 100%, com excepção no arranque do sistema. Quando começam a surgir pedidos do cliente *Outros*, a percentagem de processos *Ativos* é geralmente superior aos 95%.

É possível constatar que o sistema consegue se adaptar rapidamente após o início dos pedidos do cliente *Outros*, de modo a manter o objetivo traçado: 80% da capacidade do servidor, expressa em processos *Ativos*, a processar pedidos do sítio *Premium* e os remanescentes 20% a processar pedidos do sítio *Outros*. Existe um momento inicial, após a perturbação, em que a percentagem dos processos *Premium* desce até cerca de 40%, mas recupera rapidamente. Esta redução resulta do aumento do atraso imposto aos pacotes destinados ao sítio *Premium*, conforme veremos adiante. O aumento do atraso deve-se ao facto da percentagem relativa dos processos *Premium* ser superior aos 80%, o que não é crítico enquanto o servidor só está a processar os seus pedidos, mas o é após existirem pedidos para o sítio *Outros*. O sistema tem de forçar o cumprimento dos objetivos, mesmo que isso implique o aumento temporário do atraso dos pacotes destinados ao sítio *Premium*.

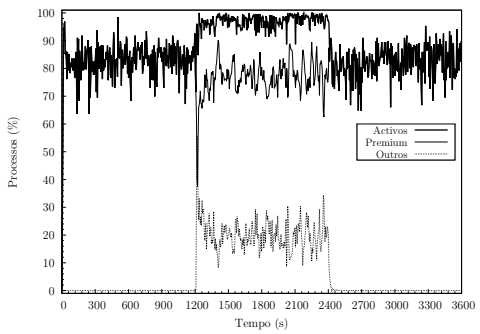


Fig. 7: Sistema de controlo em cadeia fechada com controlador PID. Rejeição de perturbações e 50ss. Servidor Apache: Percentagem absoluta de processos.

Nos intervalos t_0 a t_{1200} e perto t_{2400} a t_{3600} regista-se um erro de +20% e -20% relativamente ao objetivo do sítios *Premium* e *Outros*, respectivamente (Figura 8). Contudo, este erro não resulta do incorreto funcionamento do sistema mas do facto de não existir, nesses intervalos, pedidos destinados ao sítio *Outros*. Nos segundos imediatamente após o instante t_{1200} regista-se um erro positivo do processos do sítio *Outros* e negativo do processos do sítio *Premium*. Este erro resulta da ação do sistema ao detetar que, no instante t_{1200}, existe uma percentagem relativa superior à definida para os processos do sítio *Premium* e uma percentagem relativa inferior à definida para os processos do sítio *Outros*, o que força o sistema a impor um atraso aos pacotes destinados ao sítio *Premium* nesse instante (Figura 9).

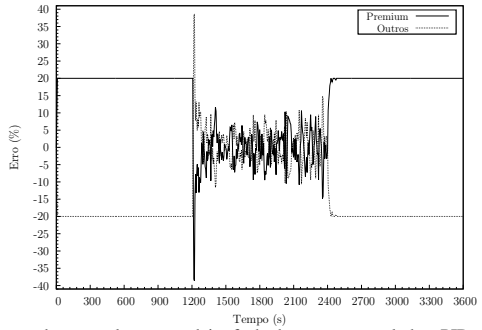


Fig. 8: Sistema de controlo em cadeia fechada com controlador PID. Rejeição de perturbações e 50ss. Servidor Apache: Erro.

Se atentarmos ao atraso imposto aos pacotes constatamos que:

- Até ao instante t_{1200} é imposto um atraso de cerca de 400ms aos pacotes destinados ao sítio *Premium*. O intuito do sistema é forçar o cumprimento do objetivo de 80% de processos HTTP *Ativos* a processar pedidos do sítio *Premium* e impedir que os mesmos "monopolizem" o servidor. Caso se registasse essa saturação com pedidos *Premium*, o sistema podia levar muito tempo a estabilizar após o aparecimento dos primeiros pedidos do cliente *Outros*. Conforme foi constatado anteriormente, o sistema consegue atingir os objetivos definidos muito rapidamente;
- No instante t_{1200} é imposto um atraso de quase 1000ms ao pacotes destinados ao sítio *Premium*, devido às razões explicadas anteriormente;
- Entre o instante t_{1200} e o instante t_{2400} é unicamente

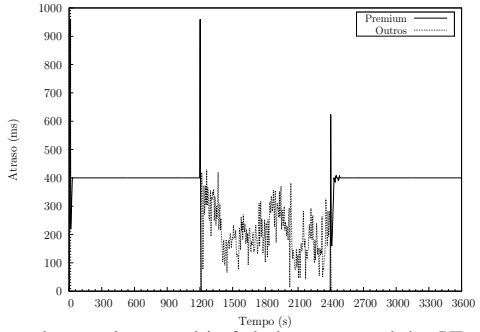


Fig. 9: Sistema de controlo em cadeia fechada com controlador PID. Rejeição de perturbações e 50ss. Gateway: Atraso.

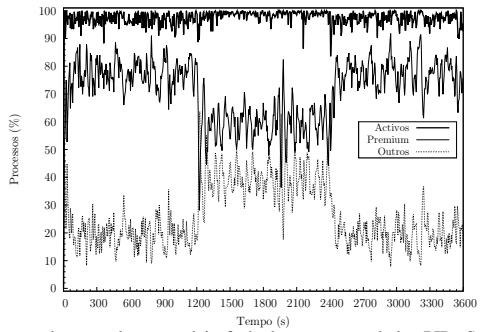


Fig. 10: Sistema de controlo em cadeia fechada com controlador PID. Seguimento de referência e 50ss. Servidor Apache: Percentagem absoluta de processos.

imposto limite aos pacotes destinados ao sítio *Outros*, sendo o atraso variável. Mais uma vez se constata o comportamento dinâmico do sistema;

- Após o instante t_{2400} e devido ao facto de não existirem novamente pedidos destinados ao sítio *Outros*, só são impostos atrasos aos pacotes destinados ao sítio *Premium*, sendo este atraso maior no instante t_{2400}. Este maior atraso permite ao sistema atingir mais rapidamente o objetivo definido.

Tal como esperado, no que diz respeito ao tráfego de rede, existem três zonas distintas, as quais coincidem com os intervalos em que só existem pedidos destinados ao sítio *Premium* (instante t_0 a instante t_{1200} e instante t_{2400} a instante t_{3600}) e o intervalo em que os pedidos destinados ao dois sítios coexistem (instante t_{1200} a instante t_{2400}). Como é lógico, neste sistema, o tráfego de rede, salvo algum fator externo (que não se regista) resulta basicamente do volume de informação a transferir dos pedidos dos clientes e das respostas do servidor. Assim sendo, é natural que o tráfego com os dois clientes ativos se traduza em mais pedidos e respostas, o que resulta num aumento do tráfego de rede.

No caso do cliente *Premium*, não é possível fazer uma análise comparativa relativamente às observações anteriores já que assentam em pressupostos distintos, nomeadamente a co-existência permanente de pedidos dos dois clientes. No que diz respeito ao cliente *Outros* é possível fazer essa análise, já que no período em que efectou pedidos, os mesmos coexistem com os pedidos do cliente *Premium* e com o mesmo objetivo. A taxa de ligações (lig./s) do cliente *Outros* é muito semelhante ao *Erro nulo*: 53,3 e 53,4, respectivamente. Também a taxa de pedidos (ped./s) é bastante semelhante: 86,1 contra 84,4. O mesmo sucede para a taxa média de respostas (resp./s): 64,2 contra 61,8. É, assim, validado o comportamento do sistema durante o período em que os pedidos coexistiam.

3) Seguimento de Referência: Neste caso, o objetivo é variável, isto é: em determinados intervalos o objetivo é 80% da capacidade do servidor, expressa em processos *Ativos*, a processar pedidos do sítio *Premium* e os remanescentes 20% a processar pedidos do sítio *Outros* (instante t_0 a instante t_{1200} e instante t_{2400} a instante t_{3600}) e noutro intervalo o objetivo é 60%/40% (instante t_{1200} a instante t_{2400}).

Tal como em casos anteriores, constata-se a tendência para a saturação do servidor Apache (Figura 10). A percentagem de processos HTTP *Ativos* está geralmente compreendida entre os 90% e os 100%. Tal como na *Rejeição de perturbações*, é possível observar três zonas distintas, estando as mesmas relacionadas com o objetivo definido: 80%/20% e 60%/40%.

É possível notar que, mais uma vez, o sistema funcionou como pretendido, ou seja: cumpre os objetivos definidos, é rápido a reagir às alterações e é estável (Figura 11). À semelhança da *Rejeição de perturbações*, perto do instante t_{1200} as percentagens denotam grande diferença relativamente ao pretendido mas, mais uma vez, corresponde ao instante em que o sistema se está a adaptar à alteração dos objetivos, mais concretamente à redução das percentagens relativas do cliente *Premium* e aumento das percentagens relativas do cliente *Outros*.

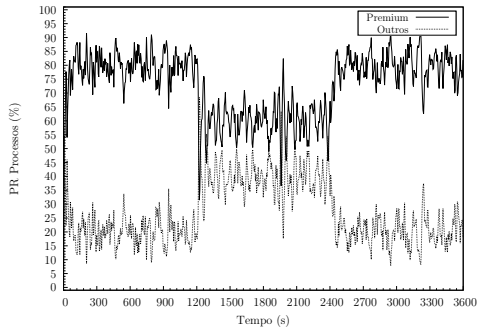


Fig. 11: Sistema de controlo em cadeia fechada com controlador PID. Seguimento de referência e 50ss. Servidor Apache: Percentagem relativa de processos Ativos.

De um modo geral, o erro das percentagens dos processos está compreendido num intervalo aceitável: -15% e +15%. Existem quatro excepções:

- Instante t_{1200}: Adaptação do sistema aos novos objetivos, $60\%/40\%$;
- Perto do instante t_{2000}: Sem razão específica. Resulta da natureza estocática dos pedidos HTTP;
- Instante t_{2400}: Adaptação do sistema aos novos objetivos, $80\%/20\%$;
- Perto do instante t_{3200}: Sem razão específica. Resulta da natureza estocática dos pedidos HTTP.

Entre o instante t_{1200} e o instante t_{2400} (intervalo com o objectivo $60\%/40\%$) são aplicados pontualmente pequenos atrasos aos pacotes destinados ao sítio *Premium*, facto que não sucede fora deste intervalo. Este situação pode ser explicada, pelo facto do objetivo dos processos *Premium* não diferir muito do objetivo dos processos *Outros*, 60% e 40% respectivamente. Devido à natureza variável dos pedidos, bem como ao facto de uma sessão poder fazer mais de um pedido (o que ocupa um processo durante um período de tempo não determinado), o sistema necessitou de aplicar atrasos aos pacotes destinados

aos dois sítios.

No que diz respeito ao tráfego de rede, e ao contrário do que sucede com no caso da *Rejeição de perturbações*, não se registam zonas distintas no período de observação. O mesmo está compreendido entre os 10Mbit/s e os 35Mbit/s, com maior frequência entre os 15Mbit/s e os 30Mbit/s, pelo que a alteração dos objetivos durante o período de observação não teve influência neste aspeto.

O cliente *Outros* tem uma taxa de ligações superior ao cliente *Premium* (104,1%), mas apresenta uma taxa de pedidos e uma taxa média de repostas bem inferior, 30,3% e 26,1%, respectivamente.

Por fim, convém salientar que nem o CPU nem a memória ou o tráfego de rede foram fatores limitativos nos resultados apresentados, já que em todos os casos durante o período de observação:

- O processador do servidor Apache esteve sempre mais de 80% *idle*;
- A memória utilizada do servidor Apache foi sempre muito estável, rondando os 512 MB o que representa cerca de 25% da capacidade do servidor (2 GB). É de mencionar que na configuração do servidor HTTP Apache o parâmetro `StartServers` é igual ao parâmetro `MaxClients` e `ServerLimit`: 256, pelo que o servidor inicia a sua atividade logo com a capacidade máxima de processos;
- O tráfego de rede nas interfaces da *Gateway* foi sempre inferior a 60 Mb/s, bastante inferior à sua capacidade: 1 Gb/s.

O programa *httperf* reportou que todas as respostas foram do tipo 2xx, ou seja: "Sucesso", o que indica que a ação foi recebida com sucesso, compreendida e aceite. Nunca se registaram erros devido à falta de descritores de ficheiros, falta de portas TCP disponíveis ou exaustão da tabela de descritores de ficheiros do sistema. Este aspeto é muito relevante, pois caso algum destes contadores seja diferente de zero, revela incapacidade do cliente em criar e manter a carga de pedidos ao servidor HTTP Apache, o que inviabiliza os resultados. Não existem, igualmente, outro tipo de erros, o que justificaria avaliar a sua razão.

IV. Conclusões

Neste trabalho é definido e implementado um sistema original de controlo em cadeia fechada de QoS, aplicado ao servidor HTTP Apache. Este sistema implementa uma ferramenta robusta e estável de recolha e análise de informação, bem como a gestão de recursos dinâmicos. Por fim, é comprovada a sua capacidade de realizar a diferenciação de serviço pretendida entre as duas classes de sítios: *Premium* e *Outros*. Esta qualidade é comprovada para o *Erro nulo*, *Rejeição de perturbações* e *Seguimento de referência*.

Existem, no entanto, muitas linhas de trabalho em aberto as quais, julgamos, permitirão ampliar a aplicabilidade do sistema e facilitar a sua integração com o servidor HTTP Apache. Como principais linhas de trabalho futuro podemos mencionar:

- Modificar o sistema de modo a ser possível alterar, em tempo de execução, a percentagem de processos destinados ao sítio `Premium`.
- Melhorar a gestão da fila de espera. Este objetivo pode ser atingido definindo uma fila de espera dinâmica (de comprimento variável) e registo do tempo que os pedidos aguardam serviço.
- Equacionar a alteração deste sistema para comportar duas vertentes: a priorização de clientes privilegiados e/ou de sítios privilegiados. Neste trabalho foi apenas abordada a vertente sítios privilegiados.
- Determinar uma metodologia para afinação das constantes do controlador PID do sistema (k_p, k_i, k_p).

Anexo A

Parâmetro	Sistema sem controlador	
	`Premium`	`Outros`
Taxa de ligações (a)	50,2	50,2
Taxa de pedidos (b)	488,7	490,1
Taxa média de respostas (c)	487,4	489,2
Respostas por ligação (d)	9,956	9,957

(a) lig./s, (b) ped./s, (c) resp./s, (d) resp./lig.

Quadro II: Quadro resumo dos clientes HTTP. Sistema de controlo em cadeia fechada sem controlador. 50ss. Erro nulo.

Parâmetro	Sistema com controlador PID					
	Erro nulo		Rej. Pertub.		Seg. Ref.	
	`Premium`	`Outros`	`Premium`	`Outros`	`Premium`	`Outros`
Taxa de ligações (a)	50,8	53,4	66,8	53,3	50,8	52,9
Taxa de pedidos (b)	439,9	84,4	338,4	86,1	403,3	122,3
Taxa média de respostas (c)	426,4	61,8	283,4	64,2	389,9	101,6
Respostas por ligação (d)	46,963	16,840	13,801	17,682	46,677	25,509

(a) lig./s, (b) ped./s, (c) resp./s, (d) resp./lig.

Quadro III: Quadro resumo dos clientes HTTP. Sistema de controlo em cadeia fechada com controlador PID (k_p=5,k_i=3,k_d=8). 50ss.

Bibliografia

[1] X. Huang, "UsageQoS: Estimating the QoS of Web services through online user communities," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 1, p. 1, 2013.

[2] M. A. Serhani, Y. Atif, and A. Benharref, "Towards an Adaptive QoS-driven Monitoring of Cloud SaaS," *Int. J. Grid Util. Comput.*, vol. 5, no. 4, p. 263–277, 2014.

[3] K. H. Chan and X. Chu, "Using Fuzzy PI Controller to Provide QoS on Web Servers," *Jornal*, 2007.

[4] J. Hellerstein *et al.*, *Feedback Control of Computing Systems*. John Wiley & Sons IEE Press, 2004.

[5] M. D. A. M. J. Almedia, P. Cao, J. A. M. D. A. Manikutty, and P. Cao, "Providing differentiated levels of service in web content hosting," in *Proc. First Workshop Internet Server Performance*, 1998.

[6] L. Eggert and J. Heidemann, "Application-level differentiated services for Web servers," *World Wide Web*, vol. 2, no. 3, p. 133–142, 1999.

[7] T. F. Abdelzaher and K. G. Shin, "Qos provisioning with qcontracts in web and multimedia servers," in *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*. IEEE, 1999, p. 44–53.

[8] J. M. Blanquer *et al.*, "QoS for internet services: done right," in *Proceedings of the 11th workshop on ACM SIGOPS European workshop*. ACM, 2004, p. 8.

[9] M. Loudini and S. Rezig, "Enhancing Web Server Relative Delay Services by an Integrated SA-fuzzy Logic Controller," *Int. J. Web Eng. Technol.*, vol. 8, no. 1, p. 27–57, Mar. 2013.

[10] K. H. Chan and X. Chu, "Design of a Fuzzy PI Controller to Guarantee Proportional Delay Differentiation on Web Servers," in *Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management*, ser. AAIM '07. Springer-Verlag, 2007, p. 389–398.

[11] P. Pivoňka, "Comparative analysis of fuzzy PI/PD/PID controller based on classical PID controller approach," in *Proceedings of the 2002 IEEE International Conference on Fuzzy systems*, 2002, p. 541–546.

[12] Linux Foundation, "iproute2," http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2, acedido 10 agosto 2015.

[13] M. Devera and D. Cohen, "HTB Linux queuing discipline manual - user guide," http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm, acedido 10 agosto 2015.

[14] Linux Foundation, "netem," http://www.linuxfoundation.org/collaborate/workgroups/networking/netem, acedido 10 agosto 2015.

[15] Hewlett-Packard Research Laboratories, "httperf," http://www.hpl.hp.com/research/linux/httperf/, acedido 10 agosto 2015.

TVPulse: Improvements on detecting TV highlights in Social Networks using metadata and semantic similarity

Afonso Vilaça
Instituto de Telecomunicações
Aveiro, Portugal 3810-193
Email: afonso.vilaca@av.it.pt

Mário Antunes
Instituto de Telecomunicações
Aveiro, Portugal 3810-193
Email: mario.antunes@av.it.pt

Diogo Gomes
Instituto de Telecomunicações
Universidade de Aveiro
Aveiro, Portugal 3810-193
Email: dgomes@ua.pt

***Abstract*—Sharing live experiences in social networks is a growing trend. That includes posting comments and sentiments about TV programs. Automatic detection of messages with contents related to TV opens new opportunities for the industry of entertainment information.**

This paper describes a system that detects TV highlights in one of the most important social networks - Twitter. Combining Twitter's messages and information from an Electronic Programming Guide (EPG) enriched with external metadata we built a model that matches tweets with TV programs with an accuracy over 80%. Our model required the construction of semantic profiles for the Portuguese language. These semantic profiles are used to identify the most representative tweets as highlights of a TV program. Measuring semantic similarity with those tweets it is possible to gather other messages within the same context. This strategy improves the recall of the detection. In addition we developed a method to automatically gather other related web resources, namely Youtube videos.

I. Introduction

The number of social networks users worldwide is increasing every year, with 2.078 billion active accounts in 2015[1]. Twitter is a microblogging network, characterized by its 140-character messages, called tweets. It has more than 302 million active users (May 2015)[2]. The simplicity by which users can share their experiences live, including TV watching, makes it ideal for our use case.

Tweets content recognition and detection of messages related to TV programs would provide a sense of the impact of those TV shows on the network. This is of particular interest for the TV industry[3]. The frequency of tweets related to a given TV show can tell one not only the most popular programs, but also the highlights of those programs. Web and interactive TV applications can provide to the users an experience that joins television with social networks. The automation of the detection makes this service applicable on many scenarios, with real time responses. A particular use of content recognition from tweets is the summarization of TV programs into videos with the highlighted moments and the tweets with more impact. Relevant tweets offers a textual report of the show from the users reactions. Other option is to offer web resources related to the detected contents.

A simple way to get TV programs information is from the Electronic Program Guide (EPG). That information can be enriched with information from external sources.

In this paper we describe a system built to extract and store information from Twitter and from publicly available EPG's, build a semantic profile for the extracted contents and match tweets with TV programs. It is important to state that our work focus is on the Portuguese language, which ultimately distances ourselves from the state of the art. In that way, this becomes a novel work, including the construction of semantic profiles for Portuguese language.

II. State of the Art

The first step in text mining projects is text processing[1]. Several techniques are reported in literature. One of the first procedures is usually to lower-case all words. In microblogging messages, grammar rules for capitalization and accentuation are not followed by users, and this technique is justifiable. For the same reason, accents are also usually removed. Terms removal is also applied in many applications, and what is removed depends on the goal. The most common words, also called stop words, act as noise in sentences and their removal is advisable. Those words may be identifiable from the collected dataset or can be retrieved from a stored list built for the specific language. Other terms may or not have interest to the application being developed. Examples of those terms are swear words, short words, hyper links, punctuation, numbers or *emoticons*. Typically, terms are also stemmed in order to get a homogenization of their stemmed versions. Many stemming algorithms exists, some more general, other optimized to specific languages. In Portuguese language a known stemming algorithm is the RSLP stemmer[2]. All text filtering options should be optimized for the specific case study.

[1] http://wearesocial.net/blog/2015/01/digital-social-mobile-worldwide-2015

[2] http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

[3] https://www.ibm.com/developerworks/community/blogs/025bf606-020a-48e9-89bf-99adda13e9b1/entry/by_the_numbers_social_media_impacts_the_entertainment_industry

A second choice to be made in text mining, after the filtering process, are the features to be used[1], [3]. An obvious choice are the terms that passed through the filtering process. Other features can be sequence of consecutive terms, like bigrams (two) or trigrams (three). Social networks have their own syntax, where some characters in the beginning of the word give a particular meaning to it. The most popular in Twitter are *hashtags* (started by '#') and user names (started by '@'). Using these special terms as features is usually valuable for many machine learning problems.

Words and features can be grouped according to their grammatical properties. Part-of-speech (POS) tagging on chats and microblog messages is a big challenge of Natural Language Processing (NLP), considering the large use of typing errors, abbreviations, dialect variations and use of slang words and peculiar vocabulary. There is however good progresses mainly for English. We highlight the work of Owoputi et al.[4]. Their model of POS tagging is a first-order maximum entropy Markov model (MEMM) that achieves an accuracy of 93%.

Both information retrieval and clustering techniques are necessary to measure distances between words, sentences or documents. These measures can be just the count of equal words, an edit distance, that represents the cost that takes to transform a word into another. Or even semantic distance, based on statistics of co-occurrences of words in the same document, sentence or window (sequence of words with a fixed length). A popular edit distance is the Levenshtein distance[5]. There are other posterior measures derived from this one. They are useful in string matching where words with small edit variations between them are assumed to belong to the same family or the difference be caused by a typing error. To measure similarity of words in terms of their meaning it is necessary to associated them to concepts. Mohammad and Hirst[6] propose a semantic distance using distributional profiles (DP) of concepts. They not only build to each word a DP, but they also group words to represent concepts and create profiles from those concepts using bootstrapping. Concepts are inferred from the context and decisions are made based on concepts distances. They showed that distributional concept-distance measures outperformed word-distance measures on ranking word pairs.

Other approaches can be followed to tag tweets. Classification algorithms can be used when there is a tagged dataset. Cremonesi et al.[7] collected tweets from specific TV programs and movies pages and trained 1-class-SVM models. They achieved results on associating tweets to programs and movies with precision of 92% and recall of 65%.

Another work on detection of events on TV programs from Twitter messages, in Japanese, is Nakazawa et al. [8]. They associate tweets to TV programs based on hashtags and inspect their content to identify hot moments. They extract people names and keywords from the terms co-occurring with those names. Their success on tagging events is above 66.8%.

Semantic characterization of tweets is commonly used to classify or cluster them: Genc et al. [9], Abel et al. [10] and Ozdikis et al.[11]. The first work associate tweets to Wikipedia pages, the second extract contexts from messages to construct user profiles, the last one also focus on event detection. All of them show that semantic similarity is a better procedure to relate text messages in terms of their context, comparing with the standard methods.

In a preliminary work we present a first version of our system with a successful highlights detection[12]. We had however some limitations, mainly in the used of manually inserted metadata. Even so, the achieved precision was higher than 80%. We also inspected with a positive perspective the potential of the use of semantic similarity on relating tweets.

III. System's architecture

Fig. 1 is a diagram of the system's architecture. Tweets are collected from Twitter Search API[4] using a JAVA agent. Since we are interested in the Portuguese community, we only collect tweets created in Portugal, meaning that they're almost all written in Portuguese. A tweet is represented not only by its text, but also by other information, like its id, user information, time of its creation, place, number of re-tweets, *hashtags*, if it is a reply to someone, etc. TV programs are also collected programmaticaly from Sapo Services [5]. A TV program is represented by its title, channel name and acronym, start and end times, a description and a short description. Tweets are stored in a Cassandra database (for scalability) and TV programs in a MySQL database (for easier information retrieval). Additionally we add to TV programs related metadata to the database, extracted from Wikipedia pages. To do it we use some python modules: *wikipedia*[6], *requests*[7] and $beautifulsoup4$[8]. Searches are done on the Portuguese language, using programs titles and the word *Portugal*, since homonym programs exist in other countries. The information is extracted from pages tables and from specific sections with interesting indexed contents, like *Elenco* (cast) or *Apresentadores* (presenters). This procedure automates and eases the quest for extra information. The only human dependent part is the association of hashtags not directly derived from the program title. For the most popular TV programs we manually insert a list of related hashtags.

Each tweet or program information is processed using a text processing pipeline implemented in Python with the following transformations. The text is converted from upper cases to lower cases, removal of accents, punctuation, numbers, stop words and swear words, finally a stemming transformation based on the RSLP algorithm [2]. Stemming allows the homogenization of words from the same family. Each processed document (tweet or program) is tokenized into a bag of words (BOW).

From the BOW we extract their features. Unigrams, bigrams and trigrams are computed for tweets text and programs title,

[4]https://dev.twitter.com/overview/api
[5]https://store.services.sapo.pt/en/cat/catalog/other/meo-epg
[6]https://pypi.python.org/pypi/wikipedia/
[7]http://www.python-requests.org
[8]https://pypi.python.org/pypi/beautifulsoup4/4.4.0

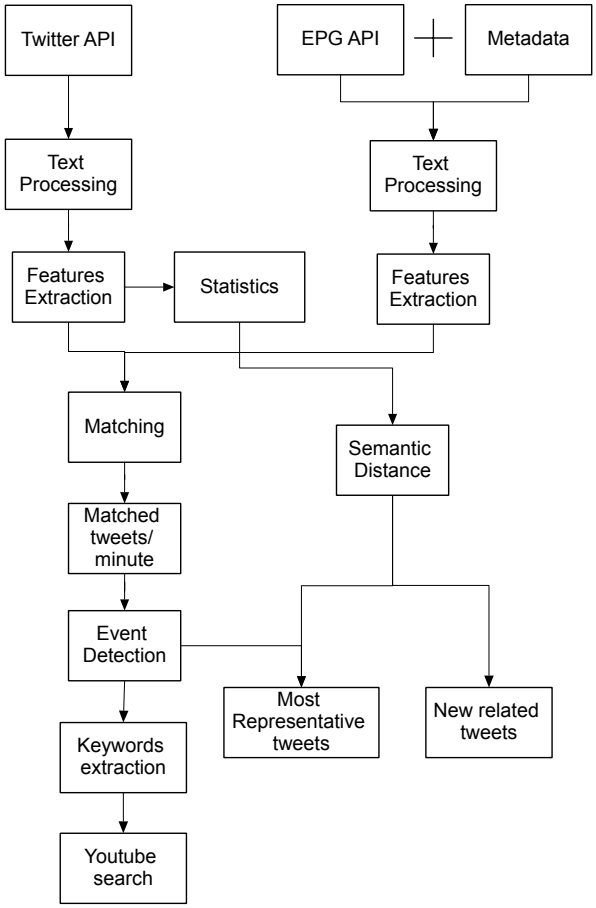


Fig. 1. *System diagram.*

description, short description, channel name and acronym and metadata. Tweets hashtags are also considered as features.

Portuguese distributional profiles are being extracted continually from each tweet. Terms frequencies and number of co-occurrences are stored in a MySQL database. From these statistics it is possible to build semantic features based on terms distributional profiles. To measure the distance of two words we use the cosine distance[6]:

$$Cos(w_1, w_2) = \frac{\sum_{w \in C(w_1) \cup C(w_2)} (P(w|w_1) \times P(w|w_2))}{\sqrt{\sum_{w \in C(w_1)} P(w|w_1)^2} \times \sqrt{\sum_{w \in C(w_2)} P(w|w_2)^2}} \quad (1)$$

where $C(w_i)$ is the set of words that co-occur with word w_i. The conditional probabilities are the relative frequencies on the word profile. Considering the high-dimensionality of the co-occurrences matrix, its reduction speeds up the process and may improve the accuracy, too. Two reduction methods can be used: the power law[13], also known as 80-20 rule, and the elbow method [14]. We apply both to study their influence.

To check if a tweet is related to a program or not, we developed a matching procedure. Co-occurrences of previously computed features in both sides are counted. A special treatment is given to *hashtags*. It is checked if the text of a tweet *hashtag* is in the program title (with spaces removed) and vice versa. This is worthy in the cases where there is no auxiliary *hashtags* table for the program. Some cases have required very specific functions such as the one that detects if a program is a football match. If it is the case, it looks in the tweet for a regular expression that represents a goal (*golo* in Portuguese): $\hat{}g+o+l+o+\$$. We empirically built a tree that gives a score to the tweet-program pair based on the matching results of each feature. That score discretely varies from 0 to 1, where 0 means that the tweet and the program are not related and 1 means that they are very related. 0.5 is used as minimum acceptable value to attribute a tweet to a program.

One of the goals of this work is to detect relevant events on TV programs with impact in Twitter. In order to achieved this we analyze matching results minute by minute during the program duration. Besides counting the number of matched tweets per minute, we compute a second derivative of that frequency ($f(m_{i+1}) - 2f(m_i) + f(m_{i-1})$). The following two measures allow us to decide if on that minute occurs a relevant event or not:

$$Event(m_i) = \begin{cases} True, & f''(m_i) \leq \overline{f''(m)} - \sigma_{f''(m)} \\ & \wedge f(m_i) > 5 \\ False, & otherwise \end{cases} \quad (2)$$

where $f(m_i)$ is the number of matched tweets for the minute m_i and $f''(m_i)$ the value of its second derivative. $\overline{f''(m)}$ is the average for all the minutes of the program and $\sigma_{f''(m)}$ the standard deviation.

To understand the topic of the event two approaches are followed. First, consists only in showing the most used words. Second, we try to identify the most representative tweets. This is achieved through semantic similarity. A graph is built, where nodes are tweets and edges are the semantic similarities between them (mean of words similarities). The most representative tweet is that which the sum of its edges is higher, i.e. the one closest to the graph centroid. Again in special cases, such as if the program is a football match, the number of tweets with reference to a goal is also computed to support the goal detection on that event.

An extra feature is added, that meets the goal of finding web resources related to a TV program and its highlights. The most popular unigrams and bigrams are used as keywords on Youtube searches. We do it using the Youtube Data API[9]. Each extracted keyword or keyword pair is joined to the program title and channel name for a search with a proper filter: the time a video was published must be after the program start time; its duration must be less than 20 minutes. For validation, we do a matching between titles from the retrieved videos and the keywords. At the end the remaining videos are rated according to a linear function of their number of views, likes, dislikes, comments and favorites, where each of the variables have a different weight. The proposed videos are the better rated: maximum of 3 for an event and 10 for the entire program. From those videos we also extract their comments. They can be used to enrich the TV program vocabulary and be a resource for future research work.

[9]https://developers.google.com/youtube/v3/?hl=en

Our work is made available to others through a public API. The API implemented using CherryPy [10] responds to queries with matched tweet-program pairs for a given time interval, informing when an event occurs and providing the related Youtube videos. The channel acronym or the program title can be specified. It is also possible to choose only tweets with a specific *hashtag* and to control the system accuracy by specifying the minimum matching score acceptable.

As we have shown in [12], generally tweets related to a same TV program have higher similarity than tweets related to two distinct ones. Sustained on this fact, we applied a second strategy to improve the matching recall, capturing tweets not necessarily with words present on programs EPG and metadata, but still belonging to the same vocabulary. This procedure is divided in two parts. First we collect tweets related to the TV program using the method described above. We use a minimum score of 0.6 to ensure high precision on the matched tweets. All the drawn tweets are used to build a proper semantic for the TV program, based on words frequencies and co-occurrences. A graph of semantic similarities is built, like the ones used after event detection, now for the whole program. A set of representative tweets is selected and the rejected tweets are compared to them in terms of semantic similarity. This part is in being tested. We do not achieve the optimal number of representative tweets and the best threshold on semantic similarity yet. Even so some results are already presented.

IV. Results

The frequency of collected tweets varies during the day and also depends on the day of the week. In a week day 3 distinct periods are distinguishable. From approximately 3 a.m. to 7 a.m. the activity is very low, less than 20 tweets/min. From 8 a.m. to 5 p.m. the frequency is typically between 25 and 60 tweets/min. In the evening the activity is higher, achieving 200 tweets/min, mainly between 9 p.m. and 11 p.m. On the weekend the activity is higher, the frequency varies from 50 to 150 tweets/min between 10 a.m. and 18 p.m. and from 100 to 200 tweets/min between 18 p.m. and 12 a.m. We also notice the existence of sudden peaks in certain minutes. For example, a sudden growth of 200 tweets/min from one minute to other, returning to the same value in two minutes. We realize that some higher peaks coincide with moments in which the Portuguese national team of football scores a goal. We also analyzed the most used *hashtags*. In that top, few were related with TV. Those *hashtags* were mainly about football matches and entertainment shows or contests, typically aired on Sunday evening.

We focus our tweet-program matching analysis on those apparently popular programs, based on the tweets frequency and popular *hashtags*. In Fig. 2 it is presented the number of matched tweets per minute for a program called *Ídolos*. The detected events are marked. Twelve hot moments were identified. Based on the most representative tweets, it was possible to relate each peak with a moment in the program. Analyzing several programs, we observed that the event detection is as successful as the popularity of the program in Twitter. In programs with an average of detected tweets bellow 5 tweets/min there are some small peaks where the relation to an event is dubious. In football matches, goals have a big impact and are always correctly detected.

[10]http://www.cherrypy.org

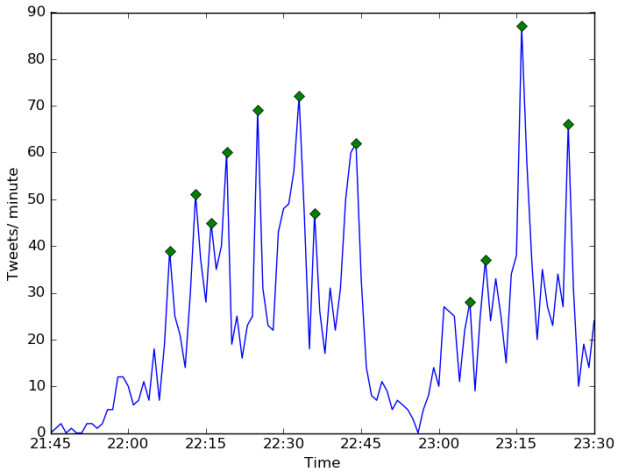


Fig. 2. *Matched tweets frequency for the TV program "Ídolos" on 12th May 2015. The green diamonds represent the detected events.*

TABLE I
Phase 1 matching analysis

TV Program	Avg. Freq. (tweets/min)	Precision
Got Talent Portugal	1.0	0.84
Dança com as Estrelas	3.5	0.95
Benfica x Setúbal	5.0	0.80
Ídolos	29.0	0.99

On the first phase of our work, when we were using manual inserted information as metadata, we computed the precision of our model by manual verification. We tested four programs: *Got Talent Portugal*, *Ídolos*, *Dança com as Estrelas* and *Benfica x Setúbal - Taça da Liga 1ª Meia Final*. The first three are entertainment contests and the last one is a football match for the Portuguese League Cup semi-final. We present the results in Table I. We observe that for the TV contests the precision grows with the frequency of matched tweets. We also see that the matching is less precise for the football match.

After the implementation of metadata extraction from Wikipedia, we compared its performance of the new method with the previous one. We computed not only the precision but also the recall. We considered 10 distinct and separated minutes during the simultaneous emission of two of the programs analyzed before, in a different day: *Ídolos* and *Dança com as Estrelas*. Tweets were manually tagged and after the matching we computed the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Table II shows the results. We can see that the popularity of the TV programs changed. From the first case to the second, the average frequency of matched tweets for *Ídolos* decreased

TABLE II
PHASE 2 MATCHING ANALYSIS

TV Program	Ídolos		DCAE	
Metadata	Manual	Wikipedia	Manual	Wikipedia
TP	20	20	62	77
TN	1507	1503	1407	1408
FP	0	4	2	1
FN	24	24	80	65
Precision	1	0.83	0.97	0.99
Recall	0.45	0.45	0.44	0.54

TABLE III
PHASE 3 MATCHING ANALYSIS

TV Program	Porto x Benfica	
Matching with similarity	No	Yes
Precision	0.95	0.91
Recall	0.44	0.62
F-measure	0.60	0.73

from 29.0 to 2.0 tweets/min and for *Daną com as Estrelas* increased from 3.5 to 6.4. The use of Wikipedia as metadata source, does not significantly influences the precision of *Daną com as Estrelas*. For *Ídolos* the precision drops but remains above 80%. The reason for this change is on the quantity of spurious terms that exist on Wikipedia pages, even on tables and on the selected sections. The quantity of those misleading terms change from page to page. We can also realize that the recall is low for both cases, meaning that half of related tweets are being ignored. Without using machine learning algorithms but gathering TV programs information from different sources we achieve results with precision values close to what the models described on the State of the Art show.

The ambition to improve the recall lead us to implement a solution that could take advantage from the semantic profile of a TV program. We analyzed this third approach on a classical football match in Portugal: *FC Porto x Benfica - Primeira Liga*. After the construction of the tweets similarity graph for this program, we measured the similarity of the neglected tweets with the 20 most representative tweets. From the 3200 with higher similarity, we calculated the similarity threshold that maximizes the F_1 score: $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

We achieve a value of 0.49 for similarity, corresponding to an F_1 score of 0.84. Again, we extracted a set of tweets from 10 distinct separated minutes, and evaluated the performance of our model with and without the use of semantic similarity on matching. From Table III we observe that although the precision lightly decreases, the recall substantially rises with the use of semantic similarity, increasing the F-measure. We realized that some detected tweets are only matched with a context understanding, e.g.: *"O melhor está a jogar"* or *"Jogamos mal pa c**... ganhamos 2-0. Jogamos bem.... perdemos"*. These messages are composed by words that are not found on the TV programs sources. Although it is not possible to take generic conclusions from this analysis, it gave us reasons to deeply explore the use of semantic similarity on the detection of tweets related to TV programs, namely by testing it with other programs and tuning some parameters, like the number of representative tweets and the similarity threshold.

Finally, by observing lists of suggested Youtube videos, from the programs mentioned above, we concluded that the automatic search is successful, retrieving related contents. The videos are always related to moments of the program (same or other episode) or with an highlighted person. For some of the detected moments no video may be retrieved or the retrieved video may not correspond to the event. However that is not happening on events with a big impact (large peaks).

V. CONCLUSION

We have built a solution based on state-of-art social media text mining techniques, applied to the Portuguese language and TV commenting habits. Albeit all the limitations of our system, we are able to successfully detect highlights in a TV program solely based on what viewers comment on Twitter. We improved our model, automating the gathering of metadata and starting a new approach by using semantic properties of the messages. Another extra was included, a successful automatic search of Youtube videos related to the TV programs and their main topics. Future work will focus on improving system performance in terms of matching results and processing time.

ACKNOWLEDGMENT

This work was made possible thanks to a grant by PT Inovação - Project TVPulse. The authors would also like to thank Univ. Aveiro Social iTV research group with whom have collaborated.

REFERENCES

[1] J. Perkins, *Python Text Processing with NLTK 2.0 Cookbook* Packt Publishing, 2010.

[2] V. M. Orengo and C. Huyck, *A Stemming Algorithm for the Portuguese Language* SPIRE Conference, Laguna de San Raphael, Chile, November 13-15, 2001.

[3] S. M. Weiss, N. Indurkhya and T. Zhang, *Fundamentals of Predictive Text Mining* Springer-Verlag London Limited, 2010.

[4] O. Owoputi, C. Dyer, K. Gimpel, N. Schneider and N. A. Smith *Improved part-of-speech tagging for online conversational text with word clusters* In Proceedings of NAACL, 2013.

[5] V. I. Levenshtein *Binary codes capable of correcting deletions, insertions, and reversals* Soviet Physics Doklady, Feb 1966.

[6] S. Mohammad and G. Hirst *Distributional measures of concept-distance: a task-oriented evaluatio* EMNLP '06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 2006.

[7] P. Cremonesi, R. Pagano, S. Pasquali and R. Turrin *TV Program Detection in Tweets* EuroITV'13, Como, Italy, June 24âĂŞ26, 2013.

[8] M. Nakazawa, M. Erdmann, H. Hoashi and C. Ono *Social Indexing of TV Programs:Detection and Labeling of Significant TV Scenes by Twitter Analysis* ï£ijWAINA '12 Proceedings of the 2012 26th International Conference on Advanced Information Networking and Applications Workshops, Fukuoka, Japan, Mar 2012.

[9] Y. Genc, Y. Sakamoto and J. V. Nickerson *Discovering Context: Classifying tweets through a semantic transform based on Wikipedia* Proceedings of the 6th International Conference, FAC 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011.

[10] F. Abel, Q. Gao, G.-J. Houben and K. Tao *Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web* 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29 âĂŞ June 2, 2011, Proceedings, Part II.

[11] O. Ozdikis, P. Senkul and H. Oguztuzun *Semantic expansion of hashtags for enhanced event detection in Twitter* Proceedings of VLDB 2012 Workshop on Online Social Systems, Istanbul, Turkey, Aug 31, 2012

[12] A. Vilaça, M. Antunes, D. Gomes, *TV-Pulse: detecting TV highlights in Social Networks* Proc. 10th ConfTele 2015 - Conference on Telecommunications, Aveiro, Portugal, Sep 2015.

[13] A. Clauset, C. R. Shalizi and M. E. J. Newman *Power-Law Distributions in Empirical Data* SIAM Review 51, 661-703, 2009.

[14] D. J. Ketchen and C. L. Shook *The Application of Cluster Analysis in Strategic Management Research: An Analysis and Critique* SIAM Review 51, 661-703, 2009.

Encaminhamento de chamadas VoIP em redes Peer-to-peer

António Daniel Lopes, Maria João Nicolau, António Costa, Joaquim Macedo, Alexandre Santos
Centro Algoritmi, Universidade do Minho, Portugal
Email: {a55778@alunos, joao@dsi, costa@di, macedo@di, alex@di}.uminho.pt

***Resumo*—A utilização da rede IP para transportar voz (VoIP) oferece potencialmente maior largura de banda e uma qualidade superior de chamada, com menores custos e melhor utilização de recursos. Contudo, a arquitetura cliente-servidor do VoIP tradicional tende a centralizar o registo de utilizadores em servidores específicos de acordo com a sua localização, e encaminha as chamadas sempre pelas melhores rotas IP entre servidores. Esta abordagem cria problemas de escala e torna o sistema mais vulnerável a ataques de negação de serviço. As redes *peer-to-peer*, pela sua natureza distribuída, podem contribuir para minimizar estes problemas, pois permitem armazenar a informação de utilizadores e respetivas localizações em vários nós. Os pedidos podem ser distribuídos por múltiplos servidores em vez de canalizados apenas para um e as chamadas de voz podem ser encaminhadas por múltiplos caminhos alternativos entre *peers*, assegurando a qualidade de serviço que nem sempre o melhor caminho definido pela rede pode oferecer.**

Neste trabalho propõe-se um protocolo de reencaminhamento automático das chamadas pela rede *peer-to-peer*, com um número ajustável de saltos, de modo a obter melhorias em termos de desempenho global da rede. O protocolo foi implementado em JAVA, numa rede *peer-to-peer* totalmente baseada em SIP. O facto do protocolo adotado para a criação e manutenção da rede *peer-to-peer* ser o SIP, facilitou a integração entre as duas componentes, uma vez que o SIP é utilizado em várias operações de sinalização requeridas pelo VoIP. A implementação JAVA foi testada em ambiente emulado com o emulador CORE, com uma topologia e vários cenários de teste, que permitiram comprovar que as alterações propostas permitem efetivamente acomodar mais chamadas com os mesmos recursos.

I. Introdução

O VoIP [1] pode ser desenvolvido de acordo com o paradigma "Cliente-Servidor", no entanto esta alternativa tem alguns problemas de escalabilidade. O grande número de utilizadores pode provocar congestionamento e sobrecarga, tanto na rede como nos próprios servidores. Uma melhor solução passa pela utilização de um sistema completamente distribuído para a implementação do VoIP, sendo uma das possibilidades a utilização de redes *peer-to-peer*.

As redes *Peer-to-Peer* (P2P) [2] são redes de *overlay* que podem ser estruturadas ou não estruturadas e que permitem localizar e aceder a recursos de forma distribuída e escalável, tendo por isso sido ao longo dos anos usadas para distribuição de conteúdos. Revelam-se pois bastante atrativas como suporte a comunicações de áudio e vídeo entre dois ou mais interlocutores. Nesse contexto a rede P2P forma-se com o intuito de fornecer simplesmente o mapeamento entre o URL do destinatário da chamada e a sua localização efetiva. O IETF (*Internet Engineering Task Force*) tem neste momento um grupo de trabalho, denominado P2PSIP Working Group [3] a desenvolver propostas com vista à normalização das redes P2P com sinalização SIP para uso em aplicações VoIP.

Neste trabalho pretende-se em primeiro lugar avaliar mecanismos que tirem o máximo partido das vantagens que o *overlay* oferece, nomeadamente o encaminhamento da *media*. Para isso propõe-se uma solução de encaminhamento de chamadas de voz a ser efetuado pelos *peers* do *overlay*. A solução proposta é implementada numa rede P2P de modo a poder testar o encaminhamento de dados e obter resultados experimentais que provem a sua validade.

O artigo está estruturado da seguinte forma. Após a secção I que introduz o contexto e define os objectivos, segue-se a secção II com alguns trabalhos relacionados. Na secção III é apresentada a rede *overlay* base e suas funcionalidades bem como o algoritmo de encaminhamento de chamadas de voz concebido para tirar partido da rede *peer-to-peer*. A secção IV discute a implementação em Java e as APIs adotadas, seguindo-se a secção V com os testes e resultados experimentais obtidos. Os vários cenários de teste foram desenhados de forma a comprovar as funcionalidades da aplicação desenvolvida e avaliar o seu desempenho. Por último, a secção VI apresenta as conclusões e propostas de trabalho futuro.

II. Trabalhos relacionados

O grupo de trabalho P2PSIP [3] foi criado pelo IETF com o objetivo de padronizar os protocolos necessários para criar redes *peer-to-peer*. Esta necessidade de criar um protocolo padrão pretende resolver as incompatibilidades atualmente existentes entre protocolos utilizados por diferentes fabricantes. Existem vários drafts ativos, nomeadamente o protocolo RELOAD, propostas especificas para a descoberta de recursos e serviços na rede, a DHT utilizada, e várias extensões ao protocolo RELOAD que lhe permitem funções adicionais. Recentemente, em janeiro de 2014, o draft do RELOAD passou a RFC [4].

O protocolo RELOAD (Resource LOcation And Discovery) é a proposta do IETF para a criação de um protocolo de sinalização flexível que possa ser utilizado com outros protocolos, desde que tenham os mesmos requisitos do SIP, por exemplo, o P2PSIP. Uma aplicação VoIP que utilize SIP pode utilizar o RELOAD para localizar outros *peers* no *overlay*, podendo também (opcionalmente) utilizar o *overlay* para estabelecer ligações entre os *peers* que se encontram protegidos

por firewalls ou NATs, tirando partido dos mecanismos que o RELOAD implementa para ultrapassar esses problemas.

Em [5] os autores apresentam os Bare PC num ambiente ponto-a-ponto. Os Bare PC são dispositivos onde não existe qualquer Sistema Operativo e as aplicações operam diretamente sobre o hardware. Estes são de particular interesse para se obter uma comunicação ponto-a-ponto segura, fiável e eficiente. Neste artigo os autores começam por apresentar a arquitetura dos Bare PC e as vantagens em utilizá-los em comunicações p2p. Em seguida apresentam uma aplicação VoIP e os resultados obtidos nas várias experiências realizadas, incluindo perda de pacotes, atraso, jitter, e MOS (Mean Opinion Score). Nestes Bare PCs, o som é captado através de um microfone e é digitalizado usando ADC PCM de 16 bits e armazenado num buffer. Depois é amostrado t ms de voz do buffer que resulta em 8t bytes de dados depois da compressão usando o codec G.711. Seguidamente são adicionados os cabeçalhos RTP, UDP e IP à amostra retirada. Os autores referem que não implementaram mecanismos de supressão de silêncio nem "disfarce" de perda de pacotes que resultaria num aumento do desempenho. Por outro lado e uma vez que os seus interesses se resumiam à comunicação *peer-to-peer*, não usaram o SIP para gestão da conexão. Para testarem o desempenho obtido pela aplicação VoIP, os autores realizaram várias experiências no seu laboratório usando uma rede de teste simples e isolada.

Em [6] os autores apresentam uma rede *peer-to-peer* que permite o estabelecimento de sessões multimédia entre dispositivos móveis usando o SIP. Nesta arquitetura não existe qualquer tipo de servidores centrais e o ambiente de teste consiste no estabelecimento de chamadas VoIP entre dispositivos através da rede. Os autores começam por apresentar a arquitetura do sistema, os conceitos gerais do SIP e como é feita a descoberta de recursos no *overlay*. Seguidamente apresentam alguns testes efetuados. Na arquitetura proposta existem dois tipos de entidades: os *peers* tal e qual como noutra rede p2p comum, com as mesmas funções, e os clientes que se conectam aos *peers* para obter ou adicionar informação à rede. O SIP UA pode ser visto como um *peer* ou um cliente e o software que os autores utilizaram deve estar instalado nestes dispositivos. Como em outras implementações p2p, os dados são armazenados em DHTs e também são usados mecanismos de NAT-Transversal. Para efetuar os testes os autores utilizaram dois Nokia E61 e a openDHT do PlanetLab.

III. Aplicação VoIP na rede P2P

Neste trabalho o objectivo principal é o desenvolvimento e avaliação de uma aplicação VoIP que utilize uma rede P2P baseada totalmente em SIP. Nesta secção começa-se por descrever a rede P2P e depois as principais decisões tomadas e respetiva justificação para o desenvolvimento da aplicação VoIP, que se dá pelo nome de P2PSIPVoIP.

A. Rede overlay P2P

Para formar o *overlay* base, cada nó usa um protocolo especifico para se juntar aos outros nós. O protocolo P2PSIP usado na rede é o dSIP [7] devido a ser um protocolo totalmente baseado no SIP e o que mais se identifica com o trabalho. O facto do dSIP ser totalmente baseado em SIP é vantajoso pois o SIP é um protocolo simples, normalizado e com um bom desempenho. Além disso, o facto de ser um protocolo suportado por um grande número de dispositivos, permite a reutilização de stacks SIP implementadas, minimizando assim o número de protocolos que um *peer* precisa de suportar. Um outro aspeto positivo do SIP é que, sendo tráfego conhecido, pode atravessar as firewalls sem ser bloqueado, tal como acontece com o tráfego HTTP.

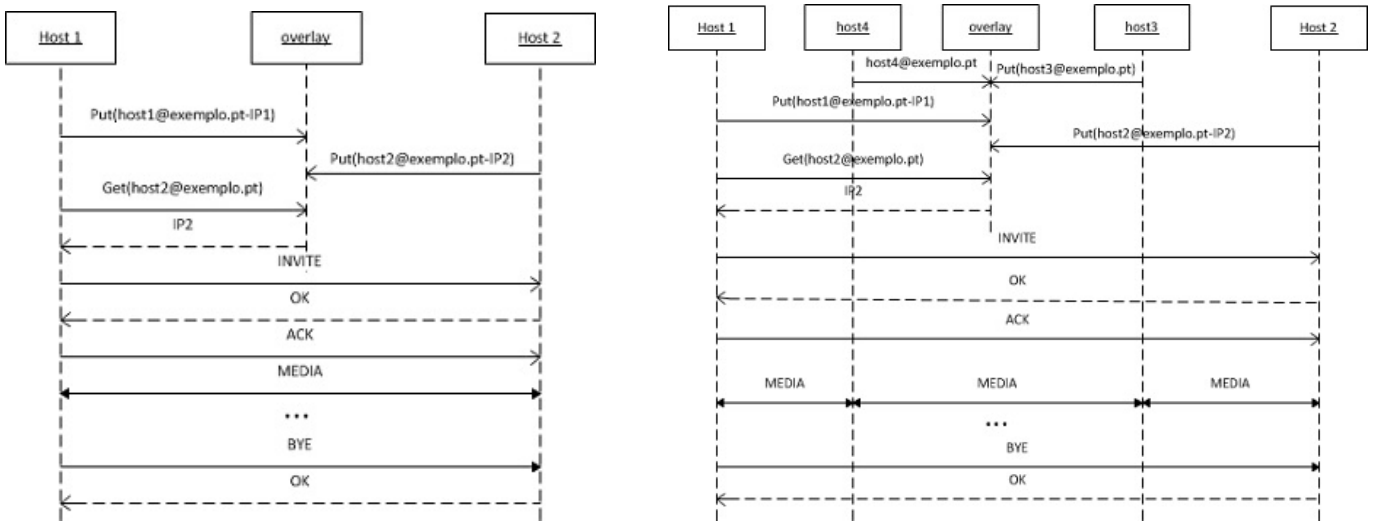
Uma vez formado o *overlay*, este é utilizado para armazenar e localizar recursos, oferecendo de uma forma distribuída, os serviços que habitualmente um Servidor de Registo e Localização oferece numa arquitetura SIP tradicional. Os recursos armazenados pelo *overlay* são compostos por um par chave/valor, onde a chave é o identificador do recurso. No contexto deste trabalho, a chave de um recurso é o endereço SIP do utilizador, e o valor é o endereço IP e porta de contacto.

O protocolo dSIP impõe restrições aos algoritmos DHT e define que pelo menos o algoritmo Chord [8] deve ser implementado. Contudo, na rede desenvolvida, para além da implementação obrigatória do Chord também está implementado o EpiChord [9], que é uma variante do Chord que pretende melhorar o desempenho geral do *overlay*. Na rede desenvolvida, ambos os algoritmos DHT utilizam o SHA-1 de 160 bits, como algoritmo de *hashing*, para gerar o *hash* dos identificadores dos *peers* e dos recursos.

Nesta arquitetura foi considerada e implementada uma hierarquia em dois níveis, isto é, com dois tipos de *peers*. Num *overlay* P2P tradicional não hierarquizado, todos os *peers* possuem a mesma importância, participando ativamente nas tarefas de encaminhamento, armazenamento e localização de recursos disponibilizadas pelo *overlay*. Contudo, há casos em que atribuir a todos os *peers* a mesma importância pode não ser o mais desejável, quer por questões de segurança, quer pelo desempenho do *overlay*. Neste caso foi criada uma hierarquia com dois níveis. O nível superior da hierarquia é constituído pelos dispositivos com mais capacidades, designados por *peers*. São estes que formam efetivamente o *overlay*, atuando como *peers* normais que armazenam recursos e encaminham mensagens entre si. O nível inferior é constituído por *peers* com menos recursos, designados por *clientes*. Os *clientes* não participam diretamente no *overlay* e necessitam do suporte de um dos *peers* do nível superior para todos os pedidos de localização e armazenamento de recursos. Nesse sentido, os *peers* foram modificados de modo a poderem aceitar e processar mensagens vindas de *clientes* externos ao *overlay*.

B. Aplicação VoIP

Antes da implementação da aplicação VoIP foi necessário adicionar funcionalidades ao *overlay* para dar suporte à sinalização da chamada entre *peers*. Para isso foram adicionados três novos tipos de mensagens: (1) *Peer Invite* - envio de um INVITE para o *peer* a contactar; (2) *Peer Ack* - envio do ACK após receber confirmação positiva do *peer* a contactar;



(a) Sem reencaminhamento

(b) Reencaminhamento com n *peers* intermédios

Figura 1. Encaminhamento das chamadas VoIP

e (3) *Peer Bye* - mensagem de sinalização de fim de chamada originada por qualquer um dos *peers* da chamada.

Após adicionar estas funcionalidades foram estudadas três alternativas na implementação do P2PSIPVoIP. Numa primeira implementação, o *overlay* é utilizado apenas para a descoberta de recursos, isto é, apenas para saber qual o IP e a porta no qual o agente SIP destinatário se encontra. A sinalização da chamada e transmissão dos dados são efetuados diretamente pela rede física (fora do *overlay*). A vantagem desta solução é o facto dos *peers* não precisarem de nenhum algoritmo de encaminhamento da chamada de voz, que reencaminhe os dados para outros *peers*, o que torna a aplicação mais leve. As desvantagens são a dificuldade em atravessar NATs e firewalls e o fraco desempenho quando há um elevado tráfego na ligação direta, não tirando assim partido do *overlay*.

Na figura 1a está ilustrado um exemplo desta solução, onde o *Host1* comunica com o *Host2*. Em primeiro lugar *Host1* e *Host2* registam-se no *overlay* enviando uma mensagem Put devidamente parametrizada com o respetivo URI e o par $(IP, Porta)$ no qual o agente local SIP está ativo. Posteriormente, o *Host1* envia uma mensagem Get, com o URI do *Host2*, de modo a obter o par $(IP, Porta)$ do agente SIP destinatário. Após obter a resposta, envia um INVITE para o *Host2*, que este aceita com OK. A transmissão de dados é feita, neste caso, diretamente entre eles. No exemplo apresentado é o *Host1* que origina a mensagem de BYE que indica o fim de chamada, mas pode ser qualquer um dos *peers* a fazê-lo.

Na segunda implementação, o *overlay* não é apenas utilizado para descoberta de recursos mas também para reencaminhar a *media* por um *peer* intermédio. Nesta solução, o *peer* emissor deve ter capacidade de detetar se a melhor rota para o recetor está ou não congestionada, decidindo em conformidade se deve continuar a enviar os dados por essa rota ou tentar reencaminhar para um *peer* intermédio do *overlay*. O *peer* intermédio escolhido vai apenas encaminhar os dados de uma forma simples para o destinatário. A vantagem desta solução advém do facto de se poder encontrar um caminho alternativo para a transmissão dos dados nos casos em que ligação direta está congestionada. Permite ainda aproveitar o *overlay* para atravessar NATs e firewalls. Em contra ponto, a aplicação tem como processamento adicional descobrir um *peer* intermédio e tomar a decisão se deve encaminhar os dados diretamente para o recetor ou através de um *peer* intermédio.

A terceira implementação é muito semelhante à segunda, com a diferença que nesta, o número máximo de *peers* intermédios a ser utilizado é definido pelo utilizador. Esta solução não restringe o algoritmo a um só salto, aumentando a probabilidade dos dados chegarem ao recetor quando existe um grande congestionamento na ligação direta, bem como nas ligações vizinhas. A desvantagem desta solução é o facto de obrigar os *peers* intermédios a manterem em *cache* uma tabela de encaminhamento tal como vai ser explicado mais à frente. Além disso, a procura de *peers* para reencaminhar os dados vai gerar mais tráfego no *overlay*.

Na figura 1b está ilustrado um diagrama de sequência que exemplifica esta terceira implementação. Tal como no exemplo anterior, o registo dos *peers* faz-se com o envio de uma mensagem Put. Após o registo, o *Host1* envia um Get com o URI do *Host2* a fim de descobrir qual o par $(IP, Porta)$ do destinatário. Nesta solução a sinalização da chamada também é feita pelo caminho direto entre ambos. Após o *Host2* aceitar a chamada e admitindo que a ligação direta está congestionada, os dados são encaminhados por outros *peers*, neste caso o *Host3* e o *Host4*. Estes *peers* intermédios aparecem na figura como sendo os mesmos encaminhadores em ambas as direções mas isso não é obrigatório. Podem ser estes dois numa das direções e quaisquer outros na direção oposta.

C. Algoritmo de encaminhamento das chamadas VoIP

O reencaminhamento das chamadas através da rede *peer-to-peer* é necessário sempre que é detetada uma má qualidade de chamada. A métrica de qualidade considerada é o atraso OWD (*One Way Delay*). A qualidade é percecionada pelo recetor, mas cabe ao emissor agir no sentido de obter um novo caminho que mantenha o OWD experimentado pelo recetor abaixo do patamar de referência definido ($OWD < 150ms$). Ao ser notificado pelo recetor da má qualidade da chamada, o emissor determina qual o melhor *peer* através do qual a possa reencaminhar. Podemos pois dividir o processo em 4 partes: (1) monitorização da qualidade da chamada e alerta de reencaminhamento; (2) escolha de um *peer* vizinho para potencial intermediário; (3) aferição do atraso experimentado na ligação através do *peer* selecionado; (4) reencaminhamento da chamada pelo *peer* selecionado.

Cada *peer* mantém duas tabelas com informação de estado: uma tabela de *peers* vizinhos, candidatos a intermediários ($Vizinhos \equiv [\![(Peer, RTT)]\!]$), e uma tabela de encaminhamento com os *peers* intermediários já escolhidos para as chamadas em curso ($Encaminhamento \equiv [\![(IPDestino, ProximoPeer, RTTTotal)]\!]$). A tabela de vizinhos deve ser atualizada de forma proativa, de modo a diminuir o atraso no reencaminhamento. Para o efeito são usadas duas mensagens: o pedido $Ping[Peer, Timestamp]$ e a resposta $Echo[Peer, Timestamp]$. Estes pedidos são enviados periodicamente a todos os *peers* conhecidos do algoritmo DHT em uso. Por cada vizinho, guarda-se o respetivo endereço e o RTT (*Round-Trip Delay Time*) calculado com base nas etiquetas temporais. Só são selecionáveis vizinhos com RTT inferior a $2 * OWD$, ou seja $RTT < 300ms$.

Quando o emissor recebe um alerta de má qualidade, seleciona um conjunto de *peers* candidatos a *relay* na sua tabela de vizinhos, por ordem decrescente de RTT e com $RTT < 2 * OWD$. De seguida envia uma mensagem de $Probe[Recetor, N, RTT\ Acumulado]$ ao primeiro da lista. Na implementação atual a lista é ordenada aleatoriamente para melhor distribuição de carga. O parâmetro N é a profundidade, ou seja, o número máximo de *peers* intermédios que pode ser considerado no caminho. O *peer* testado responderá com uma mensagem de $Response[Recetor, N, RTT\ Acumulado]$ que permitirá verificar se é um caminho viável. O processo repete-se até encontrar um caminho viável ou até esgotar a lista de candidatos. Quando um *peer* recebe um $Probe$ deve medir o RTT para o destino, caso o mesmo não conste da sua tabela de vizinhos, enviando consecutivamente um conjunto de mensagens de $Ping$. Se o RTT aferido não permitir assegurar a qualidade, e se o valor de N ainda o permitir, o *peer* poderá de igual modo testar os seus vizinhos com mensagens de $Probe$ de profundidade $N-1$. Quando o algoritmo encontra um *peer* intermédio viável, inclui uma nova entrada na tabela de encaminhamento $< IP\ Destino, Peer\ Selecionado, RTT >$.

Após o estabelecimento da chamada, os pacotes de dados começam sempre por ser enviados diretamente para o IP do destinatário. A sequência de pacotes de voz é transportada em RTP sobre UDP. Na implementação atual acrescentou-se um cabeçalho próprio com o endereço do recetor final e uma etiqueta temporal do emissor. A mensagem de voz tem o formato $Voz[UDP, Recetor, Timestamp, RTP, Payload]$ É com base nessa etiqueta temporal que o recetor calcula o OWD, subtraindo o valor ao tempo de chegada. Se o valor for superior a $150ms$, envia uma mensagem de alerta $Warning[Emissor, Recetor]$ para o emissor, para que este reencaminhe a chamada. Este cálculo exige que os relógios dos sistemas estejam devidamente sincronizados, o que é facilmente assegurado em ambiente de teste (mesma máquina). Uma alternativa a considerar para implementação futura, consiste em obter esses dados diretamente do processamento dos pacotes RTP e RTCP, o que exigirá uma maior integração com a implementação desses protocolos do que a que está a ser usada.

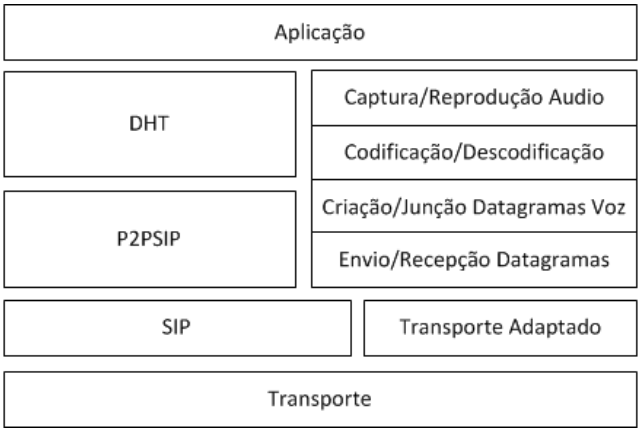


Figura 2. Arquitectura das aplicações

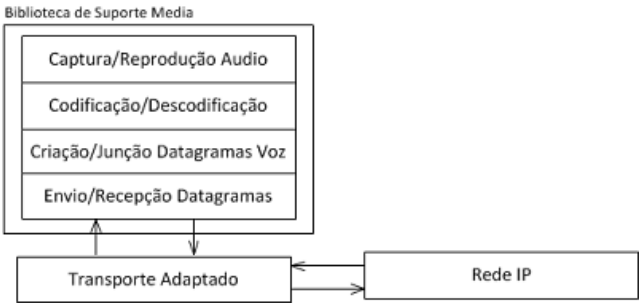


Figura 3. Transporte adaptado

IV. Implementação

A arquitetura desenvolvida foi projetada de modo a ser facilmente utilizável por várias aplicações que não apenas o VoIP. Deste modo, o sistema foi estruturado em dois grandes blocos: os módulos responsáveis pelo *overlay* e suas funcionalidades, e os módulos responsáveis pela comunicação, transmissão e reencaminhamento de dados entre *peers*. Em ambos os casos a arquitetura foi estruturada em camadas, justapostas hierarquicamente, conforme ilustrado na figura 2.

A. Camadas relativas ao overlay

A *camada de Aplicação* é a camada de interação com o utilizador. Inclui a implementação do interface gráfico que permite ao utilizador configurar os seus parâmetros, como por exemplo o *username*, o algoritmo DHT a utilizar e o *overlay* ao qual se quer juntar. Permite também efetuar e receber chamadas de voz. A *camada DHT* é responsável pela implementação do algoritmo P2P utilizado para criar e manter o *overlay*. É nesta camada que as mensagens P2PSIP são construídas e processadas, se aplicam as funções de *Hash* aos identificadores, se calculam as distâncias entre dois identificadores e se faz a gestão de recursos. Esta camada presta serviços à *camada de Aplicação*, disponibilizando funções para procurar e armazenar recursos no *overlay*. A *camada P2PSIP* é responsável por isolar as camadas superiores da implementação do SIP utilizada. É uma camada de adaptação que esconde os detalhes da biblioteca de funções SIP que está a ser usada. Deste modo, a utilização de diferentes bibliotecas SIP depende apenas de modificações nesta camada. Finalmente a *camada SIP* que disponibiliza métodos à camada P2PSIP para a conversão de mensagens P2PSIP⇒SIP e SIP⇒P2PSIP. Disponibiliza também outros métodos auxiliares necessários à manutenção do *overlay*, como por exemplo a temporização de eventos.

B. Camadas de transporte e aplicação

As camadas referentes à chamada VoIP e ao encaminhamento de dados estão ilustradas na figura 3. Na camada de *captura/reprodução áudio* faz-se a captura e reprodução do áudio, utilizando o microfone e os altifalantes do dispositivo. Na camada *codificação/descodificação*, os dados são codificados ou descodificados de acordo com o *codec* suportado pelo dispositivo e selecionado no estabelecimento da chamada, de modo a serem transmitidos através do *overlay* ou reproduzidos, conforme a situação. A camada de *criação/junção de datagramas* é responsável por abrir e terminar os canais RTP. É também nesta camada que, no caso da transmissão, se adicionam os cabeçalhos RTP aos dados codificados na camada superior. No caso de receção, é aqui que os cabeçalhos RTP são analisados e os datagramas tratados no seu devido tempo. Finalmente a camada de *envio/receção de datagramas* onde são abertos os canais UDP, para transmissão na rede dos datagramas criados pela camada superior e receção dos dados provenientes da rede IP.

Para suporte ao reencaminhamento de dados e de modo a não modificar os mecanismos internos da biblioteca de funções escolhida, foi adicionada uma nova camada denominada de *transporte adaptado*. Em vez de transmitir e receber os dados diretamente da rede, a camada de *envio/receção de datagramas* força sempre uma passagem pelo *transporte adaptado*. Esta camada é responsável pelo envio, receção e reencaminhamento da *media* na rede IP. São também aqui recebidas mensagens de outros tipos, nomeadamente $Probes$, $Pings$ ou $Warnings$, e realizadas ações de acordo com o tipo de mensagem. É também nesta camada que é calculado o valor OWD, a quando da receção da *media*, e, caso esse valor seja superior ao valor referência, se procede à criação e envio de mensagens $Warning$.

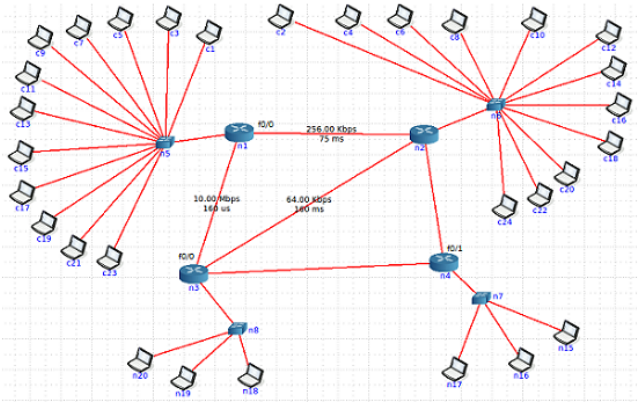


Figura 4. Cenário de teste

V. Testes e Resultados

Os testes efetuados focam-se no desempenho e validação do algoritmo de encaminhamento da *media* em situações que o volume de tráfego entre dois pontos é muito grande. Os testes ao software foram efetuados numa rede emulada, onde foram recriados cenários realistas, com testes de carga muito exigentes.

A. Ambiente de teste

Os testes foram feitos numa máquina com o CORE (*Common Open Research Emulator*) [10], que é uma ferramenta que permite emular redes, redes essas se que podem ligar a outras redes emuladas e/ou redes reais. Os testes efetuados no CORE serviram para validar a implementação e avaliar o desempenho do algoritmo com base num conjunto de métricas. Foi criada uma topologia composta por 4 *Routers*, 4 *Switches* e 30 *Hosts* conforme se ilustra na figura 4.

Nos testes de desempenho, foram obtidos os valores para as métricas consideradas em três diferentes cenários: (1)*Chamadas diretas* - foram feitas tentativas de até 12 chamadas em simultâneo entre os *peers* ligados ao router $n1$ e os *peers* ligados ao router $n2$, em duas situações específicas: $256kbps$ na ligação direta entre $n1$ e $n2$ com um atraso de propagação de $75ms$, resultando num RTT de $150ms$, e $512kbps$ com atraso de propagação de $50ms$, resultando num RTT de $100ms$; (2) *Chamada com encaminhamento da media num peer intermédio* - foram feitas até 12 chamadas em simultâneo entre os *peers* ligados ao router $n1$ e os *peers* ligados ao router $n2$, sendo a largura de banda da ligação direta entre $n1$ e $n2$ de $256kbps$ com um atraso de propagação de $75ms$ resultando num RTT de $150ms$; (3) *Chamada com encaminhamento da media em n peers intermédios* - foram feitas até 12 chamadas em simultâneo entre os *peers* ligados ao router $n1$ e os *peers* ligados ao router $n2$, em duas situações específicas: $256kbps$ na ligação direta entre $n1$ e $n2$ com um atraso de propagação de $75ms$, resultando num RTT de $150ms$, e $512kbps$ com atraso de propagação de $50ms$, resultando num RTT de

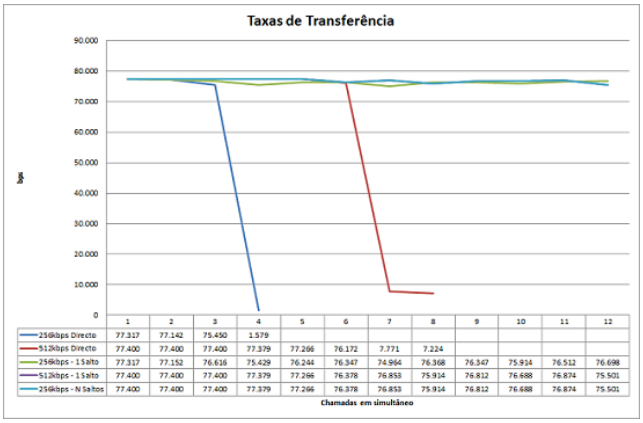


Figura 5. Variação da taxa de transferência com o número de chamadas

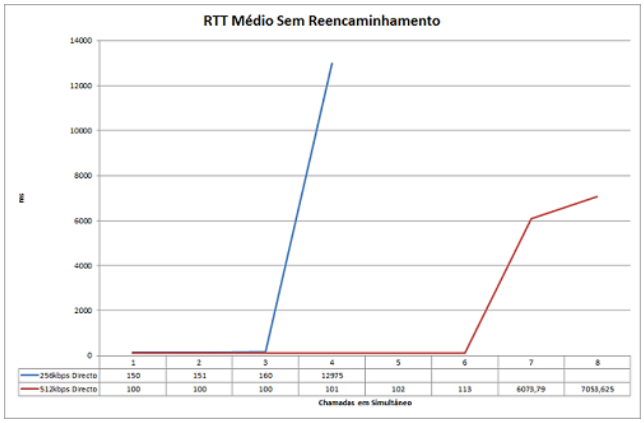


Figura 6. RTT sem reencaminhamento

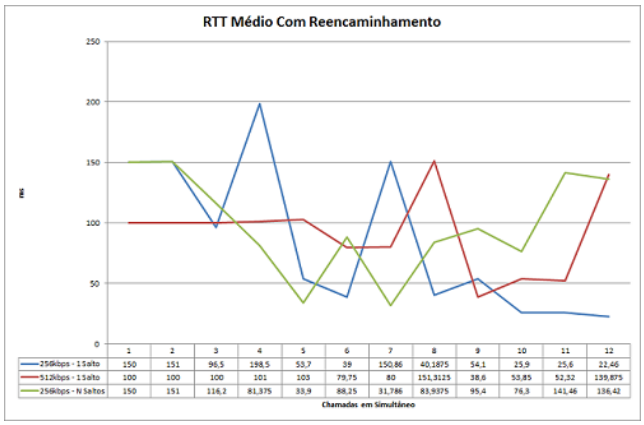


Figura 7. RTT com reencaminhamento

$100ms$. A ligação entre $n2$ e $n3$ teve uma largura de banda de $64kbps$ com um atraso de propagação de $160ms$ resultando num RTT de $320ms$.

B. Métricas

Os parâmetros analisados foram o número total de datagramas de voz recebidos, o total de datagramas com OWD menor que $150ms$, o RTT médio referente ao número de datagramas recebidos e o RTT médio referente ao número de datagramas com OWD menor que $150ms$. Todos eles foram medidos no recetor. O número de datagramas recebidos com OWD menor que $150ms$ foram usados no cálculo da taxa de transferência.

C. Testes de funcionalidade e desempenho

Após verificar o funcionamento do algoritmo de reencaminhamento foram obtidos os valores das métricas. Na figura 5 está representado o gráfico com a taxa de transferência média para cada uma das chamadas em simultâneo. Para obtenção da média do número de bits foram acumulados os números totais de datagramas de voz recebidos com OWD menor que $150ms$ (visto que são estes que interessam) em cada uma das chamadas e feita uma média entre eles. Após obter esta média, multiplicou-se por 3096 (número de bits contido por cada datagrama). O tempo de simulação em todos os testes foi de 5 minutos, portanto, 300 segundos. A taxa de transferência ideal em cada uma das direções da chamada é de $77.4kbps$.

No gráfico da figura 5 pode-se verificar que existe um número de chamadas limitado quando não existe reencaminhamento da *media* e que depende dos parâmetros definidos para a ligação. Neste caso e numa ligação com $256kbps$ com RTT de $150ms$ o número máximo de chamadas em simultâneo admissíveis é 3. Com 4 chamadas a taxa de transmissão é tão baixa que torna a comunicação sem QoS mínimo.

Na tentativa de testar 5 chamadas em simultâneo neste cenário, o congestionamento na rede não permitiu que estas se estabelecessem, isto é, as mensagens SIP perderam-se no processo de sinalização e nem todas foram estabelecidas. No caso em que a ligação tem uma largura de banda de $512kbps$ com um RTT de $100ms$ verifica-se que a ligação permite até 6 chamadas em simultâneo. Com 7 e 8 chamadas ainda se consegue estabelecer a ligação mas devido ao enorme congestionamento existem grandes perdas da *media* que se refletem na taxa de transmissão.

Em relação ao RTT médio, no caso em que não existe reencaminhamento, também há um ponto de rutura no mesmo intervalo de tempo que na taxa de transmissão. No caso em que a ligação tem uma largura de banda de $256kbps$ a partir da terceira chamada o RTT atinge um valor superior a 12 segundos. Aquando duma largura de banda de $512kbps$, com 7 chamadas atinge um valor superior a 6 segundos e com 8 chamadas um superior a 7 segundos. Estes valores estão ilustrados no gráfico da figura 6.

No caso em que existe reencaminhamento o RTT médio varia sensivelmente entre $20ms$ e $200ms$. Analisando o gráfico ilustrado na figura 7, no cenário em que a largura de banda da ligação direta entre os routers $n1$-$n2$ é de $256kbps$, o RTT diverge muito na situação de reencaminhamento com um ou *n peers* intermédios.

Esta situação deve-se ao facto do algoritmo de reencaminhamento demorar mais ou menos tempo a estabilizar e encontrar um novo caminho, isto é, enquanto a *media* é transmitida por onde existe muito tráfego, acumula-se um elevado RTT

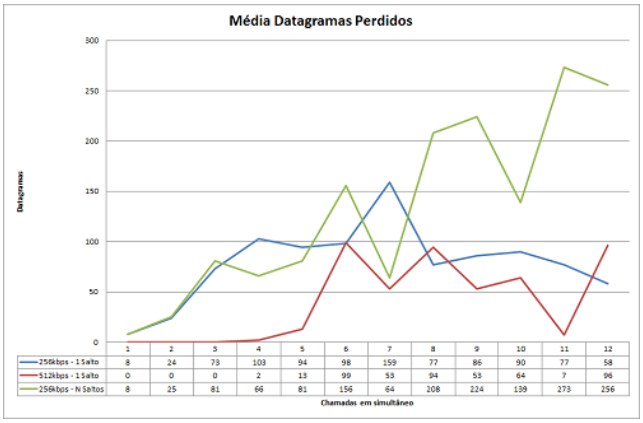


Figura 8. Variação das perdas de pacotes com chamadas em simultâneo

que se reflete na média final. Este tempo de estabilização também se reflete no número de pacotes perdidos e que estão ilustrados no gráfico da figura 8. Um outro fator que influencia o RTT é se o algoritmo desvia todas as chamadas pelas ligações onde o RTT é muito baixo ou se mantém chamadas pela ligação direta onde o RTT base neste caso é de $150ms$. Comparando com a situação em que não existe reencaminhamento, a ligação direta aguenta com um máximo de 2 chamadas em simultâneo sendo que com a terceira já há muitas perdas de dados, enquanto que com o algoritmo de reencaminhamento, a partir da terceira chamada, os dados já começam a ser reencaminhados por outros *peers* tornando assim uma comunicação mais estável. No caso em que a ligação tem uma largura de banda de $512kbps$, o RTT também depende do tempo em que o algoritmo demora a estabilizar e do número de chamadas que este consegue manter na ligação direta. Até à quinta chamada em simultâneo o RTT médio é de $100ms$ porque todas elas vão pela ligação direta. A partir daí o algoritmo de reencaminhamento é ativado e dependendo dos casos o RTT pode diminuir ou aumentar mas sempre dentro do mínimo requerido para uma comunicação VoIP.

VI. Conclusões e Trabalho Futuro

Neste artigo foi proposto um protocolo de reencaminhamento automático das chamadas pela rede *peer-to-peer*, com um número ajustável de saltos. Cada *peer* possui uma lista com potenciais *peers* que façam reencaminhamento de chamadas. Quando a qualidade da chamada fim-a-fim não é suficiente, os emissores mandam mensagens sonda (*probes*) para os seus vizinhos com o objetivo de encontrar um caminho alternativo para a *media*. O algoritmo de encaminhamento permite ainda que no caso em que os vizinhos não conseguem atingir o recetor, lancem *probes* para os seus vizinhos, aumentando assim as hipóteses de encontrar um caminho alternativo.

A implementação JAVA desenvolvida inclui a aplicação VoIP para uma rede *peer-to-peer* já existente e o algoritmo de encaminhamento proposto, a executar pelos *peers* que constituem o *overlay*. Além disso, foram adicionadas novas funcionalidades à rede *peer-to-peer* que deram suporte à sinalização da chamada VoIP e foi implementada uma interface de interação com o utilizador que permite efetuar ou receber chamadas. O sofware desenvolvido foi testado em vários cenários de modo a verificar todas as suas funcionalidades. Os resultados mostram também que com o reencaminhamento de dados, permite acomodar mais chamadas com os mesmos recursos, com a qualidade de serviço requerida.

Como trabalho futuro, sugere-se a realização de testes de desempenho a uma escala muito maior. Adicionalmente, o protocolo de encaminhamento de chamadas pode ser melhorado em vários aspetos. Como os *peers* não consideram as ligações físicas partilhadas com outros *peers*, os *peers* pertencentes à mesma rede podem trocar *probes* desnecessários entre sí. Além disso um *peer* pode receber vários *probes* de origens distintas para o mesmo destino. Todos estes *probes* duplicados podem ser evitados. Seria também interessante implementar o protocolo proposto pelo IETF, denominado por RELOAD, para criação e manutenção do *overay* e comparar as suas vantagens e desvantagens face ao SIP. Na rede *peer-to-peer* seria interessante implementar um mecanismo que dinamicamente promovesse clientes a *peers* e que despromovesse *peers* a clientes. Outro aspeto a considerar é a segurança, tanto na rede *peer-to-peer* como na aplicação VoIP.

Acknowledgment

This work has been supported by FCT – Fundação para Ciência e Tecnologia, in the scope of the project: UID/CEC/00319/2013.

Referências

[1] T. Zourzouvillys and E. Rescorla, "An introduction to standards-based voip: Sip, rtp, and friends," *IEEE Internet Computing*, no. 2, pp. 69–73, 2010.

[2] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, 2001, pp. 101–102.

[3] IETF-WG. Peer-to-peer session initiation protocol working group. [Online]. Available: https://tools.ietf.org/wg/p2psip/

[4] C. Jennings, B. Lowekamp, E. Rescorla, and S. Baset, "H. schulzrinne,"resource location and discovery (reload) base protocol," RFC 6940, January, Tech. Rep., 2014.

[5] G. H. Khaksari, A. L. Wijesinha, R. K. Karne, L. He, and S. Girumala, "A peer-to-peer bare pc voip application," in *Proceedings of the IEEE Consumer and Communications and Networking Conference (CCNC)*, 2007, pp. 803–807.

[6] M. Matuszewski and E. Kokkonen, "Mobile p2psip-peer-to-peer sip communication in mobile communities," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. IEEE, 2008, pp. 1159–1165.

[7] D. Bryan, B. Lowekamp, and C. Jennings, "dsip: A p2p approach to sip registration and resource location," *draft-bryan-p2psip-dsip-00 (work in progress)*, 2007.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[9] B. Leong, B. Liskov, and E. D. Demaine, "Epichord: parallelizing the chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006.

[10] J. Ahrenholz, "Comparison of core network emulation platforms," in *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*, 2010.

Encaminhamento Míope

Nelson Campos e José Legatheaux Martins

NOVA LINCS and Department of Informatics, Faculty of Sciences and Technology,
NOVA Lisbon University, 2829–516 Caparica, Portugal
Email: nac19324@campus.fct.unl.pt, jose.legatheaux@fct.unl.pt

***Resumo*—A emergência do paradigma SDN (Software Defined Networking) permite conceber novas arquitecturas para suporte de engenharia de tráfego e maximização da utilização da rede. No entanto uma aplicação *força bruta ou simplista* dos conceitos SDN não é escalável. Neste trabalho apresenta-se o conceito de encaminhamento míope ou SSR - *Short-Sighted Routing*, uma forma de gestão do encaminhamento na rede, na linha da filosofia SDN, com suporte de engenharia de tráfego e distribuição de carga dinâmicas, e avalia-se qual a resiliência do SSR, se for adulterada a visão do controlador sobre o estado real da rede. Isso permite avaliar indirectamente o efeito do relaxamento da urgência da reconfiguração dinâmica da distribuição de carga, com implicações na melhoria da escalabilidade. O estudo permite concluir que a maximização da utilização da rede é pouco afectada, na maioria das redes, por esse relaxamento.**

***Keywords*—Routing, Traffic Engineering, Network Optimization, SDN, Unpredictable traffic**

I. Introdução

A forma mais simples de encaminhar pacotes num *backbone* da rede de um operador é usar encaminhamento pelo caminho mais curto [1]. No entanto, dados os custos destas redes e a diversidade de serviços prestados, hoje em dia as mesmas são geridas usando encaminhamento multi-caminho e engenharia de tráfego. À luz da teoria da optimização, a maximização da utilização da rede pode ser formalizada como um problema de optimização de fluxos multicomodidade que pode ser resolvido por programação linear[2, 3]. Os dados do problema são o grafo da rede e uma matriz de tráfego [4], ou conjunto de solicitações (*demands*), ambos por hipótese conhecidos e fixos.

Para permitir obter a solução óptima mais eficientemente, assim como para facilitar a implementação na rede, a optimização pode ser realizada restringindo os caminhos a utilizar a um subconjunto de caminhos, computados a priori [3]. Neste caso a maximização da utilização consiste em procurar a distribuição de carga óptima pelos caminhos disponíveis. Uma solução sem divisão de tráfego por múltiplos caminhos está provado ser NP-difícil [5].

Na prática as matrizes de tráfego são difíceis de estimar e variam com o tempo, pelo que geralmente a optimização usa estimativas das matrizes de tráfego que se espera serem representativas para o próximo ciclo de optimização (cuja duração pode ser de várias horas ou dias)[2]. Esta abordagem é conhecida por *optimização offline*.

Idealmente, o processo de optimização deveria ser realizado dinamicamente, *i.e.*, *online*, mas tal exige estimar dinamicamente novas matrizes de tráfego, assim como a adaptação dinâmica, quer dos caminhos usados, quer da distribuição do tráfego pelos mesmos. Quando os caminhos a usar são fixos a priori, a adaptação é mais simples e resume-se a ajustar a distribuição de carga pelos mesmos. A obtenção atempada de matrizes de tráfego consome muitos recursos [4], e a adaptação dinâmica dos caminhos e das distribuições, apesar de inúmeras propostas [6, 7, 8, 9, 10], não reune consenso [11].

Na maioria das redes dos operadores usa-se uma solução baseada em MPLS-TE [12]. Cada ponto da matriz de tráfego (cada *demand*) está associado a um *trunk*[1] (ou vários *trunks* quando se usa discriminação de fluxos por qualidade de serviço ou redes privadas virtuais), cada *trunk* é também caracterizado pela capacidade que pode usar, e os diferentes *trunks* são afectados a caminhos na rede usando o algoritmo CBR (Constraint-Based Routing) [2]. Quando se usa a opção de *auto-bandwidth*, periodicamente a entrada de pacotes em cada *trunk* é medida para se estimar a capacidade real usada e, se necessário, os *trunks*, começando pelos de maior prioridade, são afectados a outros caminhos correndo de novo o algoritmo. Não só a solução é complexa, requerendo equipamentos caros e a executarem algoritmos distribuídos complexos, como na prática a reconfiguração dinâmica da rede leva a períodos de instabilidade, convergência lenta e inflação do tempo de trânsito [13].

Com a emergência recente do paradigma SDN (Software Defined Networking) [14, 15], é possível conceber uma outra solução de maximização dinâmica da utilização da rede. Quando um novo mico-fluxo entra na rede, o controlador SDN da mesma, tendo conhecimento da carga usada pelos fluxos existentes, escolhe um caminho para o novo micro-fluxo de forma a maximizar a utilização da rede. Esta solução só seria realista numa rede de pequena dimensão [16], pois numa rede de grande dimensão a agregação de fluxos é indispensável por necessidades de escalabilidade [17]. Numa rede de operador por exemplo, o dinamismo dos fluxos individuais geraria um inimaginável tráfego de controlo para a sua monitorização e controlo, e seriam necessárias reconfigurações frequentes do encaminhamento de fluxos de longa duração (*elephant flows*).

Apesar desses problemas de escalabilidade, têm sido ensaiadas, em produção [18], ou através de estudos descritos na literatura [19], soluções baseadas no paradigma SDN de optimização dinâmica das redes de interconexão dos centros de dados de vários gigantes da Internet. Em ambos os casos acima citados é usado encaminhamento multi-caminho sobre

[1] Um *trunk* é um conjunto de fluxos individualizados ou micro fluxos (*e.g.*, várias conexões TCP) com a mesma origem e o mesmo destino e que devem ser tratados da mesma forma do ponto de vista da qualidade de serviço, ou que pertencem à mesma rede privada virtual.

um conjunto de túneis definidos a priori, e a distribuição de carga pelos mesmos é feita dinamicamente a partir do resultado de um algoritmo de optimização executado por um controlador central. No entanto, como em ambos os casos o operador da rede é simultaneamente o responsável pelas principais aplicações que usam a rede de interconexão[2], as matrizes de trafego são estimadas a partir das *demands* das aplicações a priori conhecidas, e condicionadas às disponibilidades da rede de interconexão. Para esse efeito são usados *shapers* que limitam o débito máximo de cada aplicação e escalonamento das aplicações com maiores requisitos de débito (*backups*, replicação pró-activa de grandes volumes, ...) de forma a distribuir a carga.

Por estas razões, a solução não é aplicável à rede de um operador convencional que não conhece a priori as necessidades, nem pode condicionar livremente o tráfego dos seus clientes. Mas numa rede desse tipo, mantendo um conjunto de caminhos estabelecidos a priori, e conhecendo em cada momento a matriz de tráfego, um controlador central poderia adaptar dinamicamente a distribuição dos fluxos de forma a optimizar a utilização da rede. Claro que continuaria a ser necessário colectar a informação sobre as matrizes de tráfego, e realizar a adaptação dinâmica aos caminhos. A escalabilidade da solução continua a ser um desafio em aberto [20].

No entanto, esse desafio será mais facilmente ultrapassado se se confirmar que fazendo a estimativa das matrizes e a adaptação dinâmica de forma espaçada, *i.e.*, mais relaxada (*e.g.*, de 15 em 15 minutos), o desempenho da rede, com base em informação potencialmente errada, não se afasta significativamente do desempenho óptimo. A essa solução chamamos neste artigo **encaminhamento míope** ou **SSR** - **Short-Sighted Routing**, porque se baseia numa visão desfasada ou *turva* do estado da rede. Trata-se de uma solução mista entre optimização *on-* e *off- line*. Este artigo apresenta um estudo que revela que o SSR é realista em redes de operadores, mesmo quando a matriz de tráfego real da rede é bastante diferente da que foi usada para o cálculo da distribuição do tráfego.

Na próxima secção são apresentados a arquitectura da rede e o **SSR**. Na secção III são apresentados o modelo usado e os testes de avaliação que foram realizados. A secção IV apresenta os resultados obtidos. Segue-se a discussão do trabalho relacionado na secção V. O artigo termina com a apresentação das conclusões e do trabalho futuro.

II. A REDE E O ENCAMINHAMENTO MÍOPE

Nesta secção descrevem-se a arquitectura da rede e o funcionamento macroscópico do **SSR**.

A rede, ver a figura 1, é composta por um *controlador* e um conjunto de *switches*, que se dividem em *n edge switches* e vários *core switches*. Uma matriz de tráfego é uma matriz $n \times n$ em que cada par (x,y) representa o débito requerido pelo tráfego que entra na rede no *edge switch x* e tem por destino o *edge switch y*.

Edge switches - Recebem o tráfego vindo do exterior da rede, classificam-no em função do destino, computam estatísticas agregadas de forma a estimarem cada linha da matriz de tráfego em que figuram como origem, e executam a distribuição dos pacotes pelos caminhos definidos a priori, de acordo com as ordens do controlador. Adicionalmente, recebem o tráfego encaminhado pelos *core switches* que a eles se destinam e entregam-no ao exterior da rede.

[2]Os clientes não podem usar a rede privada de interconexão dos centros de dados.

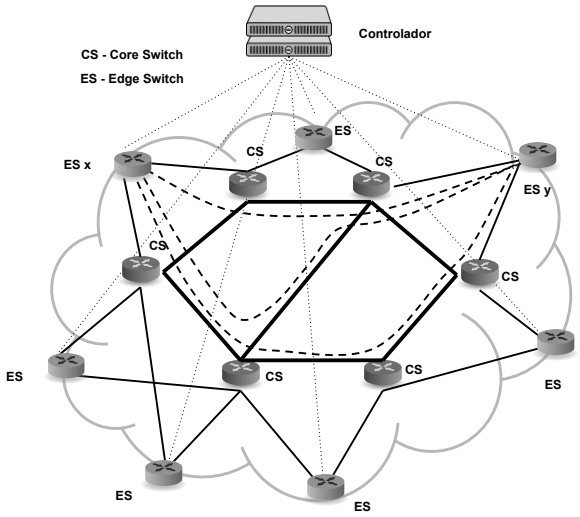


Figura 1: Arquitectura da rede incluindo o controlador, os *CS - core switches*, os *ES - edge switches* e um exemplo de 3 túneis entre os *edge switches x* e *y*

Core switches - Implementam estaticamente os diferentes caminhos de interligação dos *edge switches* através de túneis (IP over IP, MPLS, VLANs [21], ...). Os caminhos são computados a priori, de forma a maximizarem a diversidade de canais sem comprometerem o tempo de trânsito, usando algoritmos conhecidos [3, 21, 22].

Controlador - Recebe periodicamente as estatísticas agregadas colectadas pelos *edge switches*, executa o algoritmo de optimização e transmite o seu resultado aos *edge switches* na forma de uma nova distribuição do tráfego pelos caminhos existentes. É também responsável por parametrizar os caminhos fixos nos *CS - core switches*.

O SSR será adequado se o encaminhamento for satisfatório, isto é, maximiza a utilização e não penaliza fluxos, mesmo se a rede apenas realizar ajustes periódicos, e não contínuos, da distribuição do tráfego. Assim, no início de cada período é usada uma matriz de tráfego. Por hipótese, essa matriz é exacta (pois baseia-se nas estatísticas recolhidas até aí, e que são transmitidas no início do período, agregadas linha a linha, por todos os *edge switches* ao controlador). Logo, a distribuição de tráfego calculada no início do período também é óptima. Durante o período, a matriz vai evoluindo dinamicamente, mas a distribuição do tráfego mantém-se inalterada, usando as proporções calculadas no início. Assim, até ao fim do período, a adequação da distribuição vai diminuindo. Qual é a dimensão dessa degradação em função dos diversos cenários a seguir apresentados? A resposta é apresentada nas secções III e IV.

Neste estudo considera-se que é possível encaminhar os diferentes fluxos individualizados pelos diferentes caminhos de forma a implementar exactamente a distribuição do tráfego

calculada pelo controlador. Na prática isso é impossível devido à dimensão dos pacotes e dos fluxos, mas o erro cometido é desprezado. O estudo também ignora o problema da implementação da transição entre configurações de distribuição de tráfego. Pressupõe-se que tal é possível através da alteração progressiva do encaminhamento de um número adequado de fluxos individualizados.

III. Metodologia de testes

Para estudarmos a resiliência do SSR, usámos várias topologia de redes de *backbones* reais: Géant, NTT e B4. A estas juntámos 4 redes obtidas através do projecto RocketFuel [23]. Cada rede foi mapeada num grafo $G = (V,E)$, os POP (*Point of Presence*) foram mapeados nos nós do grafo (V) e os canais nos arcos (E). Adicionalmente, todos os POPs foram equiparados a *edge switches* por enviarem e receberem tráfego. Daqui para a frente os termos rede e grafo serão usados como sendo equivalentes. Na tabela I apresentam-se algumas das características das redes escolhidas.

Rede	# nós $\|V\|$	# arcos $\|E\|$	caracterização
Abovenet	15	30	*Backbone* EUA
ATT	35	68	*Backbone* EUA
B4 (Google)	12	19	Rede de centros de dados
Géant	32	49	*Backbone* europeu
NTT	27	63	*Backbone* mundial
Sprint	32	64	*Backbone* EUA
Tiscali	30	76	*Backbone* europeu

Tabela I: Caracterização das redes usadas nos testes

As matrizes de tráfego foram geradas sinteticamente usando o modelo de gravidade [3, 4]. O tráfego com origem e destino em dois POPs é proporcional às respectivas categorias (definidas em 3 classes e proporcionais à população da região). Para a geração de uma matriz inicial de referência ou *pivot*, usamos como parâmetros o grafo da rede, a categoria dos POPs e o débito total que a matriz deve receber. Como não possuímos as capacidades reais dos canais, nos nossos ensaios usámos um valor fixo para todos o canais. Como ficará claro a seguir, dada a forma como os ensaios foram realizados, esta simplificação não retira valor aos resultados. Para obtermos o número desejado de matrizes aleatórias e para simular a evolução dinâmica das matrizes de tráfego, usámos funções de distorção das matrizes de referência que procuram recriar alguns padrões de variação de tráfego em situações representativas:

Distorção aleatória - Procura recriar uma evolução aleatória de uma matriz de tráfego. Tem como parâmetro o intervalo de distorção. Quando maior este valor, mais afastada a matriz distorcida está da original. Cada um dos pontos da matriz é aleatoriamente distorcido usando uma distribuição linear das distorções. O tráfego total não é alterado.

Ataque DDoS - Esta função de distorção procura recriar a ocorrência de um ataque distribuído de negação de serviço (DDoS) na rede. Um nó é escolhido como vítima do ataque e é passado como parâmetro a intensidade do ataque. A intensidade do ataque é o factor pelo qual são multiplicados todos os elementos da coluna da matriz de tráfego respeitante à vítima do ataque.

Evento *flash-crowd* - Procura recriar a ocorrência de um evento que leva a que conteúdos acessíveis a partir de um determinado POP sejam muito requisitados (*e.g.*, *website* de notícias durante um tragédia, *stream* de um jogo de futebol, *etc*...) A função é semelhante à do ataque DDoS, só que em vez de multiplicar uma coluna por um factor, multiplica uma linha por esse factor.

Mudança de política BGP - Parte do tráfego que atravessa a rede não tem origem e destino em clientes directos da rede e tem que ser encaminhado para outros sistemas autónomos (AS). Esta função de distorção procura recriar uma situação onde uma proporção do tráfego que sai da rede pelo nó x passou a sair pelo nó y, devido a uma mudança de política do Border Gateway Protocol (BGP). São escolhidos dois nós e é passada como parâmetro a proporção do tráfego que é transferida para o outro.

Para cada uma das redes foi computado um conjunto de k caminhos simples (sem ciclos) entre cada par ordenado de nós usando o algoritmo descrito em [22]. Após vários ensaios fixámos k em 4 para permitir diversidade e um valor maior não introduzia alterações significativas nos resultados. Para determinar a distribuição de carga de uma matriz de tráfego por esses caminhos que maximiza a utilização da rede, usámos um algoritmo, que designámos por *Optimal MultiPath (OMP)*, ao invés de um *solver* genérico, visto que é possível obter com eficiência um bom resultado. O algoritmo é de seguida apresentado.

Seja $G = (V,E)$ um grafo simples (sem lacetes nem arcos paralelos), orientado, conexo e pesado, sendo o peso de cada arco estritamente positivo. $n = \#V$ é o número total de nós do grafo. Cada canal POP-a-POP *full-duplex* da rede é mapeado em dois canais *simplex* simétricos correspondendo cada um a um arco do grafo. O peso de cada arco $e \in E$ denota a capacidade desse canal e é representado por $c(e)$.

Uma matriz de tráfego D (d de *demand*) é uma matriz $n \times n$, onde cada elemento da matriz D_{xy} corresponde a um par ordenado de nós origem-destino (OD) e indica o tráfego esperado com origem no nó x e destino no nó y. Denotamos P como o conjunto de todos os caminhos disponíveis e por P_{xy} o conjunto de todos os caminhos disponíveis para o par OD (x,y). O tráfego D_{xy} é distribuído pelos caminhos P_{xy}. Para a descrição do algoritmo vamos ainda usar o conceito de *Link Utilization (LU)* que corresponde à fracção da capacidade de um canal que é usada para encaminhar tráfego. O seu máximo designa-se por *Maximum Link Utilization (MLU)* [24].

O algoritmo divide a matriz de tráfego num número (*iterations*) de matrizes idênticas, cuja soma é igual à matriz original. Cada uma dessas matrizes (d_i) representa uma fracção idêntica do tráfego global da rede (D). Em seguida, iterativamente, determina um caminho para cada entrada $d_i(x,y)$ de cada uma das matrizes d_i que minimiza a LU dos canais dos caminhos P_{xy}. No fim, o algoritmo calcula qual é a fracção de D_{xy} que deve ser distribuído por cada um dos caminhos P_{xy}, de forma a minimizar o LU dos diferentes canais e calcula o seu máximo: MLU.

O resultado obtido está dependente da ordem pela qual

são processados as diferentes entradas de cada matriz d_i, mas se o número (*iterations*) de matrizes for adequado, a fracção de tráfego processada em cada iteração é pequena, a ordem é indiferente e o resultado final tenderá para um resultado próximo do óptimo, ou seja o mais baixo MLU possível. Realizámos muitos testes e em quase todas as situações o MLU estabilizava com $iterations = 20$, mas por precaução todos os resultados finais foram calculados com $iterations = 100$. Na rede com maior número de canais, as 100 iterações levaram cerca de 1 segundo a computar usando uma implementação em Java e um processador Intel T5750 a 2.0 G Hz, mas a maioria dos testes terminou em muito menos tempo.

Segue-se uma descrição mais formal do algoritmo. O resultado final da aplicação do algoritmo é um encaminhamento definido por um conjunto de valores $f = \{f_{xy}(p) \mid x,y \in V,\ p \in P_{xy}\}$, onde $f_{xy}(p)$ especifica a fracção do tráfego de x para y que é encaminhado pelo caminho p. Seja $p_{xy}(i,j) = \{p \in P_{xy} \mid (i,j) \in p\}$ o conjunto de todos os caminhos de x para y que atravessam o arco (i,j). O total de tráfego (*load*) que atravessa o arco (i,j) é dado por:

$$l(i,j) = \sum_{x,y} \sum_{p \in p_{xy}(i,j)} D_{xy} \times f_{xy}(p) \tag{1}$$

O MLU de um encaminhamento f sobre uma matriz de tráfego D é definida pelo rácio máximo do total de tráfego que é encaminhado sobre um arco a dividir pela capacidade desse arco, para todos os arcos:

$$MLU(f,D) = \max_{(i,j) \in E} \frac{l(i,j)}{c(i,j)} \tag{2}$$

O algoritmo OMP, ver o algoritmo 1, encaminha iterativamente uma fracção de tráfego (uma entrada de cada uma das matrizes elementares) pelo caminho disponível com maior capacidade residual, *i.e.*, a menor LU. Usando fracções de tráfego pequenas, nenhum canal é sobrecarregado, e o resultado final é uma distribuição mais equilibrada. Com fracções infinitamente pequenas, a distribuição converge para uma solução que minimiza o MLU.

Numa dada rede, o resultado da execução do OMP sobre uma matriz de tráfego D é um conjunto de proporções de distribuição de tráfego que maximiza de forma aproximada a utilização da rede. Chamemos a essa distribuição o encaminhamento $f = OMP(D)$. Naturalmente, a cada f e a cada D está associado um MLU, denotado por $MLU_f(D)$. Repare-se que f e D são independentes, pois f pode também ser usado para encaminhar D' ao qual está associado $MLU_{SSR} = MLU_f(D')$.

O **SSR**, usa o OMP para calcular f a partir de uma matriz D e controla rede para que distribua o tráfego de acordo com f. Depois, enquanto não se realizar a aquisição de uma nova matriz, qualquer matriz de tráfego D' continua a ser encaminhada de acordo com a distribuição f à qual corresponderá o $MLU_f(D')$.

Para se avaliar a qualidade do **SSR**, o processo de teste é similar entre os vários ensaios. Fixa-se uma matriz de tráfego *pivot* ou de referência D, e calcula-se o encaminhamento $f = OMP(D)$. Em seguida, geram-se diversas matrizes de tráfego

Input: um grafo $G = (V,E)$, um conjunto de caminhos P, e uma matriz de tráfego D
Output: um conjunto de valores $f = \{f_{xy}(p) \mid x,y \in V,\ p \in P_{xy}\}$
begin
  Define: $p.available() = \min_{e \in p}\ (1 - \frac{e.assigned}{e.capacity})$
  Initialize: $\forall\ e \in E,\ e.assigned = 0$
  Initialize: $\forall\ e = (x,y) \in E,\ x,y \in V,\ e.capacity = c(e)$
  Initialize: $\forall\ (x,y \in V,\ p \in P_{xy}),\ f_{xy}(p) = 0$
  for i = *1* **to** *iterations* **do**
    foreach (x,y) **do**
      $p = \arg\max_{p \in P_{xy}}\ p.available()$
      foreach $e \in p$ **do**
        $e.assigned = e.assigned + \frac{D_{xy}}{iterations}$
      end
      $f_{xy}(p) = f_{xy}(p) + \frac{1}{iterations}$
    end
  end
end

Algorithm 1: *Optimal Multipath*

distorcidas D', usando uma função de distorção, e para cada uma delas calcula-se $f' = OMP(D')$, $MLU_{SSR} = MLU_f(D')$, e $MLU_{OPT} = MLU_{f'}(D')$. Tendo MLU_{SSR} e MLU_{OPT}, estamos em condições de calcular $P_{SSR} = \frac{MLU_{SSR}}{MLU_{OPT}}$ para cada uma das matrizes de tráfego distorcidas. O seu máximo indica a que distância ficou o MLU_{SSR} do MLU_{OMP}.

Como a comparação da carga dos canais é realizada usando funções lineares e o resultado dos ensaios é uma proporção, a capacidade real dos canais não é relevante desde que se considerem todos da mesma capacidade. No limite, um MLU calculado até poderia ser maior que 1 (100%).

Porque as matrizes de tráfego são geradas com factores aleatórios, efectuamos vários ensaios com diferentes matrizes de tráfego. Para os ensaios com o tipo de distorção aleatória ensaiámos com 100 matrizes de tráfego diferentes. Para a distorção do tipo ataque DDoS e evento *flash-crowd*, foram realizados ensaios em número igual ao número de nós de cada rede, escolhendo em cada ensaio um nó vítima diferente. No caso da mudança de política BGP, como é necessário escolher 2 nós, cobrir todas as possibilidades resulta num número muito elevado de ensaios. Portanto, limitámo-nos a escolher os 10 nós de maior peso (mais importantes) e cobrir todas as combinações entre esses 10 nós (90 ensaios). De referir também que para cada tipo de distorção, fizemos ensaios com 3 valores do parâmetro intensidade da distorção: 25%, 50%, 75%.

IV. Resultados dos testes e sua discussão

Foram realizados testes de convergência do algoritmo OMP e testes para todas as redes do comportamento do **SSR** e do encaminhamento pelo melhor caminho em face das distorções acima descritas. Por falta de espaço escolhemos apresentar apenas os resultados do comportamento do **SSR** perante distorção referentes apenas a duas redes do nosso *workbench*: Sprint e Géant. A primeira porque apesar de ser uma das redes mais complexas, apresenta resultados muito bons, e a segunda porque apresentou dos piores. Com excepção da rede NTT,

cujo comportamento é próximo do da rede Géant, os resultados de todas as outras redes são próximos dos da rede Sprint.

As figuras 2 e 3 apresentam os gráficos da função de distribuição acumulada da probabilidade de um ensaio ter resultado numa perda de qualidade de x (eixo das abcissas) ou menos, para as duas redes referidas. Os gráficos da função de distribuição acumulada são úteis porque permitem observar rapidamente qual foi o pior caso em cada bateria de ensaios.

Na rede Sprint verifica-se que em todos os ensaios, com qualquer tipo de distorção da matriz, o MLU_{SSR} ficou sempre a menos de 2,5% do óptimo, com excepção da distorção aleatória com o maior intervalo de distorção (75%), em que o pior MLU_{SSR} do ensaio ficou a menos de 5% do óptimo. Estes resultados mostram que o SSR é muito resiliente para a rede Sprint, um resultado comum a todas as redes menos a Géant e a NTT.

Na rede Géant, os testes com os diferentes intervalos de distorção máxima, respectivamente 25%, 50% e 75%, apresentam resultados tais que os MLU_{SSR} dos ensaios estão no máximo, respectivamente, a 10%, 20% e a 25% do óptimo. A excepção é a distorção aleatória com intervalo máximo de 75%, ver o gráfico (a) da figura 3, onde o resultado é um pouco pior. Não conseguimos explicar porque é que duas das redes apresentam resultados menos bons do que todas as outras.

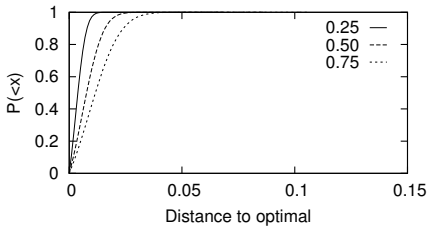
Alguns dos ensaios são muito conservadores já que em condições normais não se espera, por exemplo, uma distorção aleatória de 75% entre intervalos de ajuste da distribuição de carga nos caminhos. Em todos os ensaios o comportamento do encaminhamento pelo melhor caminho conduz a um MLU muito maior que o obtido através do **SSR**.

V. Trabalho relacionado

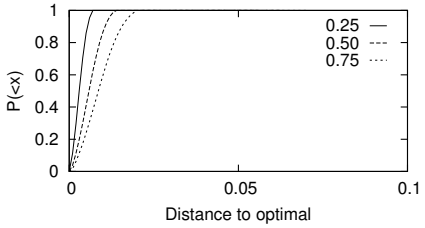
O problema da maximização dinâmica da utilização de uma rede através de engenharia de tráfego foi extensivamente estudado durante a última década [2, 6, 7, 8, 9, 10]. Todas essas propostas, assim como a solução actualmente utilizada operacionalmente [25], usam um sistema de controlo em que cada *switch* participa activamente num sistema distribuído complexo, que requer equipamentos caros, complexos e difíceis de parametrizar e fazer evoluir. Esta visão da arquitectura de controlo da rede é colocada em causa pelas propostas SDN.

Em [18, 19] uma aproximação SDN é aplicada com sucesso à maximização dinâmica da utilização em redes privadas, em que o gestor da rede é simultanemente o seu cliente, o que não é facilmente generalizável a redes WAN normais. Nas WANs normais, a aproximação SDN coloca problemas de escalabilidade de aquisição do estado da rede e transmissão de informação de controlo aos *switches*, que é uma questão muito discutida na comunidade [20]. Neste trabalho tentamos balancear a escalabilidade com a granularidade e a precisão do controlo.

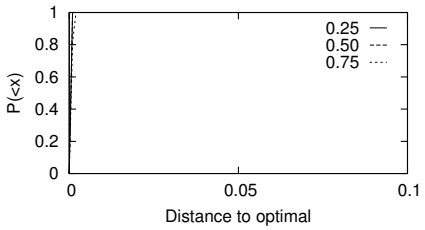
O reconhecimento da dificuldade do problema da maximização dinâmica da utilização, levou muitos investigadores a propor o designado *hose model* e o *oblivious or two phase routing* [24, 26], que são uma forma de encaminhamento definida *off-line*, usando distribuição de carga por múltiplos caminhos, de conjuntos de matrizes de tráfego que satisfazem restrições bem definidas, *i.e.*, que "cabem dentro da rede".



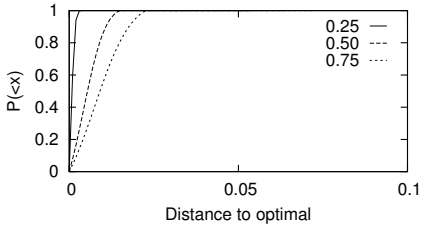
(a) Distorção aleatória



(b) Ataque DDoS



(c) Evento *flash-crowd*



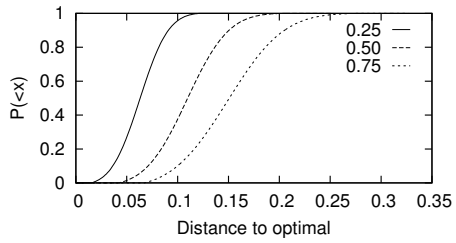
(d) Mudança de política BGP

Figura 2: Função de distribuição acumulada - Sprint

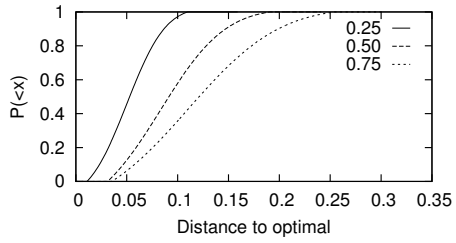
Para além de não garantirem maximização da utilização de todas as matrizes, frequentemente com resultados piores que o **SSR** quando são realizados ensaios semelhantes aos nossos [24], requerem um número muito elevado de caminhos dentro da rede pois inspiram-se do *Valiant Load Balancing* [27]. No entanto, tais aproximações podem revelar muito bons resultados em redes especiais como as dos centros de dados [28].

VI. Conclusões e trabalho futuro

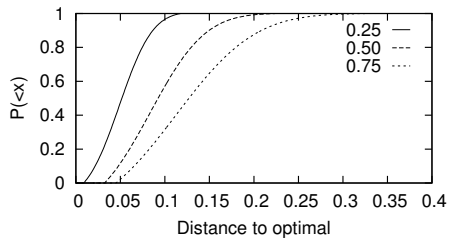
Neste trabalho apresentámos a proposta **SSR**, uma forma de controlo da rede, na linha da filosofia SDN, com suporte de engenharia de tráfego dinâmica, e avaliámos uma faceta particular desta proposta. Nomeadamente, qual a diferença da qualidade da utilização da rede face à utilização óptima, quando as matrizes de tráfego reais diferem das usadas na



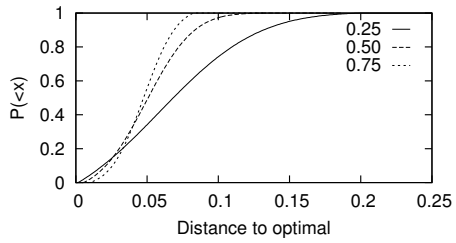
(a) Distorção aleatória



(b) Ataque DDoS



(c) Evento *flash-crowd*



(d) Mudança de política BGP

Figura 3: Função de distribuição acumulada - Géant

computação do balanceamento de carga na rede. Desta forma avaliámos indirectamente o efeito de se alargar o intervalo de tempo entre os momentos de ajuste dinâmico do tráfego o que promove a escalabilidade da solução SDN.

Surpreendentemente, o estudo permitiu concluir que o SSR tem um comportamento muito bom (e.g. resultados geralmente inferiores a 2,5% de distância do óptimo com matrizes deformadas até ao máximo de 75%) na grande maioria das redes estudadas, e um comportamento aceitável mesmo nas restantes redes (no mesmo quadro o resultado está a menos de 25% do óptimo). Desta forma, fica claro que é possível adaptar o período entre reconfigurações do **SSR** às características da rede, às folgas de capacidade disponíveis, e ao investimento em aquisição de dados e controlo da rede. Estes resultados põem em causa a necessidade do recurso a soluções teoricamente óptimas mas muito mais complexas.

No quadro da investigação da maximização da utilização de redes, excluindo trabalhos em teoria da optimização, o trabalho futuro incluí o estudo do comportamento do **SSR** perante falhas, procurar uma explicação para comportamentos tão diferentes em algumas redes, introduzir diferentes classes de serviço e modelos de tráfego mais ricos, e o aprofundamento em geral da implementação e teste da proposta num ambiente mais próximo do ambiente de produção.

Referências

[1] C. Huitema, *Routing in the Internet*. Prentice-Hall, 2000.

[2] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys and Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[3] O. M. Heckmann, *The Competitive Internet Service Provider*, 1st ed., ser. Wiley Series in Communications Networking & Distributed Systems. Chichester, UK: Wiley-Interscience, April 2006.

[4] M. R. Paul Tune, "Traffic matrices: A primer," *in H. Haddadi, O. Bonaventure (Eds.), Recent Advances in Networking, ACM SIGCOMM eBook*, 2013.

[5] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current ospf/is-is networks," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 234–247, April 2005.

[6] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive multipath routing for dynamic traffic engineering," in *IEEE GLOBECOM 2003*, vol. 6, Dec. 2003, pp. 3058–3062 vol.6.

[7] C. Villamizar, "Ospf optimized multipath (ospf-omp)," February 1999, http://tools.ietf.org/html/draft-ietf-ospf-omp-02.

[8] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "Mate: Mpls adaptive traffic engineering," in *IEEE INFOCOM 2001*, vol. 3, 2001, pp. 1300–1309 vol.3.

[9] R. Boutaba, W. Szeto, and Y. Iraqi, "Dora: Efficient routing for mpls traffic engineering," *J. Netw. Syst. Manage.*, vol. 10, no. 3, pp. 309–325, Aug. 2002.

[10] M. Kodialam and T. Lakshman, "Minimum interference routing with applications to mpls traffic engineering," in *IEEE INFOCOM 2000*, vol. 2, 2000, pp. 884–893 vol.2.

[11] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker, "Dynamic route recomputation considered harmful," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 66–71, Apr. 2010.

[12] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic engineering with mpls in the internet," *Network, IEEE*, vol. 14, no. 2, pp. 28–33, Mar 2000.

[13] A. Pathak, M. Zhang, Y. C. Hu, R. Mahajan, and D. Maltz, "Latency inflation with mpls-based traffic engineering," in *ACM IMC 2011*. New York, NY, USA: ACM, 2011.

[14] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn," *Queue*, vol. 11, no. 12, pp. 20:20–20:40, Dec. 2013.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[16] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 254–265, Aug. 2011.

[17] B. e. a. Raghavan, "Software-defined internet architecture: Decoupling architecture from infrastructure," in *11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

[18] S. e. a. Jain, "B4: Experience with a globally-deployed software defined wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013.

[19] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM 2013*. New York, NY, USA: ACM, 2013, pp. 15–26.

[20] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 136–141, February 2013.

[21] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies," in *NSDI 2010*. USENIX Association, 2010, pp. 18–18.

[22] J. Horta, M. Mamede, and J. L. Martins, "Selecção de caminhos para encaminhamento multi-caminho," in *Actas do Inforum 2013*, September 2013, pp. 78–89.

[23] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the causes of path inflation," in *SIGCOMM '2003*, ser. SIGCOMM. ACM, 2003.

[24] H. e. a. Wang, "Cope: Traffic engineering in dynamic networks," in *ACM SIGCOMM 2006*, ser. SIGCOMM '06. New York, NY, USA: ACM, 2006, pp. 99–110.

[25] E. Osborne and A. Simha, *Traffic Engineering with MPLS*. Cisco Press, 2002.

[26] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-oblivious routing in the hose model," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, jun 2011.

[27] H. Liu and R. Zhang-Shen, "On direct routing in the valiant load-balancing architecture," in *IEEE GLOBECOM 2005*, vol. 2, Dec. 2005, pp. 6 pp.–726.

[28] A. e. a. Greenberg, "Vl2: a scalable and flexible data center network," in *ACM SIGCOMM 2009*. New York, NY, USA: ACM, 2009, pp. 51–62.

A SCALABLE ARCHITECTURE FOR OPENFLOW CONTROLLERS

Filipe Azevedo
Instituto Superior Técnico
Universidade de Lisboa
Email: filipe.azevedo@tecnico.ulisboa.pt

Fernando Mira da Silva
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Email: fernando.silva@tecnico.ulisboa.pt

Luís Guerra e Silva
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Email: luis.g.silva@tecnico.ulisboa.pt

***Abstract*—The architectural principles of Software-Defined Networking (SDN) and its most prominent supporting protocol - *OpenFlow* - keep gaining momentum. SDN relies essentially on the decoupling of the *control plane* from the *data plane*, placing the former in a logically centralized component to be executed on commodity hardware - the *SDN Controller*. OpenFlow's reactive programming enables the programming of the network based on real-time decisions taken as new traffic hits the data plane, but it requires the first packet of every new flow traversing any SDN Controlled device to be sent to the Controller and evaluated, which when considered in large data center network environments becomes too large to be handled by a single SDN Controller instance. In this paper we propose a new architecture for an elastic SDN Controller cluster as a solution to overcome the aforementioned limitations by allowing for the existence of multiple SDN Controller instances acting as a single Controller but handling each a subset of the OpenFlow switches that comprise the network. A proof of concept of the said architecture has been implemented by extending the Floodlight Controller and integrating it with the Linux Virtual Server project's IP Virtual Server.**

I. Introduction

The current notion of *Software-Defined Networking* (SDN) was developed in Stanford University while researching for a network configuration protocol, from which resulted *OpenFlow*. SDN refers to the architectural principles while OpenFlow is an implementation of a supporting protocol for the said architecture. SDN relies essentially on the decoupling of the Control Plane from the Data Plane, placing the former in a logically centralized component to be executed on commodity hardware - the *SDN Controller*.

OpenFlow, being a protocol that implements SDN, allows for network programmability in a flow-oriented fashion through a well-defined Application Programming Interface (API), providing two different approaches to do so: proactive and reactive flow programming. While both approaches can be used simultaneously, there is a lot to be gained from the latter, which provides a mechanism to program the network to forward data based on real-time decisions taken as traffic hits the *data plane*, while the former provides a mean for static network programming before traffic reaches the data plane. The reactive approach, however, has a much higher computational cost when compared to the proactive approach, since for every new traffic flow traversing the network Controlled by a given SDN Controller the first packet of such flow must be sent from the OpenFlow switch receiving it to the SDN Controller, have the SDN Controller evaluate the packet, determine appropriate action, program the OpenFlow switch accordingly, and then forward the packet back into the data plane.

When applied to large scale networks, the inherent computational cost of performing the aforementioned tasks becomes far too high to be handled by a single SDN Controller instance. One way to overcome this limitation is to have several SDN Controller instances running separately, each handling a subset of the OpenFlow switch set. However this approach cannot be easily implemented since network policies would have to be explicitly configured on each and every SDN Controller, it is susceptible to SDN Controller failures, requires a cumbersome configuration of the OpenFlow switches or alternatively a mechanism of coordination between the instances of the SDN Controller in order to ensure equal load sharing across SDN Controllers. Finally, for any given flow traversing multiple OpenFlow switches Controlled by different SDN Controllers instances the aforementioned process would have to be executed on each SDN Controller instance involved.

In this paper, we propose a new architecture, intended mainly for data center environments, that offers both load balance between different SDN Controller instances and a resilient infrastructure, which enables for a scalable use of reactive flow programming regardlessly of the network size. This new architecture does so by relying on two key concepts: the clustering of SDN Controllers and the introduction of a load balancing layer. The clustering mechanism takes the notion of the SDN Controller as a logically centralized component and introduces the concept of *elastic SDN Controller clustering*, implementing the mechanisms necessary to scale out the SDN Controller as needed and in real time by providing means to add and remove SDN Controller instances from the cluster as needed with no service disruption. The load balancing layer, which is also executed on commodity hardware, is logically situated between the OpenFlow switches and the SDN Controllers, acting as a level of indirection that assigns a Controller to each OpenFlow switch in a consistent fashion that insures an equal distribution of OpenFlow switches across available SDN Controller instances.

This document covers the state of the art regarding SDN and OpenFlow in Section II, followed by the proposed architecture in Section III, its implementation in Section IV and finally the

evaluation in Section V.

II. Background

A. Software-Defined Networking

Today's network nodes are built on the architectural paradigm of three strongly-coupled planes of functionality: the *management plane*, enables service monitoring and policy definition, the control plane, which enforces the policies in network devices and the data plane which efficiently forwards data. Software-Defined Networking decouples these planes from one another, defining the abstraction of *Specification*, corresponding to the management plane functionality, *Distribution*, corresponding to the control plane functionality, and *Forwarding*, corresponding to the data plane functionality. By doing so, it makes it possible to logically centralize the Specification and Distribution functionalities while keeping the Forwarding implemented in the network nodes [1]–[3] thus creating the premises for a network infrastructure that harnesses the benefits of both distributed data forwarding and centralized management, overcoming the limitations of today's decentralized network management style.

The second core feature of SDN is the ability to programmatically configure the network, which is accomplished by having the Specification expose a set of Application Programming Interfaces (APIs), to be consumed by the network applications.

These two core concepts lead to the definition of the architecture of software-defined networks in three layers, the *Application* layer, the *Control* layer and the *Infrastructure* [1], [3]. The Infrastructure layer is where the network nodes reside, performing the Forwarding functions. The Control layer encompasses the Specification and the Distribution, therefore dealing with all the network management and control mechanisms. The Application layer is where the network applications reside. These applications define the network policies, taking into account factors such as (but not restricted to) network topology and the network operator input.

The implementation of the Control layer is known as *Software-Defined Network Controller* (SDN Controller), and it is the main building block of Software-Defined Networks. A SDN Controller exposes a northbound interface, intended for interaction with network applications, corresponding to the Specification functionality and a southbound interface intended for interaction with the network devices in the scope of the Distribution functionality. In some cases the SDN Controller might also define eastbound/westbound interfaces for integration with other SDN Controllers (mostly seen in distributed Controllers) [1].

There now are several implementations for SDN Controllers [4], as well as southbound interfaces [1], however, because there is a dominant southbound interface - OpenFlow - the vast majority of network elements supporting the SDN architecture do so by implementing only OpenFlow.

B. OpenFlow

OpenFlow is a standard protocol for forwarding plane programming, that is analogous to the Instruction Set of a Central Processing Unit (CPU) in the sense that it defines primitives that can be used by external applications to program the network node much like it is done with CPUs [2]. Forwarding decisions are based in the notion of flow, which, according to the IETF, is "a sequence of packets from a sending application to a receiving application" [5].

OpenFlow is currently on version 1.5, however, because changes made from version 1.3 to version 1.5 are not relevant to the scope of this work, and because both hardware and software switch implementations mainly implement version 1.3, we will focus on version 1.3 for the purpose of this work. The OpenFlow specification defines the architecture of an OpenFlow switch implementation in three main building blocks: the *OpenFlow Channel*, the *Processing Pipeline* and the *Group Table*. For the purposes of this work we will focus on the OpenFlow Channel, which defines the interface for communications between the OpenFlow switch and the SDN Controller, through which Controllers can create, modify and remove flow entries from flow tables in a proactive or reactive fashion.

C. OpenFlow switch to SDN Controller connection

All communications between an OpenFlow switch and an SDN Controller are, by default, encrypted using TLS, although it is also possible to use plain TCP. These communications are always initiated by the OpenFlow switch, and each switch may have one or more SDN Controller IP addresses specified in its base configuration. The OpenFlow switch is required to maintain active connections with all configured SDN Controllers simultaneously. Because multiple SDN Controllers may be configured in a single switch, each Controller is assigned one of three possible roles: *Master*, *Slave* or *Equal* [6]. Although the OpenFlow specification provides mechanisms for resilience and load balancing across multiple SDN Controller instances, it requires the SDN Controllers to load balance between themselves and also to reconfigure every single switch every time an instance is added or removed from the SDN Controller cluster.

D. OpenFlow flow programming

SDN Controllers program flow entries in OpenFlow capable switches by issuing *flow table modification* messages either in a proactive or in a reactive fashion. These messages allow the SDN Controller to manipulate flow tables by adding new flow entries and modifying or removing existing flow entries. When proactively programming flow enties, the SDN Controller issues a flow table modification message to predefine a forwarding behavior ahead of data transmission. These resulting flow entries usually have their timeouts set to zero, meaning that they will not expire, and are referred to as static flow entries [6]. Having these flow entries present in the flow tables enable the traffic to be forwarded immediately when reaching the OpenFlow switch. However, it has the

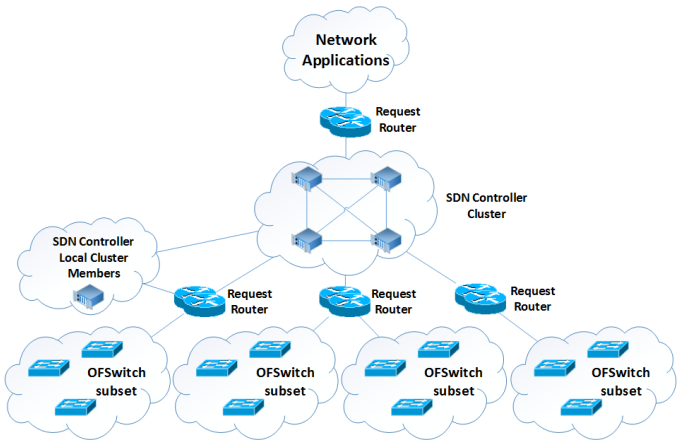


Fig. 1. Proposed architecture overview

disadvantages of having to preprogram flow entries to cover every single flow possible according to the global network policy, which leads to a quick exhaustion of the flow tables available in the switch. Furthermore, the action set being preprogrammed might not be optimal for a particular flow being processed at a particular instant. A typical usage of static flow entries are the table-miss flow entries. Table-miss flow entries, defining an instruction to forward the packet to the SDN Controller through a *packet-in* message, on the other hand enable the SDN Controller to examine the packet and define the proper actions to be executed by the switch in a reactive fashion. The SDN Controller may respond to this message with either a *packet-out* message defining the actions to be executed exclusively for that packet or alternatively with a flow table modification message that will create a new flow entry to handle all the packets matching that flow, including the original packet included in the packet-in message [6]. The resulting flow entries are referred to as *dynamic flow entries*, and it is one of the most powerful features of OpenFlow as it allows for forwarding decision-making to be performed by a centralized system that has global view and control over the network - the SDN Controller. This however comes with a considerable computational cost, since every first packet of each new flow being admitted into the network must be sent to and evaluated by the SDN Controller which in turn will instruct the switch(es) on how to behave for that flow. When considering large-scale networks, such as that of a big data center, the amount of packets that must be sent to and processed by the Controller is far too great to be handled by a single Controller instance, therefore rendering this approach unfeasible.

III. Architecture

The basic principle of the proposed architecture is to replace the SDN Controller by a cluster of SDN Controllers, which keep a consistent MIB between them. Any individual Controller in the cluster is able to manage any OpenFlow device in the same network domain. Load balance between Controllers is provided by introducing southbound and northbound request routers, which may themselves be replicated for redundancy proposes. The proposed solution enables Controllers to be dynamically added to or removed from the cluster without network disruption or OpenFlow switch reconfiguration, therefore providing an *elastic* structure as depicted in Figure 1.

A. Elastic SDN Controller cluster

OpenFlow allows for an OpenFlow switch to be connected to several SDN Controllers simultaneously, however, the latter are expected to have some sort of coordination between them in order to achieve load balancing and instance failure resiliency. Having the instances coordinated with each other and keep consistent states in a distributed environment requires therefore a SDN Controller cluster. SDN cluster instances must then provide eastbound/westbound API that allow for cluster membership operations. Because the goal is to perform load balancing between all instances, each instance will be actively managing any given number of OpenFlow switches in the same management domain, and therefore the network state is distributed in nature. In order to improve scalability and resiliency, new SDN Controller instances can be added and removed on demand from the cluster, without requiring any reconfiguration or service interruption. To that extent, when a new SDN Controller instance is initialized it must advertise itself to all existing SDN Controller instances, forming a cluster if only one other instance exists prior to the advertisement or otherwise joining the existing cluster. In order for this to be accomplished, newly instantiated SDN Controller make use of a peer instance discovery mechanism. The discovery mechanism considered is that of *group communication*.
Upon joining a cluster, regardless if new or existing, the joining instance must then synchronize its MIB with any other peer instance, from which point on it will proceed to process updates from other peers. The SDN Controller instance cluster registration process is defined in Figure 2

Because SDN Controller instances are prone to failure due to software, hardware or network failures, which prevents them from being able to notify peers, it is important to provide a mechanism that makes every cluster member aware of their peers state. As such, two mechanisms have been defined for instance failure detection: session identifiers and periodic health checks. SDN Controller instances must generate a unique session identifier when initializing themselves, which will then serve as an identifying key for cluster members, and therefore must be included in every message exchanged with peering instances. When a SDN Controller instance α receives a message from a peer instance β which is identified by session identifier υ, α must validate that the membership key κ it holds for peer β has not changed (κ=υ) before processing the message. Should the session identifier have changed ($\kappa \neq \upsilon$), α must invalidate all MIB entries associated with peer β. In order to ensure fast failure detection and quickly invalidate any MIB entry that might lead to erroneous forwarding decisions, every SDN Controller instance participating in a cluster must send periodic (every Δ seconds) Keepalive notifications to its peers. Should an instance α fail to receive three consecutive

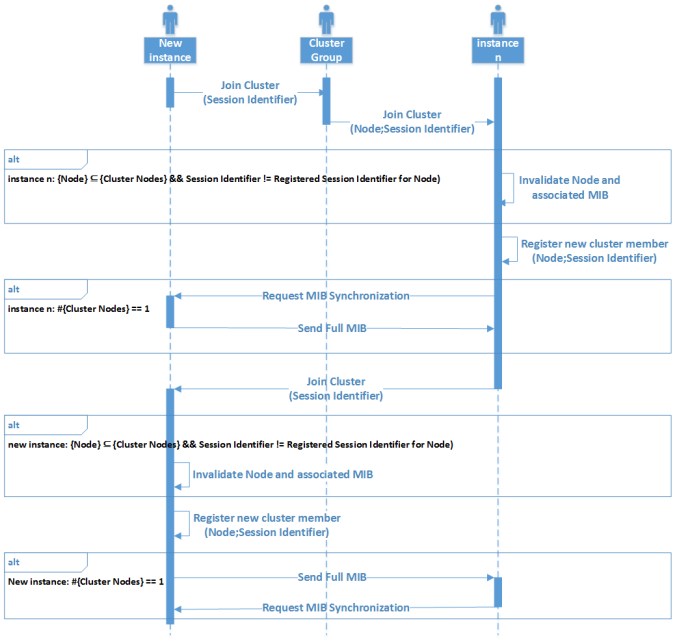


Fig. 2. SDN elastic cluster membership registration

Keepalive notifications (3Δ seconds) from a peer instance β, then α must assume that β is no longer available and therefore render all MIB entries associated with β invalid.

The propagation of MIB updates must be triggered for every network event detected by a given instance. When a SDN cluster instance propagates updates to other peer instances, it must implement *causal consistency*, thus ensuring that any dependencies between updates are met. Causal consistency ensures that messages sent by an instance α are seen by every other instance receiving them in the exact same order that they where sent by α, thus guaranteeing *causal ordering* of events throughout the cluster [7].

This architecture enables the necessary for coordination between SDN Controller instances, providing each instance with a global view of the network regardlessly of the network node(s) it manages. This approach, however, still requires that every OpenFlow switch within the management domain be reconfigured every time an SDN Controller instance is added, to or removed from, the SDN Controller cluster, which takes us back to a manual, decentralized and vendor-specific network management model.

B. Request Router

The mechanism provided by OpenFlow to load balance between clustered SDN Controller instances and inherently provide resilience requires that either OpenFlow siwtches are configured with all the IP Addresses of all the Controller instances or that the network is partitioned such that different subsets of OpenFlow switches are Controlled by different Controllers. The first approach adds a considerable amount of complexity both to the OpenFlow switch configuration and to the Controller implementation, providing however the basis for equitable load balancing and Controller resiliency. The second has the advantage of removing complexity both from the OpenFlow switches and the Controller implementation, but it is still suboptimal as it does not guarantee equal load balancing across Controller instances. In order to provide a solution that keeps the advantages and eliminates the disadvantages of both approaches, it is then necessary to implement a mechanism that transparently assigns the least-loaded instance to an OpenFlow switch - a load balancing component. This new component, the *request router*, must be completely transparent to all three components of the SDN architecture, while providing load balance between SDN Controller instances without compromising infrastructure resiliency.

The request router encapsulates all the Controller instances as a single virtual instance from the OpenFlow switches and network applications point of view, by logically sitting between the SDN Controller instances and the OpenFlow switches and having the OpenFlow switches connect to a request router instance instead of directly connecting to Controller instances. The request router is then responsible to determine which is the best SDN Controller instance to handle the OpenFlow switch and forward the request to that instance. To be able to forward connections to SDN Controller instances, it is necessary for the request router to be aware of all existing SDN Controller instances. This can be attained simply by having the request router instances joining the communication group described earlier in this paper and listening for the cluster membership messages as well as implementing the keepalive mechanism.

The request router must implement a clustering mechanism that allows the existence of several request router instances, in which active OpenFlow switch management connections are processed by any available request router instance, thus load balancing the connections between themselves such that request router instances do not become overloaded, and where any request router instance ϕ can take over the tasks performed by instance φ should φ fail.

To ensure that OpenFlow switch management connections to SDN Controller instances are load balanced between available request routers, without requiring reconfiguration of the OpenFlow switches, request router instances implement the anycast communication paradigm. By implementing anycast, OpenFlow switches will connect to whichever request router instance is logically closer to them, thus providing a mechanism to load balance OpenFlow switch management connections between request router instances, resulting in the topology depicted in Figure 1.

Considering the aforementioned specification of the request router, all OpenFlow switches must be configured to connect exclusively to the request router cluster, which will in turn forward the connection to an SDN Controller instance, as per the topology represented in Figure 1. The SDN Controller instance then replies directly to the OpenFlow switch, as portrayed in Figure 3. Having this behavior, as opposed to having the SDN Controller instance reply back to the request router instance, improves both performance and scalability as the latency for the response packets will be lower and it removes

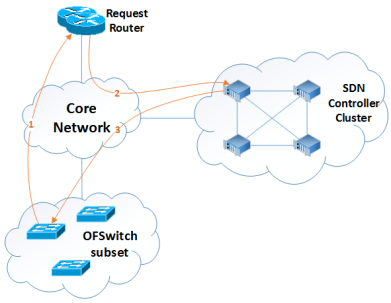


Fig. 3. OpenFlow switch management connections load balanced by Request Routers

computational load from the request router instances, thus allowing them to handle more request packets. The request router must implement a deterministic destination selection for all packets coming from the same OpenFlow switch, such that any request router instance will forward the packets from switch ρ to the exact same SDN Controller instance α. Should α become unavailable then request router instances are required to stop forwarding packets to α and assign a new SDN Controller instance β and forward all packets from ρ to β.

IV. Implementation

An experimental implementation of the proposed architecture depicted in figure 1 has been carried out by extending version 1.1 of the *Floodlight* SDN Controller, through the development of a module that implements the clustering functionalities. The Request Router functionality has been implemented on top of Linux Virtual Server (LVS), more specifically on top of the IP Virtual Server (IPVS) component, along with a managing component that integrates with the developed Floodlight module.

A. SDN Controller elastic clustering

The message propagation between instances in the same cluster has been implemented using IPv4 multicast or IPv4 unicast with TCP for specific use-cases, such as MIB synchronization which require guaranteed delivery. The Link Layer Discovery Protocol (LLDP) implementation distributed with Floodlight has also been extended in order to detect adjacencies between OpenFlow switches controlled by different controller instances. The developed module holds its own data structures, storing global state information such as a global network graph that includes OpenFlow switches, network links annotated with properties such as link capacity, network hosts and controller affinity for each network switch. MIB information exchange is performed by serializing Java objects and sending them through the network either to the multicast group or to a specific instance, depending on the use-case being considered.

B. Load balancing SDN Controllers

The request router component has been implemented on Linux virtual machines running Linux kernel 3.2 compiled with the IPVS module, which provides an efficient OSI layer 4 switching facility. IPVS is used in IP tunneling mode, which encapsulates the packets received from the OpenFlow switches in IP-IP and forwards it to the appropriate SDN Controller instance, therefore becoming completely transparent for both SDN controller instances and OpenFlow switches. The configuration of IPVS is performed automatically by a developed component that also joins the controller cluster multicast group in order to monitor cluster membership events and update IPVS accordingly. The IP anycast functionality is obtained by adding a loopback interface in each request router instance, configured with the request router's cluster virtual IP, and by running Border Gateway Protocol (BGP) to peer with the core network and advertise a path to the request router's cluster virtual IP through the request router instance. The BGP protocol is executed on version 0.99.23 of the Quagga routing software suite.

Floodlight also comes bundled with a set of network applications that provide network policy functionality out of the box. One of the bundled applications is the Forwarding module, which implements a forwarding policy that permits all flows to go through the network (also known as permit from any to any), installing the necessary flow entries in a reactive fashion [8]. The nature of this module makes it a perfect test subject for the proposed architecture, however, in order to better demonstrate the full potential of the proposed architecture this module was also extended to take advantage of the inter-instance communication mechanism, so that the forwarding policy is only computed on the SDN controller instance managing the OpenFlow switch to which the device that initiates the flow is connected to.

V. Evaluation

The scope of the testing ranges from strictly functional tests, in which the correctness of architectural model and of the implementation are validated, to some performance tests to the distributed network policy module.

A. Test environment

A virtual test environment was instantiated for the evaluation of this work resorting to a shared Infrastructure as a Service provider. This environment was composed of six Virtual Machines (VMs), one for emulating the network topology, two for running the request router cluster, and three for running SDN controller instances. Because the test environment was restricted to a virtualized support, it was also necessary to provide a virtualized network testbed solution. Version 2.2.1 of Mininet was used to emulate the topology in Figure 4, which was used in the tests described in this chapter.

B. Elastic SDN controller cluster

The first batch of tests carried out were functional tests, designed to validate the correctness of the implementation and the validity of the architecture.

For the validation of the SDN controller elastic cluster, the

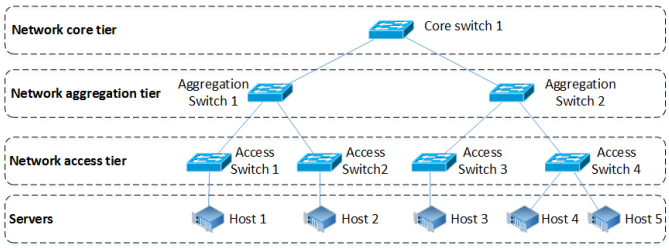


Fig. 4. Typical datacenter topology

correctness of such implementation greatly depended on the correctness of the Floodlight controller, and it is therefore considered that any behavior coherent with that of the base version of Floodlight is therefore correct. Furthermore, all architecture-specific behavior must be validated with the requirements and desired behavior previously described in Sections III and IV.

The validation of the correctness of the request router is validated with the requirements and desired behavior previously described. For the SDN controller elastic cluster specific functional tests, the scenarios tested were as follows:

1) Validate cluster membership behavior, consistency and exchanged messages. To do so the first two controller instances are to be initiated simultaneously and let them form a cluster while monitoring the network for exchanged messages. After convergence has been reached, a third controller instance is to be introduced and let it join the cluster while still monitoring the network for exchanged messages.
2) Validate MIB consistency throughout the cluster and exchanged messages by manipulating the network topology to add, remove and simulate failure of OpenFlow switches and hosts as well as simulate switch management transference between available controller instances by provoking failure of cluster instances. The management network is to be monitored for exchanged messages throughout the tests.
3) Validate the correct programming of OpenFlow switches. This functionality is to be validated by means of executing the implemented distributed network policy module to allow TCP connections between Host 1 and Host 3 as well as between Host 4 and Host 5.

The results obtained from the execution of the functional tests on the experimental implementation show that the proposed architecture provides the desired properties and moreover that the implementation's behavior is coherent with that of the Floodlight.

For the request router specific functional tests, the key points tested were as follows:

1) Validate request router clustering and state replication by having OpenFlow switches establish management connections to the IP address of the SDN controller cluster and validating that both request router instances have the same connection state information.
2) Validate that the request router properly integrates with the SDN cluster as well as with IPVS by provoking SDN controller cluster membership changes.

The request router also complied with the specifications and expected implementation behavior.

The results obtained from the tests to the SDN controller elastic cluster showed that both cluster membership and MIB are kept consistent throughout cluster members after adding, removing and provoking deliberate failures on controller instances as well as OpenFlow switches.

VI. Conclusion

We proposed a novel architecture for OpenFlow controllers based on the concept of SDN controller elastic clustering. The proposed architecture provides a scalable solution for reactive programming in large networks and, at the same time, increased redundancy and resilience in case of failure of a single controller. The prototype implementation of the proposed architecture showed that it is able to provide a full functional controller for SDN applications. Testing to distributed network policy module showed that increasing the number of clustered controller instances and consequently partitioning the OpenFlow switches among them does not affect overall performance, further proving the desired scalability properties of the proposed architecture.

Although the implementation described in this work served its purpose as a prototype, a more carefully developed solution that goes deeper into the controller core implementation will provide a considerable increase in performance.

References

[1] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," ser. Proceedings of the IEEE, vol. 103, no. 1. IEEE, 2015.

[2] Open Networking Foundation (ONF), "Software-Defined Networking: The New Norm for Networks," White paper, Open Networking Foundation (ONF), White paper, April 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[3] Open Networking Foundation, "Software-Defined Networking (SDN) Definition." [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition

[4] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (sdn) controllers," *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, 2014.

[5] Internet Engineering Task Force (IETF), "What is a Flow?" http://www.ietf.org/proceedings/39/slides/int/ip1394-background/tsld004.htm, accessed: 2014-11-27.

[6] Open Networking Foundation (ONF), "OpenFlow Switch Specification v1.3.4," OpenFlow Spec, Open Networking Foundation (ONF), OpenFlow Spec, March 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.4.pdf

[7] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley Publishing Company, 2011.

[8] Big Switch Networks, Inc., "Floodlight Forwarding Module," https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Forwarding, accessed: 2015-09-12.

Experimental Evaluation of SDN-based End-to-End Mobility Management in Wireless Environments

Flávio Meneses[1], Daniel Corujo[2], Carlos Guimarães[3], Rui L. Aguiar[4]
Instituto de Telecomunicações, Universidade de Aveiro, Portugal
Email: {*flaviomeneses[1], dcorujo[2], cguimaraes[3]*}*@av.it.pt; ruilaa@ua.pt[4]*

***Abstract*—Software Defined Networking (SDN) is seen as one of the underlying mechanisms of the next generation telecommunications architecture (i.e., 5G networks). Despite SDN being deployed at the core and back-haul of the mobile network, its usage directly over the wireless mobile terminals has been disregarded, as well as the possibilities for novel control procedures in wireless nodes. Using the OpenFlow protocol, an instantiation of SDN, this study extends the SDN mechanisms all the way to the mobile node in heterogeneous wireless environments, enabling the mobile node to assist the controller entity in the flow-based mobility procedure. The concept framework was implemented over a physical wireless testbed, validating its contribution in a mobile source-mobility use case, with results highlighting the promising benefits of extending SDN approaches for end-to-end flow control in wireless environments.**

I. Introduction

With the increase in the number of users and provided services, the Internet has been pushed to its limit, motivating pioneer architectural approaches, such as Software Defined Networking (SDN). SDN promises to empower network infrastructures to better support upcoming challenges, such as massive traffic volumes, the proliferation of connected mobile devices and sustainable integration of heterogeneous networks in mobile environments [1]. This challenges have been addressed by upcoming 5G research, with SDN as one of its key building blocks.

One of the principles of the SDN architecture is the decoupling of the control and data planes, with the OpenFlow protocol, which represents the de-facto open-source SDN instantiation [2], enabling the interaction between both planes. The Open Network Foundation (ONF) [3] also points that scalability is improved by decoupling and centralizing control, with a logically centralized controller having a broader perspective of the resources under its control.

In this work, we explore the deployment of SDN mechanisms all the way to the mobile node in heterogeneous wireless environments, enabling the integration of SDN mechanisms into the mobile node (MN). This allows it to receive commands from a network controller entity, promising benefits such as a mobile offloading and source-mobility, both achieved independently of the wireless access technology to which they are connected. Also, keeping the handover intelligence in the network potentially increases control granularity and homogeneity, since the handover can be performed via OpenFlow, without requiring additional mobility protocols.

Acknowledgment This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER PT2020 partnership agreement under the project UID/EEA/50008/2013, and by the FCT Grant SFRH/BD/96553/2013.

II. Design and Key Features

Focusing on the interaction of the MN with the network for the optimization of mobility processes, the proposed framework (presented in Fig. 1) extends SDN mechanisms all the way to the MN, enabling network decision entities (e.g., controllers) to perform flow handovers directly in the MN via OpenFlow messages, and without requiring any other mobility protocol. Moreover, the framework features different mobile nodes in heterogeneous wireless with multiple data flows, which act both as consumers and producers of information. In this way, the SDN controller is able to interact directly with the MN, allowing it to assist in potential handover decisions and policy enforcement by providing their network connectivity perspective (i.e., detected cells and their link quality).

The proposed framework, deployed using Raspberry Pi with Wi-Fi usb adapters, considers that every node has an OpenFlow agent. As such, the Open vSwitch (OvS)[1] was installed in the MN, enabling it to communicate directly with the network controller via OpenFlow messages. Considering that the MN is dual-interfaced, an OvS layer 3 bridge instantiation per interface was required, performing as well an OpenFlow control connection per interface. This requirement comes with the vision of the MN not only as a consumer, but actually also a content producer, implying the need of a module between the kernel TCP/IP stack and the physical network interface.

[1]http://openvswitch.org/

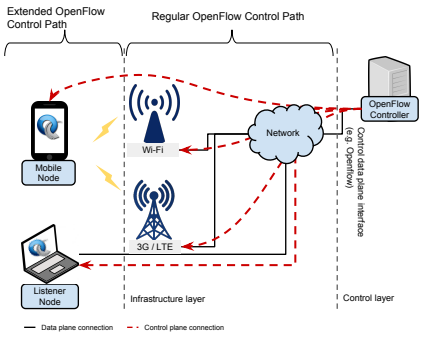


Fig. 1: Framework architecture

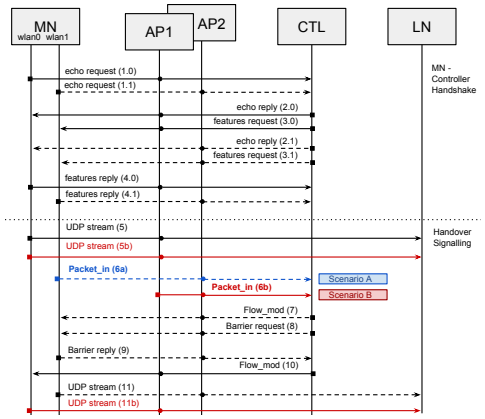


Fig. 2: Flow Diagram

To enable the traffic redirection between interfaces, a patch port was configured interconnecting both OvS bridges. While the MN is producing content, the Listener Node (LN) is the consumer of that content.

III. Evaluation

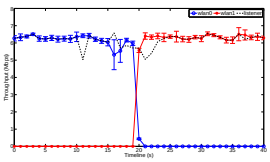
Fig. 2 shows the signaling for two distinct scenarios. **Scenario A** refers to a handover situation where the handover trigger is sent from the MN (i.e., MN initiated), shown here in red. **Scenario B** refers to a handover situation where the handover trigger is sent from the AP (i.e., Network Initiated), shown here in blue. The remaining signal messages are applicable in both scenarios[2], however while the continuous lines are for the MN communication through its *wlan0* interface, the dashed lines are regarded to the communication via *wlan1*.

The average throughput of the end-nodes (i.e., MN and LN) for both scenarios is illustrated in Fig. 3. In scenario A (i.e., Fig. 3a), as the MN moves away from AP1, the average throughput received by the LN starts to decrease, and then the MN requests for a flow handover. As soon as the MN redirects its flow to the new AP, the LN recovers its throughput.

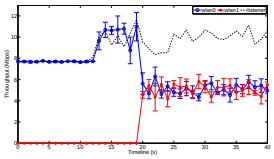
In scenario B (i.e., Fig. 3b), the handover trigger is sent from the AP, simulating a load balancing use case. When the second stream starts, the LN throughput decreases due to the overloading of the AP1. Detecting its overloaded state, AP1 advertises the controller, which in turn proceeds to the handover, redirecting the MN flow to AP2 (Fig. 3b, red line), re-establishing the receive throughput in LN.

About the OpenFlow handover signalling exchanged between the controller and the MN (shown in TABLE I), a total of 550 bytes were exchanged in the five OpenFlow messages involved. In scenario A, these five messages were handled by the MN, while in scenario B, the handover solicitation (i.e., the OpenFlow *Packet_in* message), which has a data amount of 134 bytes, is sent by the AP, improving not only the handover delay, but also contributed towards a reduction

[2]The evaluation results shown here represent an evolution of past instantiations of this concept, (i.e., [4] and [5]), and have been submitted for publication at Springer Plus.



(a) Scenario A (65dBm threshold)



(b) Scenario B (12Mbps threshold)

Fig. 3: Throughput for both scenarios

of the overhead introduced in MN by the handover signalling. However, the major amount of OpenFlow signalling data is due to the keep-alive messages exchanged every 5 seconds between the controller and OvS (by default). Since the framework aims to operate independently of the underlying access technology, we do not consider the overhead introduced by Layer 2.

TABLE I: OpenFlow signalling data

		MN initiated		Network initiated	
		Packets	Bytes	Packets	Bytes
Mobile Node	Total	36 ± 1	2422 ± 48	35 ± 1	2371 ± 41
	Handover	5 ± 0	550 ± 0	4 ± 0	416 ± 0

IV. Conclusion

The results showcased the feasibility of the proposed mechanisms towards a source mobility offloading scenario, without requiring additional mobility protocols. Despite the handovers were performed without packet loss and the minimal overhead introduced by proposed mechanisms, the handover delay pointed the need for further optimization on the deployment of OpenFlow mechanisms within MNs. However, these performance concerns were obtained in a prototype environment, where the SDN mechanisms and the OvS implementation were pushed into a behaviour that went beyond their original design.

As future work the Controller-MN interaction needs to be improved (e.g., new demands placed over both the network and the MN itself), by developing novel mechanisms that better suit the integration of SDN into the MN, but also the application evaluation of this concept applied to different kinds of mobile devices (e.g., Machine-to-Machine scenarios).

References

[1] C. W. Paper, "Cisco visual networking index: Global mobile data traffic forecast update, 2014-2019," February 2015. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] Open Networking Foundation, "SDN architecture [online]," 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[4] F. Meneses, D. Corujo, C. Guimares, and R. Aguiar, "Multiple Flow in Extended SDN Wireless Mobility," in *European Workshop on Software Defined Networks, 2015*, Aug 2015.

[5] ——, "Extending SDN to End Nodes Towards Heterogeneous Wireless Mobility," in *Globecom Workshops (GC Wkshps), 2015*, Dec 2015.

Automatic and secure WLAN configuration by a captive portal

André Fernandes
School of Engineering, Polytechnic of Porto
Porto, Portugal
andre_fernandes@eu.ipp.pt

Antonio Pinto
GCC, CIICESI, ESTGF, Polytechnic of Porto
and INESC TEC
Porto, Portugal
apinto@inesctec.pt

Jorge Mamede
INESC TEC and
School of Engineering, Polytechnic of Porto
Porto, Portugal
jmamede@inescporto.pt

***Abstract*— The popularity of Wi-Fi enabled devices has lead to an almost ubiquitous availability of public Wi-Fi hotspots in places such as airports, train stations, and even households. These hotspots allow any client to connect by means of an unsecured (or open) wireless link. Upon its first use, the hotspot requests either user authentication or payment. The need for an open wireless link is based on its easy of use, not requiring complex client configuration procedures. However, providing users with an unsecured wireless link for Internet access may hamper the user's security, confidentiality and privacy. In particular, the DNS queries required in day to day Internet usage are available in clear text and, when captured, may lead to the exposure of the user's Internet habits, for instance. The authors propose a novel mechanism that enables the automatic and transparent configuration of WLAN security (WPA, WAP2) of users successfully authenticated in a captive portal. A proof of concept was setup.**

Keywords*— *WLAN, automatic configuration, authentication, captive portal, secure connection.

I. INTRODUCTION

Public wireless Internet hotspots are common in places such as airports or shopping centers and users can freely connect to these public wireless networks. After the establishment of the wireless link, users are prompted for their credentials by means of a web page in a captive portal. Upon successful authentication, or payment, users are allowed to access the Internet. Such public hotspots adopt the use of open wireless connections as it is simpler for the user.

Figure 1 depicts the reference scenario. Three distinct entities are envisioned: the user, the captive portal operator, and the authentication operator. The user is the one that wants access to the Internet and is already in the possession of a set of valid credentials. These credentials are shared between the user and the authentication operator. The captive portal operator is the one that deploys local infrastructures and that enables user connections. The authentication operator is the one that possesses the user credential database and is assumed to have previously established an agreement with each captive portal operator.

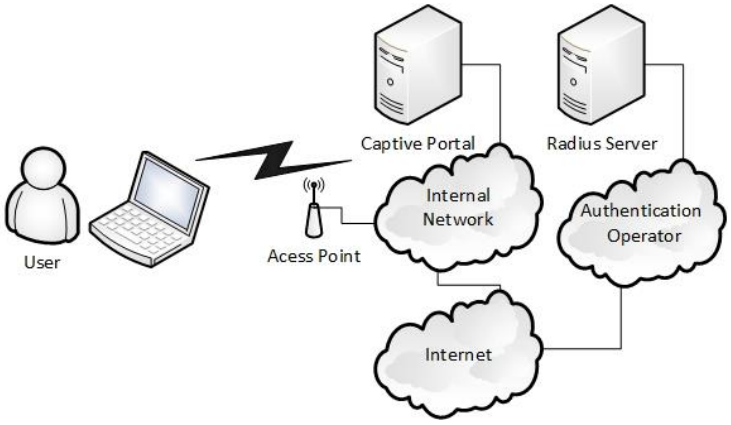


Figure 1 – Reference scenario

Despite enabling users with an easy of connection and use, the adoption of open wireless connections present some drawbacks with regard to security, privacy and confidentiality. Any user in the vicinity of the hotspot can capture all traffic exchanged by all users of the hotspot. Such may lead to: the loss of confidentiality, if application protocols are used through insecure connections (email or web browsing for instance); the loss of security, if user credentials are exchanged in clear form while browsing the web; and to the loss of privacy, if the network uses a standard DNS service where all DNS queries are exchanged in clear form.

A possible solution would be the use of more secure forms of wireless connections such as Wi-Fi Protected Access

(WPA)[1]. The problem being the setup process; it is seen as a way of hampering captive portal usage. The proposed solution addresses this issue by enabling the transparent and automatic configuration of secure wireless connections upon successful user authentication in a captive portal.

The paper is structured as follows. Section II presents the related work and relevant network protocols. Section III describes the common operation of hotspots based on captive portals. Section VI details the proposed solution and describes the architecture and implementation of the proof of concept prototype. Section V presents the obtained results and Section VI concludes this paper.

II. RELATED WORK

Scenarios analogous to the adopted scenario are common worldwide. Fon (Fon Wireless Ltd.) is an example of particular interest as it is available almost anywhere. FON consists of a community linking user-provided wireless hotspots that was founded in Madrid [2] and built its community by selling customized routers ("La Fonera") that authenticate users by connecting to a Fon server through the Internet. The router comprises an access point that announces two wireless networks: an encrypted network for the router's owner, and an open public network for other Fon users that operates a captive portal.

Fon considers passive and active members. Passive members do not share their Internet connection with the community and, if they want to use Fon spots, they have to pay for it. Active members own a Fonera router and share their Internet connection with other member in exchange for free roaming at other Fon spots [3]. Considering the Fon use case, the active members can be seen has both users and captive portal operators, while Fon will serve as the authentication operator.

A. User authentication

Users in a captive portal-based hotspot, when connecting, are confronted with a login page that, in turn, will trigger the verification of the user supplied credential. If the user is successfully authenticated, the captive portal will reconfigure itself to allow traffic from the user. The Remote Authentication Dial In User Service (RADIUS) is a widely adopted authentication, authorization and accounting protocol that commonly found in this scenarios [4].

RADIUS operations is setup with a Network Access Server (NAS) [5]. The NAS is the one that issues user credential verification requests, by sending Access-Requests messages to the RADIUS server. NAS accepts or rejects network access to the requesting user depending on the result of the credential verification done by the RADIUS server.

Upon successful credential verification, the RADIUS server sends back an Access-Accept message, if not, the server sends an Access-Reject message. Additionally, if the user is accepted, the NAS will send the Accounting-Request message to the RADIUS server. The server replies with an Accounting-Response message. After this point, users can access the Internet.

B. WiFi network access

Wireless Local Area Networks (WLAN) [6] support both unsecured and secured wireless link establishment. The unsecured is commonly referred to as an open wireless link. Users in the proximity, or farther away if highly sensitive directional antennas are used, can connect and listen to all transmitted traffic.

The wireless link can be secured, adding functionalities such as client authentication and message encryption. Wired Equivalent Privacy (WEP) [7] is a shared key authentication based on a unilateral challenge-response mechanism that also support wireless link encryption. WEP was designed to provide the equivalent security of a wired LAN by encrypting network traffic with the RC4 algorithm [8]. The IEEE 802.11i amendment was later ratified to address several security issues identified with WEP.

The WPA was introduced by Wi-Fi Alliance as the replacement of WEP. WPA development was guided towards correcting the security flaws found in the WEP protocol. Among the proposed improvements, the most significant was the use of RC4 algorithm in a more secure manner within the TKIP protocol. The WFA later launched the second version of WPA (WPA2), after the discovery of some security flaws in the TKIP. TKIP was substituted by the CCMP [9] protocol that uses the AES [10] symmetric encryption algorithm.

WPA, in any version, supports two distinct modes of operation: WPA-Personal, and WPA-Enterprise. WPA-Personal is analogous to WEP as it requires a key for link establishment. This key is a key that is shared among all users of the wireless network and is know as a Pre-Shared Key (PSK). On the other hand, WPA-Enterprise requires a set of user credentials for link establishment and is deployed in conjunction with an authentication server, such as a RADIUS server. WPA-Enterprise is common in business environments where a RADIUS server exists.

Extensible Authentication Protocol (EAP) is an authentication platform [11]. It provides a framework that enables the secure use of multiple authentication methods. EAP Tunneled Transport Layer Security (EAP-TTLS) [12] is an extension of EAP Transport Layer Security (EAP-TLS), that was created to reduce the TLS implementation complexity due to its use of digital certificates. In TTLS, the authentication server only authenticates itself to the supplicant. Clients can choose to authenticate themselves to the server. Hence it is both a one and two way authentication method. EAP-TTLS supports multiple inner client authentication protocols like Password Authentication Protocol (PAP)[13], Challenge-Handshake Authentication Protocol (CHAP)[14], Microsoft Challenge Handshake Authentication Protocol (MSCHAP) and MSCHAPv2[15]. The authentication process takes place inside a secure tunnel. [16]

III. CAPTIVE PORTAL OPERATION

Wireless hotspots based on captive portals operate in a similar way, capturing all user traffic until successful user authentication. User authentication is made through a login

web page. Figure 2 depicts a user's connection and authentication while using a captive portal.

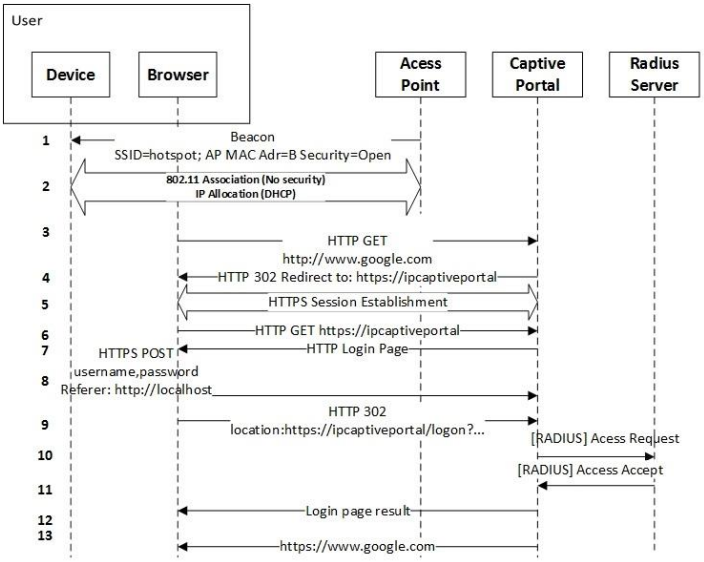


Figure 2 – Captive portal operation

The Access Point (AP) periodically broadcasts the beacon frame containing the hotspots' SSID, the AP's MAC address and the security information. The security information is set to open, or none (step 1 of Figure 2). Afterwards, the connecting device obtains IP its configuration (DHCP) and allows the user to start its web browsing. All IP communications of unknown users are blocked except for DHCP, DNS and ARP protocols.

The user, in this example, attempts to access a website (google) and is redirected to a HTTPS connections to a pre-configured initial webpage of a captive portal. The HTTPS session is created between the client and captive portal to ensure secure delivery of the user's data (step 5). The HTTP 302 message is used for this redirection. The client then sends a HTTP GET message to the URL comprised in the location field of the received HTTP 302 message. This URL contains the address of the captive portal's login page.

The user enters his credentials (username and password) on the login page. The credentials are securely delivered to the captive portal via a HTTPS POST message. When receiving the user credentials, it forwards them to RADIUS server for user authentication.

The RADIUS server, based on this information, decides whether the authentication should succeed or fail and replies back to the captive portal. Upon successful authentication, the captive portal removes the HTTP redirection rule applied to that specific client. The user is now allowed to access the Internet.

IV. PROPOSED SOLUTION

The proposed solution maintains the global operation of the captive portal while enabling the secure exchange of secure wireless network configuration files so that clients can automatically switch to them upon successful authentication.

Figure 3 depicts the proposed solution operation. In particular Figure 3 shows the difference between the standard captive operation and the proposed solution, starting at step 8. Previous steps from standard captive operation are required and assumed to have taken place.

The proposed solution introduces two new sets of applications that must be available at both the client (Client.java) and the authentication operator (Server.java). The server side application is installed once. The client side java application is made available to the clients with a link in the login page of the captive portal and must be installed once.

The client side java application is introduced so that the user does not need to introduce its user credentials more than once. The login redirect (step 8 of Figure 2) is made so that the browser makes the POST request to the client side java program. This allows the client side java program to obtain the users credentials that it will need for the configuration file decryption. The user credentials are used by the server side java program to encrypt the configuration file prior to its transmission to the client.

Afterwards, the client side java program will issue the HTTP 302 request as if was the clients browser. This will allow for the continual regular operation of the captive portal, for instance, checking user credentials and removal of HTTP redirection rules.

Concluding the process, the client side java program, now in possession of the secure wireless network configuration file, will automatically trigger the connection to the new secure wireless network. For instance, a WPA network configuration could be used.

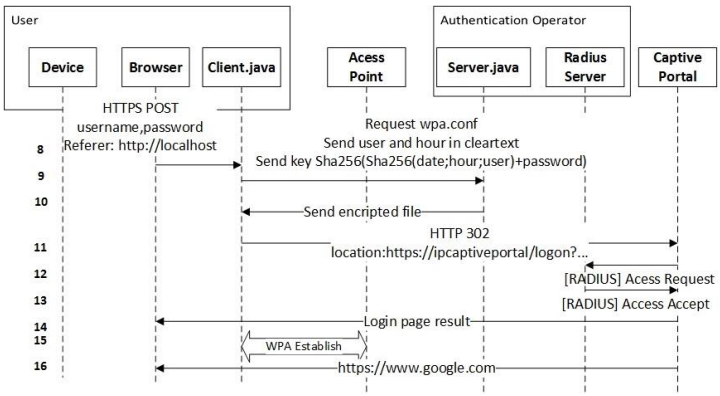


Figure 3 – Operation of the proposed solution

A. *Proof of Concept*

A proof of concept of the proposed solution was setup (see Figure 4) in a PC with 4GB of RAM memory, a 2.0 Ghz dual-core processor virtualization and running the Windows 7 operating system (64 bits). Two virtual machines were setup with CentOS 6.6 operating system (32 bits), one representing the client and another representing both the captive portal operator and the authentication operator.

The client virtual machine was setup with 845MB of virtual RAM. The other virtual machine was setup with 1GB of virtual RAM, the Coovachili captive portal (version 1.3.0), the

FreeRADIUS RADIUS server (version 2.2.0) and the hostapd package (version 2.4) to simulate the AP.

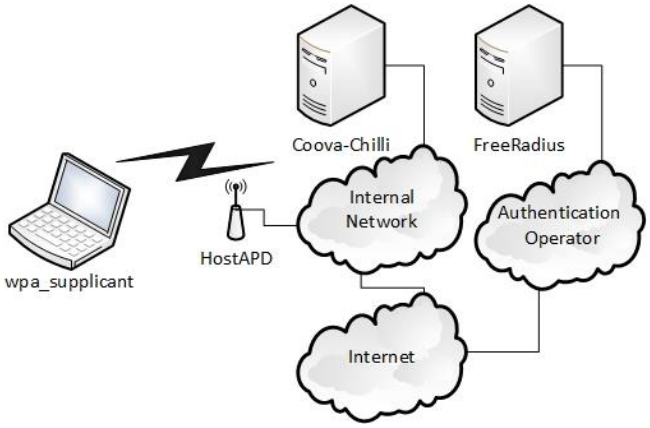


Figure 4 -Proof of concept setup

The login page of the adopted captive portal was altered (see Figure 5) so that the user would be able to select the proposed solution as an authentication mode. The normal operation is still maintained and available. The user must select "Use secure connection" to use the proposed solution.

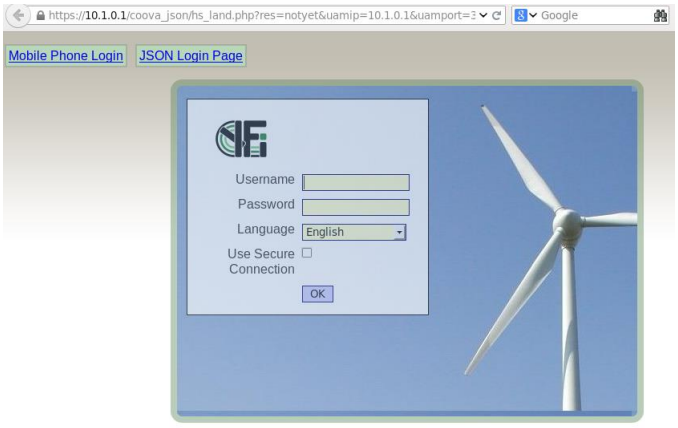


Figure 5 – Altered login page

Algorithm 1 describes the operation of the client side java program. Initially, this program obtains username and password from the redirected HTTP message. A key is then generated. The key will be used for authentication and file encryption. Afterwards, the wireless configuration file is received and decrypted. Client program then runs wpa_supplicant with the new configuration file and establishes a secure wireless link.

Algorithm 1 – Client side

1: Obtain username and password
2: Seed ← SHA256(date+time+username)
3: AuthToken ← SHA256(Seed+password)
4: Send AuthToken
5: Receive wpa.conf
6: Decrypt wpa.conf
7: Run wpa_suplicant with wpa.conf

Algorithm 2 describes the operation of the server side java program. Initially, this program waits for client connections. Upon client connection, the server will validate the connecting user, compare the two authentication tokens and, if these match, will encrypt the secure wireless network configuration file with the key generated on the server. Afterwards, sends the encrypted file to the client and waits for more users.

Algorithm 2 – Server side

1: Receive clientAuthToken
2: Obtain user's password from radius database
3: Seed ← SHA256(date+time+username)
4: AuthToken ← SHA256(Seed+password)
5: If AuthToken = ClientAuthToken
6: Encrypt wpa.conf
7: Send wpa.conf

Figure 6 shows a network capture of a successful WPA network association. The network capture starts with the traditional EAPOL start message of the EAP-TTLS and ends with the success client authentication.

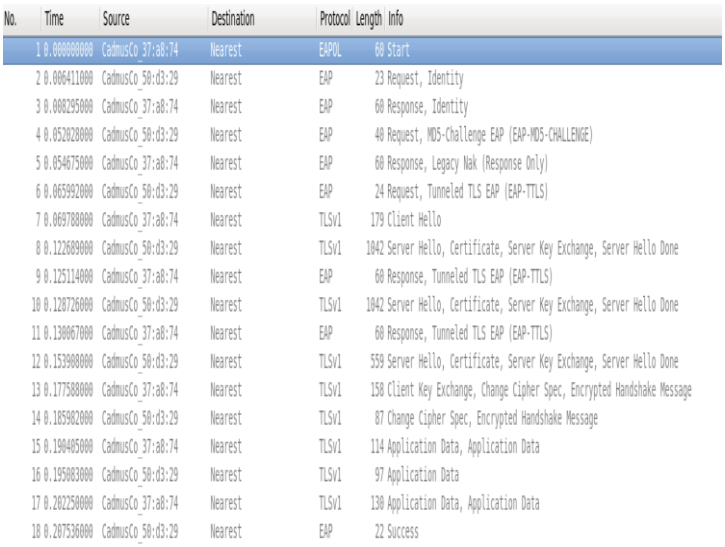


Figure 6 – Successful authentication with WPA

V. RESULTS

The setup used for the proof of concept was also used to obtain results and access the overall performance of the system. The tests were executed on a computer with 4GB of RAM, a 2.0 Ghz dual-core processor with a Windows 7 SP1 64 bit operating system. Both server and clients were setup in the same machine.

The tests focused on determining the execution time of the client side java program from the moment it issues the file request and until the file is ready to be used with the wpa_supplicant command. Different tests were made using different versions of the AES algorithm.

Each test was executed 10 fold and histograms, per AES version, were plotted. The number of times a specific value was obtained is represented as absolute frequency.

Additionally, the relative frequency, in proportion to the time interval, is also shown.

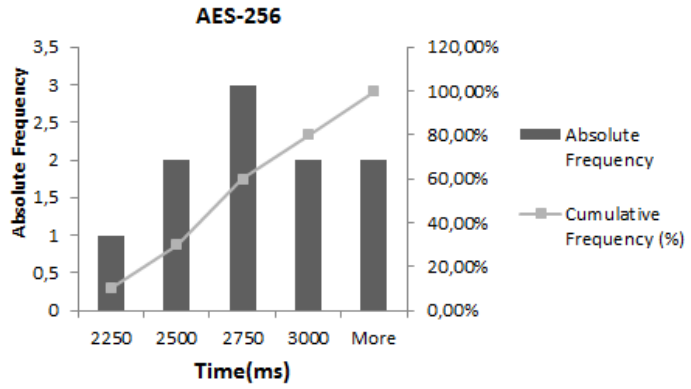


Figure 7 - Client side java program run time (AES 256)

Figure 7 shows the histogram of the times obtained while using the AES-256 version of the AES algorithm. The most frequent value (30% of all values) is in the 2500-2750ms intervals.

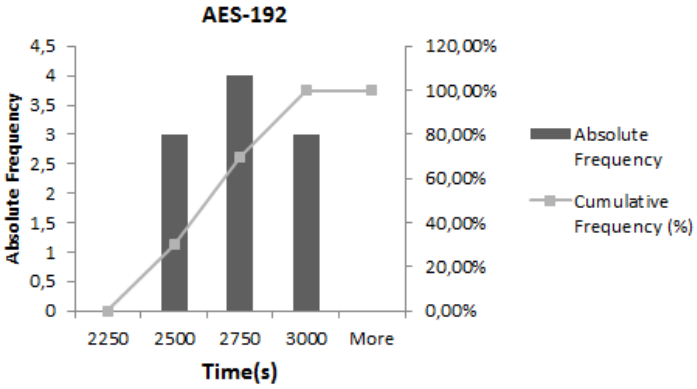


Figure 8 - Client side java program run time (AES 192)

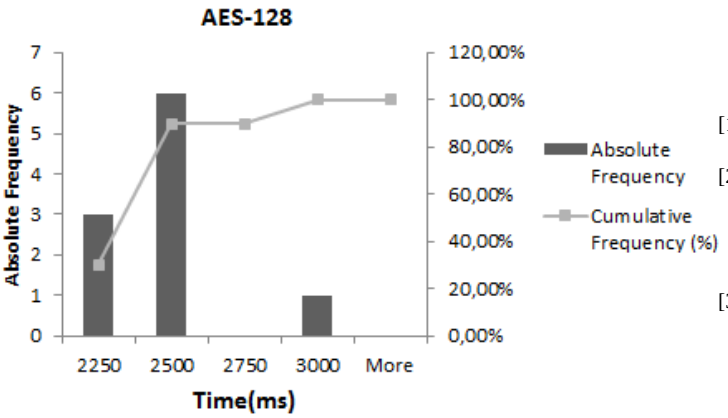


Figure 9 - Client side java program run time (AES 128)

Figure 8 shows the histogram of the times obtained while using the AES-192 version of the AES algorithm. The most frequent value (40% of all values) is in the 2500-2750ms intervals.

Figure 9 shows the histogram of the times obtained while using the AES-128 version of the AES algorithm. The most frequent value (60% of all values) is in the 2251-2500ms intervals.

Table 1 – Average executions time

Encryption	Average (ms)
AES - 256	2697.3
AES - 192	2615.4
AES - 128	2327.2

Table 1 shows the average duration per AES version. The average duration was of 2327 ms for the AES-128, 2615 ms for the AES-192, and for the more demanding AES-256 version of the algorithm.

VI. Conclusion

Public wireless hotspots based on captive portals are ubiquitous in places such as airport, trains stations and even households. Users connect to the hotspot with open wireless links, hampering the security, confidentiality and privacy of their Internet usage. The adoption of open wireless links results of its easy of use and automatic configuration.

The proposed solution maintains the use and mode of operation of traditional captive portal while enabling the automatic configurations of secure wireless networks by exchanging network configurations files and by automatically setting them up. The solution is transparent and minimizes its impact in the way users are expected to use captive portals. A proof of concept was setup and initial results are presented.

Acknowledgments

This work is financed by the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project UID/EEA/50014/2013.

References

[1] Wi-Fi Alliance, “Wi-Fi Protected Access, version 2.0,” 2003.

[2] A. Asheralieva, T. J. Erke, and K. Kilkki, “Traffic characterization and service performance in FON network,” in *2009 1st International Conference on Future Information Networks, ICFIN 2009*, 2009, pp. 285–291.

[3] G. Camponovo and A. Picco-Schwendener, “Motivations of hybrid wireless community participants: A qualitative analysis of Swiss FON members,” *Proc. - 2011 10th Int. Conf. Mob. Business, ICMB 2011*, pp. 253–262, 2011.

[4] S. Willens, A. C. Rubens, C. Rigney, and W. A. Simpson, “RFC [2865] - Remote Authentication Dial In User Service (RADIUS),” *IETF(Internet Eng. Task Force)*.

[5] D. Mitton, “RFC [2882]-Network Access Servers Requirements: Extended RADIUS Practices,” *IETF(Internet Eng. Task Force)*.

[Online]. Available: http://ietf.org/rfc/rfc2882.txt. [Accessed: 25-Sep-2015].

[6] E. Sadot, L. L. Yang, and P. Zerfos, "Architecture Taxonomy for Control and Provisioning of Wireless Access Points (CAPWAP)," *IETF(Internet Eng. Task Force)*, 2005.

[7] B. Aboba and D. Thaler, "What Makes For a Successful Protocol?," *IETF(Internet Eng. Task Force)*, 2008.

[8] A. P. <andreipo@microsoft.com>, "Prohibiting RC4 Cipher Suites," *IETF(Internet Eng. Task Force)*, 2015.

[9] C. Boulton, S. Pietro Romano, H. Schulzrinne, and M. Barnes, "Centralized Conferencing Manipulation Protocol," *IETF(Internet Eng. Task Force)*, 2012.

[10] S. Kelly, S. Frankel, and R. Glenn, "The AES-CBC Cipher Algorithm and Its Use with IPsec," *IETF(Internet Eng. Task Force)*, 2003.

[11] J. R. Vollbrecht, B. Aboba, L. J. Blunk, H. Levkowetz, and J. Carlson, "Extensible Authentication Protocol (EAP)," *IETF(Internet Eng. Task Force)*, 2004.

[12] P. Funk and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)," *IETF(Internet Eng. Task Force)*.

[13] D. Lloyd, B., L&A, Simpson, W., "PPP Authentication Protocols," *IETF(Internet Eng. Task Force)*, 1992. [Online]. Available: https://www.ietf.org/rfc/rfc1334.txt. [Accessed: 26-Sep-2015].

[14] G. Zorn and S. Cobb, "Microsoft PPP CHAP Extensions," *IETF(Internet Eng. Task Force)*, 1998.

[15] G. Z. <gwz@acm.org>, "Microsoft PPP CHAP Extensions, Version 2," *IETF(Internet Eng. Task Force)*, 2000.

[16] T. R. Kothaluru, "Evaluation of EAP Authentication Methods in Wired and Wireless Networks," no. October, 2012.

Reconfiguração de Redes de Sensores Sem Fios: Estratégias e Análise Comparativa

Emanuel Lima
Instituto de Telecomunicações
Porto
emanuellima@fe.up.pt

Óscar Gama, Paulo Carvalho
Centro Algoritmi, Universidade do Minho
Campus de Gualtar, Braga
pmc@di.uminho.pt

***Resumo* - A reprogramação das redes de sensores sem fios (RSSF) é essencial para garantir a atualização e a versatilidade deste tipo de redes. As necessidades de reprogramação prendem-se com as mais variadas razões como por exemplo, mudanças no meio ambiente, atualizações de *software*, melhorias de segurança, e alterações nos objetivos aplicacionais. Para tal, há necessidade de disseminar o código que se pretende instalar nos nós utilizando a comunicação sem fios, pois muitas vezes é a única forma de alcançar os nós depois de estes terem sido colocados no meio. Este artigo tenta identificar e avaliar quais os requisitos e as funcionalidades das principais estratégias de reprogramação usadas atualmente. É efetuada uma análise comparativa das principais características subjacentes, por forma a contribuir para uma clarificação de conceitos e promover a obtenção de soluções de reprogramação de RSSF mais flexíveis e com um melhor desempenho.**

Palavras chave - Redes de Sensores sem Fios; Reprogramação de Redes de Sensores; Protocolos de Disseminação.

I. Introdução

Com a rápida evolução dos dispositivos eletrónicos, nomeadamente na área dos sensores, e com estes cada vez mais pequenos, as Redes de Sensores Sem Fios (RSSF) têm assegurado um lugar de destaque no que diz respeito à monitorização e controlo de dados nas mais diversas áreas de atividade. O conceito subjacente a este tipo de redes é ter um conjunto de nós, constituídos por sensores físicos, um microprocessador e um módulo de comunicação sem fios, que apesar de terem uma capacidade individual de computação e comunicação limitada, quando trabalham em conjunto permitem alcançar e monitorizar diversos ambientes com um custo de implementação e manutenção reduzidos.

À medida que aplicações RSSF proliferam, o número de nós utilizados aumenta, e a necessidade de os gerir remotamente torna-se predominante. A necessidade de reprogramar os nós prende-se com o facto de estes, muitas vezes, passarem longos períodos de tempo em funcionamento, sendo a sua reprogramação necessária devido a mudanças no meio ambiente, atualizações de software, melhorias de segurança, a alterações nos objetivos aplicacionais, etc. A reprogramação remota é fundamental, pois dependendo do número de nós e da sua localização, a reprogramação direta e tradicional pode-se tornar muito fastidiosa, e além disso, dependendo do tipo de aplicações, existe frequentemente a necessidade de uma rápida reconfiguração de toda a rede, o que com o método descrito anteriormente não é viável.

A reprogramação remota das RSSF tira partido da comunicação sem fios dos nós utilizando estratégias de disseminação do código de forma que este possa chegar a todos os nós e assim ser possível a sua reconfiguração. Contudo, o desenho e implementação de estratégias de reconfiguração e disseminação é uma tarefa bastante complexa pois têm de ser consideradas as restrições dos nós a nível energético, computacional, armazenamento, comunicação e heterogeneidade de aplicações e hardware.

Neste trabalho são identificadas e analisadas várias estratégias de disseminação, sendo avaliados os requisitos a satisfazer na reprogramação dos nós de uma rede. Em particular são estudadas e comparadas estratégias baseadas em distribuição de toda a imagem do código com estratégias baseadas na distribuição das diferenças entre versões do código.

Assim sendo, este artigo está organizado da seguinte forma: após contextualização da relevância da reprogramação das RSSF, os vários tipos de reprogramação são descritos na secção II, identificando as principais abordagens que têm sido propostas na literatura; a análise e discussão das estratégias de disseminação é incluída na secção III; e a conclusão do trabalho apresentada na secção IV.

II. Reprogramação em Redes de Sensores Sem Fios

O primeiro passo na reprogramação sem fios dos nós é saber quando existe a necessidade de reprogramação e que parte do código necessita ser alterado (todo ou parcial). Esta decisão pode ser tomada pelo utilizador ou automaticamente pela rede. No primeiro caso, a ordem de reprogramação da RSSF é sempre dada por um utilizador enquanto os nós comportam-se simplesmente como escravos executando as ordens recebidas. O SNMS (Sensor Network Management System) [1] é um exemplo deste tipo de sistemas, fornecendo um protocolo que permite criar uma árvore da topologia da rede e um sistema de perguntas que permite inferir o estado e desempenho da mesma. As respostas são armazenadas e, caso exista necessidade de alterar o código nos nós, é utilizada esta árvore juntamente com o protocolo DRIP[1]. A vantagem deste tipo de sistemas que utilizam árvores é não obrigar a que os nós tenham todos os vizinhos nas suas tabelas de encaminhamento, pois só precisam de saber qual a sua posição

[1] DRIP-Protocolo de disseminação de tráfego de controlo que permite alcançar a coerência de um valor de uma determinada variável partilhada na rede.

na árvore e o *next-hop*. A desvantagem é que este tipo de sistemas só funciona em redes onde os nós são estáticos.

No caso onde a decisão de reprogramação é iniciada automaticamente pelos nós, o utilizador transmite a nova versão do código para pelo menos um nó. Neste caso, os nós comunicam uns com os outros, avaliam se a sua versão de código está desatualizada e, se for o caso, iniciam o processo de reprogramação. Este tipo de abordagem é sem dúvida a mais interessante, pois permite reconfigurar todo o tipo de nós, móveis e estáticos, além de permitir uma maior autonomia à rede, pois não é necessária intervenção externa. Neste tipo de abordagem os protocolos de reprogramação assumem um papel fundamental uma vez que são responsáveis por distribuir e gerir as atualizações de código por todos os nós da rede. O seu desenho e funcionamento deve-se basear nas seguintes diretrizes: (i) seleção dos nós que vão transmitir a informação (e.g., ter muitos nós a transmitir localmente ao mesmo tempo aumenta a probabilidade de colisões e não garante a otimização dos recursos); (ii) gestão da fragmentação da informação, de forma a ter em conta o limite de memória e recursos dos nós; e (iii) esquemas de garantia de fiabilidade, i.e., quando há perdas determinam quem e como vão ser feitas as retransmissões (e.g., podem ser feitas por *unicast*, *broadcast* ou *multicast*, ou podem ser utilizados esquemas como o FEC2 (*Forward Error Correction*), etc.).

As métricas para avaliar os protocolos de reprogramação prendem-se principalmente com: (i) questões de sobrecarga (*overhead*); (ii) o gasto energético, que tem influência direta no tempo de vida dos componentes da rede; (iii) a latência na deteção da necessidade de atualizações; e (iv) o tempo da própria disseminação.

A reprogramação ao nível do sistema continua a ser a forma mais fácil e mais comum de reprogramar uma rede de sensores sem fios. Neste tipo de reprogramação, a aplicação que corre no nó é substituída na íntegra por uma imagem binária da nova aplicação, juntamente com o código relativo ao sistema operativo. Desta forma, a substituição é completa não só a nível aplicacional como a nível do sistema operativo. Neste processo, a nova imagem é guardada na memória *flash* externa sendo depois copiada para a memória interna do microcontrolador através de um *bootloader*. Esta técnica de reprogramação pode adoptar duas abordagens diferentes no que diz respeito ao envio do novo código. Uma das abordagens consiste em enviar todo o código para os nós, enquanto a outra passa pelo envio das diferenças entre versões de código (delta). Na primeira abordagem é utilizada a funcionalidade básica da reprogramação onde o nó recebe o código e inicia imediatamente a sua reprogramação. Contudo, na segunda abordagem, o nó recebe apenas o delta, utilizando esta informação para gerar o novo código e só depois inicia a sua reprogramação.

De seguida, são apresentados e discutidos os principais estudos que têm sido desenvolvidos seguindo os dois tipos de abordagens referidas anteriormente.

A. Estratégias baseadas na distribuição de uma imagem completa de código

Apesar de esta abordagem ser bastante simples e robusta é também a mais ineficiente, pois o grande tamanho das imagens de código implicam um maior gasto energético na transmissão. Além disso, mesmo uma pequena alteração no código obriga a que seja gerada uma nova e completa imagem de código para ser transmitida para a rede, e assim se efetuar a reprogramação dos nós.

O *Crossbow Network Reprogramming Protocol* (XNP) [2] é o módulo de reprogramação utilizado no TinyOS versão 1.1. Este implementa a função básica de reprogramação fazendo *broadcast* do novo código para múltiplos nós por comunicação *single-hop* bidirecional, dentro do raio de alcance da estação base. Contudo, este protocolo não suporta métodos para reduzir o tamanho das imagens do código, nem comunicação *multihop*.

O Deluge [3] utiliza anúncios de versões de código para detetar a necessidade de atualizações na rede. Além disto, este protocolo divide o código a ser transmitido num conjunto de páginas com tamanho fixo. Esta rigidez no tamanho do conteúdo a ser transmitido permite multiplexação espacial, pois cada página é encarada como um objeto independente. Desta forma, a propagação do código é feita mais rapidamente.

Em [4], é proposto um novo protocolo como uma extensão ao Deluge que permite uma reprogramação seletiva dos nós da rede através do tipo de plataforma ou do identificador de cada nó (NodeID). A solução proposta aumenta claramente a funcionalidade do Deluge contribuindo para a reprogramação seletiva de RSSFs que possuam heterogeneidade de plataformas e aplicações.

O Stream [5, 6] é baseado no Deluge e otimiza o que é enviado pelo canal de comunicação. No Stream, o protocolo de reprogramação é instalado em cada nó antes de ser colocado no meio. Este processo é feito através da segmentação da memória *flash*. A vantagem é que, ao contrário do que acontece no Deluge onde é enviada a imagem do código e o protocolo de reconfiguração, no Stream é enviada a imagem do código e apenas a informação sobre qual o protocolo de reprogramação a utilizar. Isto faz com que a sobrecarga da rede diminua bastante, tornando-o assim mais eficiente a nível energético e mais rápido. Este protocolo é implementado no TinyOS na família Mica.

O DStream [7] é um protocolo híbrido que tem a capacidade de reprogramação *single-hop* e *multihop*. É baseado no Stream tirando partido de todas as suas vantagens em relação ao tamanho do código a ser enviado e à segmentação da memória. Os autores concluíram que em determinados cenários a reprogramação *single-hop* pode ser mais prática e eficaz que a reprogramação *multihop*. Assim sendo, o utilizador desloca-se até perto dos nós que pretende reprogramar e reprograma-os remotamente. Foram utilizadas análises matemáticas e cenários simulados e reais para identificar as condições que favorecem a reprogramação *single-hop* e *multihop*. Os autores concluíram que para redes

onde as ligações são bastante fiáveis, a reprogramação *multihop* é favorecida; contudo, para redes que sejam lineares (ou próximo de lineares) a reprogramação *single-hop* tende a ser melhor.

O SYNAPSE [8] é um protocolo cujo objetivo é melhorar a eficiência na recuperação de perdas. Enquanto outros protocolos tentam conseguir algoritmos inteligentes de forma a modificar esquemas epidémicos, o SYNAPSE foca-se em soluções extremamente eficientes para a transferência de dados nó-a-nó. Utiliza o paradigma *three-way handshake* do Deluge, o método utilizado pelo Stream na transmissão de dados e utiliza NACKs para as retransmissões.

O Infuse [9] é um protocolo fiável de disseminação que utiliza o TDMA como estratégia de controlo de acesso ao meio. O Infuse utiliza comunicação *multihop* e permite: (i) selecionar o algoritmo para lidar com a recuperação de erros inesperados provenientes do canal de transmissão (e.g. erro na receção das mensagens, variações da força do sinal); e (ii) especificar quando um nó deve ouvir um conjunto de nós vizinhos de forma a reduzir o tempo de escuta. Baseado nas opções tomadas, é possível obter quatro modos de operação do Infuse.

De forma a reduzir o problema de colisões e do nó escondido, o *Multihop Network Reprogramming Protocol* (MNP) [10] incorpora um algoritmo de seleção do nó emissor. Este algoritmo tenta garantir que numa determinada vizinhança exista pelo menos uma fonte de cada vez a transmitir a imagem do novo código. A seleção deste nó emissor passa por tentar escolher aquele que se espera que tenha mais impacto na rede. O MNP também utiliza *pipelining* para permitir melhorar a taxa de transmissão dos dados e de retransmissão, em caso de falhas. O MNP é eficiente a nível energético pois reduz o tempo que o módulo de comunicação do nó está ativo, colocando-o em modo *sleep* sempre que um vizinho está a transmitir um segmento de código que não é do seu interesse. É ainda utilizado outro mecanismo que coloca o emissor nesse mesmo estado sempre que não existe nenhum vizinho interessado no que vai ser transmitido. O MNP possibilita como opção adicional reduzir o consumo energético na fase inicial da reprogramação, permitindo configurar os níveis de serviço prestados pelo nó, tendo em conta a sua capacidade energética. Por exemplo, o nó pode ser configurado para que a probabilidade de ser o nó responsável pela transmissão da imagem do código seja proporcional ao que resta da sua bateria.

O OAP-SW [11] é apresentado como sendo um protocolo que utiliza características de pequeno mundo (Redes de pequeno mundo[2]) para melhorar a reconfiguração da rede. Este protocolo cria atalhos dentro da rede até aos nós que têm maior capacidade de processamento e utiliza-os na reprogramação. Provou-se que os tempos de reconfiguração foram drasticamente reduzidos e que o consumo energético da rede não foi consideravelmente afetado com o uso de um pequeno conjunto de nós mais potentes.

O Gappa [12] é um protocolo epidémico *multihop*, onde alguns nós recebem uma parte do novo código por um canal e transmitem aos vizinhos o que acabaram de receber por um canal distinto. Ao explorar a comunicação por multi-canal e a técnica de *pipeline*, o Gappa possibilita uma elevada concorrência em diferentes canais e em diferentes localizações, permitindo assim uma rápida transmissão de dados. Para reduzir as colisões em cada canal, o Gappa utiliza um algoritmo de seleção de emissor multi-canal que tenta garantir que, em qualquer vizinhança, está a transmitir no máximo um emissor de cada vez por canal. Durante este processo existe uma tentativa de dar preferência aos emissores que tenham mais impacto na rede. Se um emissor não conseguir transmitir num canal, pode tentar escolher outro canal e, caso não consiga em nenhum, o nó deixa de enviar pedidos de autorização da transmissão. Durante este tempo, o nó pode escolher receber o código, deixando para isso o seu módulo de comunicação ligado, ou desligado, caso esteja a funcionar só como emissor e tenha pouca bateria. Desta forma, o Gappa diminui também o tempo em que o módulo de comunicação de cada nó está ativo, diminuindo assim o consumo energético durante a reprogramação.

O *Multihop Over-the-Air Programming* (MOAP) [13] é um mecanismo de distribuição de código especialmente construído para os nós Mica-2. O MOAP foi desenhado para ser eficiente ao nível da memória e da energia à custa do aumento da latência. As escolhas no desenho foram baseadas em três áreas: protocolo de disseminação, mecanismos de retransmissão e gestão de armazenamento dos segmentos de código.

Freshet [14] é um protocolo para otimizar o consumo de energia para o *upload* do código e aumentar a velocidade de disseminação quando estão presentes múltiplas fontes. Neste caso, os nós ligam-se a um fonte que começa a disseminação do código sendo utilizado um mecanismo para lidar com as colisões. O Freshet permite que os nós recebam as páginas fora de ordem. Esta característica, juntamente com o facto dos nós poderem escutar as comunicações dos vizinhos (pois o meio de comunicação é partilhado), possibilita que numa situação em que estejam várias fontes a transmitir, os nós possam receber as páginas de código em falta de outra fonte sem ser aquela a que originalmente se conectou. O Freshet tem três fases: *blitzkrieg*, *distribution* e *quiescent*. Quando um novo código é introduzido na rede, a fase *blitzkrieg* é iniciada. Esta fase consiste na rápida propagação de informação sobre o código juntamente com alguma informação sobre a topologia da rede. A informação sobre a topologia da rede vai permitir aos nós calcular o tempo que o código irá demorar até chegar à sua vizinhança, iniciando-se assim a segunda fase, que consiste na distribuição do código. Os nós podem entrar em modo *sleep* durante a primeira e a segunda fases, poupando energia. A terceira fase ocorre quando não existe nenhum novo código para ser transmitido. Neste caso, o Freshet reduz exponencialmente a taxa de transmissão de metadados para poupar mais energia.

[2] Tipo de grafo matemático no qual grande parte das conexões são estabelecidas entre os vértices mais próximos, apresentando-se como um mundo pequeno. Neste tipo de rede, a distância média entre quaisquer dois vértices não ultrapassa um número pequeno de vértices.

B. Estratégias baseadas na distribuição das diferenças entre versões de código

O Zephyr [15] é apresentado como sendo um protocolo de reprogramação incremental. Neste protocolo é enviado um delta para cada nó quando existe a necessidade de atualização. Este delta, que reflete as modificações entre o código antigo e o novo, vai ser utilizado pelo nó para reconstruir o novo código. Em [16], os autores do Zephyr, apresentam o Hermes como sendo uma melhoria do Zephyr. Este reduz o tamanho do delta utilizando técnicas para atenuar o efeito de mudanças em funções e variáveis globais causadas pelas diferenças no código.

O R2 [17] reduz o tamanho do delta através da utilização eficiente de módulos dinâmicos de *loading* e *linking*. No R2 são utilizados metadados que são enviados para os nós de forma a eliminar a utilização de uma *jump table* (que é um requisito no Hermes [16] e Zephyr [15]) e um *bootloader* complexo. Este protocolo foi implementado no TinyOS 2.1 sendo comparado com o Deluge [3], o Zephyr e o Hermes, onde mostrou uma redução no custo de transmissão de 65% quando comparado com o Deluge e 20% quando comparado com o Zephyr e Hermes.

Através da utilização do delta, a quantidade de informação que é enviada para a rede diminui consideravelmente. Contudo, a grande desvantagem deste tipo de estratégia é que são executados algoritmos complexos e pesados nos nós para criar a nova imagem de código com base no delta.

III. ANÁLISE E DISCUSSÃO DAS ESTRATÉGIAS DE DISSEMINAÇÃO

Relativamente ao que foi exposto nas secções anteriores e da análise mais detalhada das propriedades dos protocolos de disseminação expressas na Tabela 1, podem ser feitas várias considerações sobre os aspetos a ter em conta no desenho de protocolos deste tipo, nomeadamente:

(i) Disseminação *single-hop* vs. *multihop*: Ao longo do tempo as atenções na reprogramação das RSSF tem vindo a passar de *single-hop* para *multihop* devido ao facto de esta abordagem ser mais prática e mais eficiente em termos temporais. A escolha entre estes tipos de reprogramação depende de fatores como o tamanho da rede ou a densidade dos nós, mas o mais importante é a fiabilidade das ligações. A resposta pode estar na conceção de protocolos híbridos como o DStream [7] que permitem ao utilizador uma reprogramação de nó em *single-hop* ou *multihop*.

(ii) Disseminação da imagem completa de código vs. diferenças entre versões: Apesar do envio da imagem completa do código ser a abordagem mais simples é também a mais ineficiente, pois são enviados mais dados aumentando os períodos de transmissão e, consequentemente, o consumo energético do nó. Outra desvantagem prende-se com o fato de pequenas alterações no código obrigarem à geração de uma nova imagem de código da aplicação para ser transmitida. O envio das diferenças entre versões do código (delta) permitem diminuir consideravelmente o *overhead* na transmissão, tornando este tipo de protocolos mais rápidos na disseminação da nova imagem de código. Contudo, são necessários algoritmos complexos e pesados nos nós para criar a nova imagem de código com base no delta. Assim sendo, uma abordagem que encontrasse um compromisso entre estas estratégias, permitindo escolher a melhor com base no esforço necessário para gerar uma nova imagem de código a partir do delta poderia ser uma mais valia.

(iii) Distribuição hierárquica vs. direta: Numa reprogramação hierárquica, as atualizações são primeiro enviadas para os nós que estão na camada superior da hierarquia e só depois para o resto das camadas, seguindo a lógica hierárquica. Este tipo de estratégia é mais escalável tornando-se mais eficiente em redes com maior número de

	Single-hop vs. Multihop	**Imagem do código transmitida**	**Tipo de disseminação do código**	**Distribuição dos dados inteiros/pipeline**	**Reprogramação total vs. seletiva**
XNP [2]	*Single-hop*	Toda	Direta	Inteiros	Total
Deluge [3]	*Multihop*	Toda	Direta	*Pipeline*	Total
Deluge Ext. [4]	*Multihop*	Toda	Direta	*Pipeline*	Seletiva
Stream[5, 6]	*Multihop*	Toda	Direta	*Pipeline*	Total
DStream [7]	Híbrido	Toda	Direta	*Pipeline*	Seletiva
SYNAPSE [8]	*Multihop*	Toda	Direta	*Pipeline*	Total
Infuse [9]	*Multihop*	N/A	Direta	*Pipeline*	Total
MNP [10]	*Multihop*	Toda	Direta	*Pipeline*	Total
OAP-SW [11]	*Multihop*	Toda	Hierárquico	*Pipeline*	Total
Gappa [12]	*Multihop*	Toda	Hierárquico	*Pipeline*	Total
MOAP [13]	*Multihop*	Toda	Direta	Inteiros	Total
Freshet [14]	*Multihop*	Toda	Direta	*Pipeline*	Total
Zephyr [15]	*Multihop*	Diferenças	Direta	*Pipeline*	Total
Hermes [16]	*Multihop*	Diferenças	Direta	*Pipeline*	Total
R2 [17]	*Multihop*	Diferenças	Direta	*Pipeline*	Total

Tabela 1. Análise comparativa das principais características dos protocolos de reprogramação em estudo.

nós, comparativamente com a estratégia direta onde não existe qualquer diferenciação dos nós. Com esta reprogramação hierárquica pode-se colocar nós específicos em determinada camada hierárquica , nomeadamente nós que tenham mais recursos ou que se situem em determinada posição geográfica que lhes permita alcançar o maior número de vizinhos possível.

(iv) Reprogramação total vs. seletiva: Aqui torna-se evidente que a estratégia de reprogramação deve suportar uma reprogramação seletiva de determinados nós. Isto porque é esperado que existam diferentes aplicações a correr na mesma RSSF. Desta forma, uma reprogramação seletiva é conveniente pois permite selecionar quais os nós a serem reprogramados e que tipo de aplicação vai ser afetada, sendo assim mais flexível e eficiente. A seleção poderá ser manual através dos identificadores dos nós ou automática, selecionando um conjunto de requisitos que devem ser satisfeitos pelos nós de forma a identificar quais vão ser reprogramados. A solução ideal será suportar os dois tipos de reprogramação seletiva para melhor responder às necessidades de diferentes RSSF.

(v) Distribuição de dados inteiros vs. *pipelining* e retransmissões: Uma estratégia que tem sido amplamente utilizada nos esquemas de disseminação é o *pipelining*. O principio da reprogramação baseado em *pipeline* consiste em dividir o programa em segmentos mais pequenos e enviá-los um de cada vez ou em paralelo. Isto permite aumentar a velocidade e diminuir o tempo das retransmissões que são normalmente um problema devido à pouca fiabilidade das ligações nas RSSF. Algumas estratégias a ter em conta no que diz respeito às retransmissões são utilizadas pelo SYNAPSE [8].

IV. Conclusão

Muito se tem feito para tentar resolver o problema de obter uma estratégia de reprogramação dos nós de uma RSSF fiável e genérica para todos as topologias. Contudo, esta solução ainda não foi alcançada pois trata-se de um problema complexo. Neste trabalho, fez-se uma análise comparativa das principais soluções para reconfigurar RSSF, por forma a: (i) contribuir para uma maior clareza nesta área de investigação e desenvolvimento; e (ii) promover a convergência para uma solução que, dependendo do tipo de aplicações e de métricas a ela associadas, seja flexível e forneça o melhor desempenho possível.

Agradecimentos

Este trabalho é apoiado pela FCT - *Fundação para a Ciência e Tecnologia* no âmbito do projeto UID/CEC/00319/ 2013.

Referências

[1] Tolle, G., Culler, D.: Design of an application-cooperative management system for wireless sensor networks. Proc. of the 2nd European Workshop on Wireless Sensor Networks, pp.121–132, 2005.

[2] Jeong, J., Kim, S., Broad, A: Network reprogramming. University of California, Berkeley, CA, USA, 2003.

[3] Chlipala, A., Hui, J., Tolle, G: Deluge: data dissemination for network reprogramming at scale. Univ. of California, Berkeley, Tech. Rep. (2004).

[4] Lima, Emanuel, et al. "A Protocol Extension for Selective Reprogramming of WSNs" Proceedings of the 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM). Croacia, 2015.

[5] Panta, R.K., Bagchi, S., Khalil, I.M.: Efficient wireless reprogramming through reduced bandwidth usage and opportunistic sleeping. Ad Hoc Networks. 7, pp. 42–62, 2009.

[6] Panta, R., Khalil, I., Bagchi, S.: Stream: Low overhead wireless reprogramming for sensor networks. INFOCOM 2007, 2007.

[7] Panta, R., Bagchi, S.: Single versus multi-hop wireless reprogramming in sensor networks. Proc. of the 4th Int. Conf. on Testbeds and Research Infrastructures for the Development of Networks & Communities, TridentCom '08, 2008.

[8] Rossi, M., Zanca, G., Stabellini, L., Crepaldi, R., Harris III, A.F., Zorzi, M: SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks Using Fountain Codes. 2008 5th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Networks. pp. 188–196, 2008.

[9] Kulkarni, S.S., Arumugam, M.: Infuse: A TDMA based data dissemination protocol for sensor networks. Int. Journal of Distributed Sensor Networks, 2, pp. 55–78, 2006.

[10] Kulkarni, S.S.: MNP: Multihop Network Reprogramming Service for Sensor Networks. 25th IEEE Int. Conf. Distrib. Comput. Syst. pp.7–16, 2005.

[11] Maia, G., Guidoni, D.: Improving an over-the-air programming protocol for wireless sensor networks based on small world concepts. Proc. of the 12th ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '09), 2009.

[12] Wang, L., Kulkarni, S.: Gappa: Gossip based multi-channel reprogramming for sensor networks. Distributed Computing in Sensor Systems, Vol. 4026, Lecture Notes in Computer Science, Springer, pp 119-134, 2006.

[13] Stathopoulos, T., Heidemann, J., Estrin, D.: A remote code update mechanism for wireless sensor networks, Tech. Rep., 2003.

[14] Krasniewski, M.D., Panta, R.K., Bagchi, S., Yang, C.-L., Chappell, W.J.: Energy-efficient on-demand reprogramming of large-scale sensor networks. ACM Trans. Sens. Networks. 4, pp. 1–38, 2008.

[15] Panta, R.K., Bagchi, S., Midkiff, S.P.: Efficient incremental code update for sensor networks. ACM Trans. Sens. Networks. 7, pp. 1–32, 2011.

[16] Panta, R.K., Bagchi, S.: Hermes: Fast and Energy Efficient Incremental Code Updates for Wireless Sensor Networks. IEEE INFOCOM 2009 - 28th Conf. Comput. Commun. pp. 639–647, 2009.

[17] W. Dong, Y. Liu, C. Chen, J. Bu, C. Huang, and Z. Zhao, "R2: Incremental reprogramming using relocatable code in networked embedded systems," Comput. IEEE Trans., vol. 62, no. 9, pp. 1837–1849, 2013.

Serviço de *E-mail* Marítimo Utilizando Redes Tolerantes ao Atraso

João Rodrigues
Instituto Superior de Eng. do Porto
Instituto Politécnico do Porto
Porto, Portugal
1130201@isep.ipp.pt

Jaime Dias
Centro de Telecomunicações e Multimédia
INESC Tec
Porto, Portugal
jdias@inescporto.pt

Jorge Mamede
INESC TEC e
Instituto Superior de Eng. do Porto,
Instituto Politécnico do Porto
Porto, Portugal
jbm@isep.ipp.pt

Resumo— *Delay Tolerant Network* (DTN) é uma arquitetura de redes que procura resolver os problemas associados à conetividade intermitente e possibilita a existência de comunicações em ambientes onde o protocolo tradicional TCP/IP não funciona. A arquitetura DTN é adequada a cenários com uma topologia de rede dinâmica, densidade de nós reduzida e contatos entre os nós de curta duração. Com a instalação de vários componentes, incluindo a arquitetura DTN através do protocolo Bundle, foi possível implementar um serviço de correio eletrónico adaptado ao cenário marítimo. Este artigo expõe a arquitetura do serviço, os componentes instalados, as especificações da *testbed*, e a avaliação feita ao desempenho do sistema de *e-mail*.

Palavras-chave—Delay Tolerant Network; DTN; E-mail; Comunicaçoes maritimas;

I. INTRODUÇÃO

O *e-mail* é hoje em dia uma das aplicações da Internet mais utilizadas em todo mundo. Contudo, nem todas as pessoas podem usufruir desse serviço quando se encontram em zonas remotas. Enquanto que em solo terrestre o acesso dos utilizadores ao serviço de *e-mail* está assegurado, no cenário marítimo devido sobretudo às caraterísticas geográficas isso não acontece. Os protocolos de encaminhamento de dados utilizados em terra, desenvolvidos para redes totalmente conectadas, não são adequados ao cenário marítimo, onde não existe qualquer infraestrutura de redes implementada. A rede marítima é caraterizada por contatos de breve duração entre os nós, uma topologia de rede altamente dinâmica e uma baixa densidade de nós, tornando a rede propensa a interrupções e significantes atrasos nas comunicações.

Protocolos de transporte como TCP, são baseados no princípio de comunicação interativa extremo-a-extremo. Ou seja, a transmissão de dados assenta em algumas premissas, tais como, a existência de uma conexão ininterrupta extremo-a-extremo entre origem e destino e latências reduzidas. No ambiente marítimo não é possível cumprir com estes requisitos, pelo que é necessário utilizar uma arquitetura de rede que seja tolerante a atrasos e interrupções.

Para suprir as dificuldades existentes nas comunicações marítimas e possibilitar a utilização de um serviço de *e-mail* em embarcações, recorreu-se à implementação da arquitetura de rede tolerante a atrasos, denominada *Delay Tolerant Network* (DTN) [1].

Neste trabalho, propõe-se uma solução de correio eletrónico baseada em DTN, que visa permitir a utilização desse serviço em zonas marítimas remotas. Neste artigo serão apresentados os resultados das experiências feitas ao protótipo desenvolvido. As experiências realizadas tiveram como objetivo avaliar o desempenho e eficiência dos componentes instalados, bem como o serviço de *e-mail* no seu todo.

A. Delay Tolerant Network

Delay tolerant network (DTN) é uma arquitetura de rede que possibilita a existência de comunicações em ambientes com conectividade intermitente, atrasos significativos e alta taxa de erros de transmissão, utilizando o princípio de funcionamento *hop-by-hop* na entrega de dados. As comunicações são assentes em contatos oportunísticos que acontecem entre os nós da rede, que proporcionam a transferência de dados através do paradigma *store-carry-forward*. O método *store-carry-forward*, muitas vezes chamado *store-and-forward*, permite que as mensagens ou fragmentos das mensagens sejam transferidos pela rede saltando de um nó para o outro, tendo como suporte o armazenamento persistente de cada nó (Fig. 1).

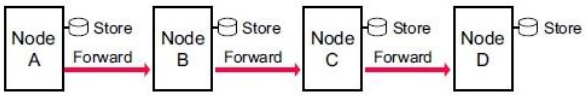


Fig 1. Mecanismo Store-Carry-Forward [2].

O armazenamento persistente é necessário em cada nó pelas seguintes razões:

- A ligação com outro nó e o envio de pacotes pode demorar um longo período de tempo.
- As velocidades de transmissão e a eficiência de atuação dos dois nós podem não ser iguais.
- Após a transmissão de uma mensagem, devido aos erros que podem ocorrer, ou caso o nó esteja configurado para inundar a rede (*flooding*), pode ser necessário retransmitir a mesma mensagem.´

A arquitetura DTN implementa em cada nó o princípio para troca de pacotes *store-carry-forward*, através da adição da camada protocolar Bundle. A camada Bundle serve para agregar as camadas protocolares inferiores, de forma a permitir que as aplicações consigam comunicar ao longo dos nós. Os nós podem ser constituídos por arquiteturas protocolares semelhantes ou por arquiteturas diferentes (Fig. 2). A camada Bundle é também responsável por enviar *bundles* entre os nós, sendo que *bundle* é também o nome das mensagens que circulam na rede DTN [2,3].

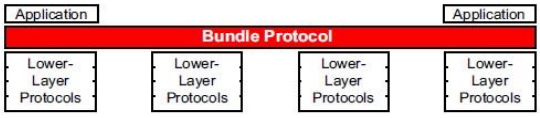


Fig 2. Camadas do protocolo DTN [2].

II. Trabalho Relacionado

Apesar do protocolo Bundle ter sido originalmente desenvolvido para comunicações interplanetárias, já foi várias vezes implementado em aplicações terrestres. Ao longo dos anos a arquitetura DTN foi estudada e proposta por diversos investigadores para solucionar vários problemas associados às comunicações em áreas remotas.

Em [4], os autores puseram em prática o que tinham em parte modelado dois anos antes em [5], e implementaram um sistema que permitiu fornecer o serviço de correio eletrónico a aldeias isoladas. Esse serviço foi fornecido através da instalação de uma rede baseada no conceito DTN, utilizando como parte da infraestrutura um sistema de transportes ferroviário real. O principal objetivo do estudo foi avaliar o desempenho do sistema implementado, verificando a qualidade do serviço de *e-mail* em ambiente real, mesmo quando não existe conetividade extremo-a-extremo.

O projeto N4C em [6], propôs uma solução baseada em DTN com o intuito de estender o legado da internet a regiões remotas, onde não é barato nem viável a existência de infraestruturas de comunicações permanentes. Esta proposta nasceu de um projeto de 36 meses, desenvolvido por um conjunto de 12 parceiros europeus, que consistiu no estudo, desenvolvimento e experimentação de uma arquitetura, uma infraestrutura e várias aplicações em ambiente real. Uma das aplicações implementadas neste projeto foi o serviço de *e-mail*.

Em [7], os autores propuseram a utilização da tecnologia DTN para ligar aldeias a cidades em África, utilizando equipamentos móveis com o sistema operativo *Android* para guardar a informação. Nesta proposta, as pessoas que viajam entre cidades e aldeias são responsáveis pelo transporte dos dados.

Autores em [8] estudaram o desempenho das comunicações em ambiente marítimo, comparando a eficiência de diferentes protocolos de encaminhamento tais como os tradicionais extremo-a-extremo Ad Hoc On Demand Distance Vector (AODV) e Optimized Link State Routing (OLSR), e os protocolos DTN, Epidemic Routing e Spray and Wait. Os resultados concluíram que os protocolos DTN conseguem globalmente atingir um melhor desempenho, do que os protocolos de encaminhamento tradicionais.

Em [9], foi avaliado o desempenho dos protocolos de encaminhamento utilizados em DTN. A avaliação foi feita num ambiente DTN rodoviário (*Vehicular Delay Tolerant Network*), de forma a perceber a eficiência dos vários protocolos quando existe uma topologia de rede altamente dinâmica, com contatos curtos entre os nós e conectividade intermitente.

III. Testbed do Sistema

Nesta secção é apresentada a *testbed* que serviu de plataforma à avaliação do desempenho da arquitetura, dos protocolos e das aplicações do protótipo implementado. A *testebed*, montada em ambiente controlado, foi modelizada tendo por base um cenário marítimo real constituído por embarcações, um nó presente na costa e a Internet, como está ilustrado na Fig. 3.

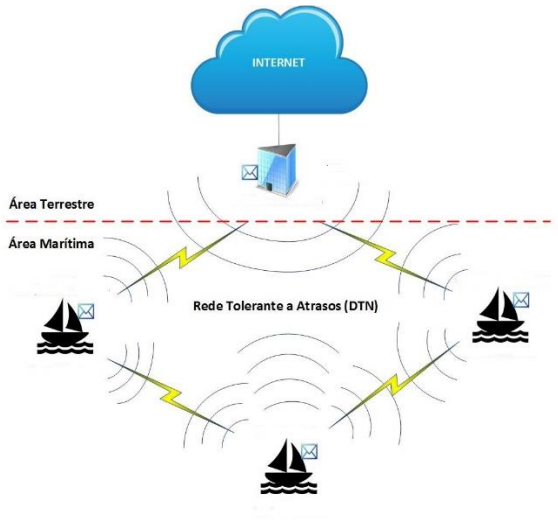


Fig 3. Esquema do Sistema.

A modelização do cenário marítimo foi feita através da utilização de máquinas virtuais, que foram configuradas individualmente, consoante as necessidades de cada nó. As máquinas virtuais foram implementadas sobre um computador portátil com um processador *dual core* e 4 Gbytes de memória RAM, sendo que cada nó foi implementado com 1 Gbyte de RAM e 8 Gbytes de armazenamento. Cada nó presente na rede é uma máquina virtual independente, sendo que as máquinas comunicam entre si através da rede interna do *software* de

virtualização. Portanto, para avaliar o desempenho do protótipo no contexto marítimo, foram feitas várias alterações à topologia da rede interna utilizada pelas máquinas virtuais, de forma a aproximar as condições dos testes às condições vividas no cenário marítimo. A partir da Fig.4 é possível obter, ainda que de forma abstrata, uma visão completa da estrutura do protótipo utilizado nos testes práticos.

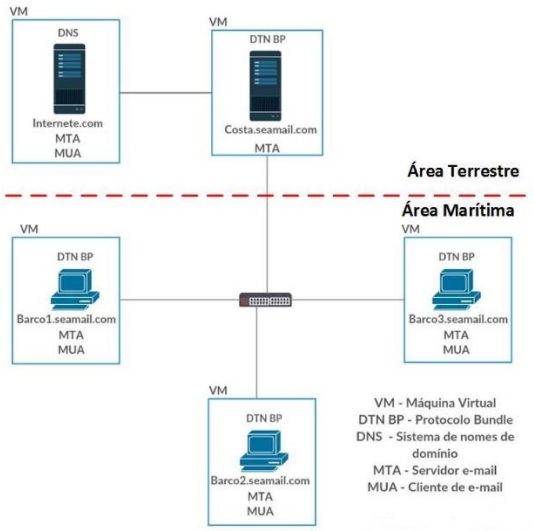


Fig 4. Arquitetura do Sistema.

A. *Componentes do sistema*

Para implementar um sistema de *e-mail* adaptado a redes DTN foi necessária a instalação de alguns componentes. Os principais componentes implementados sobre o sistema operativo Ubuntu 14.10 estão descritos a seguir.

- Protocolo Bundle - DTN-2.9.0 é a versão mais recente da implementação de referência do protocolo Bundle. O pacote dtn2, é o responsável por toda a arquitetura DTN instalada na máquina. O pacote dtn2 tem alguns requisitos, como a existência de uma base de dados persistente. Após a instalação, o protocolo Bundle é implementado em cada máquina através da inicialização do *daemon* dtn.
- Postfix – *Mail Transfer Agent*(MTA) utilizado em todas as máquinas do projeto. Postfix serve perfeitamente para o sistema proposto e é o MTA de referência na comunidade Linux.
- Integração do postfix com dtn2 - Aplicação composta por scripts em *python*, serve para integrar o servidor de *e-mail* com o *daemon* dtn. O programa recebe *e-mail*s do postfix, coloca-os em *bundles* e faz o processo inverso quando necessário. Antes de executar os scripts, é necessário instalar um API do *python*, para permitir ao programa utilizar funcionalidades da arquitetura dtn, sem se envolver em detalhes do software dtn2.

B. *Estrutura do sistema*

A estrutura do sistema é constituída por 3 tipos de nós. Contudo o número de nós presentes na *testbed* pode ser largamente superior, visto que, todas as máquinas que representam embarcações têm o mesmo tipo de configuração. A estrutura do protótipo é constituída pelos seguintes elementos:

- Nós “BarcoX.seamail.com” - Cada nó que representa uma determinada embarcação está equipado com o servidor de *e-mail* postfix, com o protocolo Bundle (software dtn2) e com uma aplicação integradora que faz a conversão de *e-mail*s em *bundles* e vice-versa, sendo que a única forma de comunicar com outros nós é através da arquitetura DTN. O servidor de *e-mail*s presente em cada embarcação é responsável por um domínio, por exemplo “barco1.seamail.com”. Caso as *bundles* que chegam às embarcações tenham como destinatário a designação DTN do próprio nó, as *bundles* são entregues às aplicações que irão extrair e passar o *e-mail* ao postfix. Caso tenham como destinatário outro nó DTN, as *bundles* ficam em armazenamento persistente para serem possivelmente transferidas para outros nós.
- Nó “Costa.seamail.com” – O nó da costa funciona como uma *gateway* para os *e-mail*s que estão de entrada ou de saída da rede DTN. O nó costeiro está configurado com o servidor de *e-mail* postfix, com o protocolo Bundle e com o software que faz a integração do postfix com o4 protocolo Bundle. O nó costeiro tem duas interfaces de rede, que permitem simultaneamente a interação com a rede DTN e com a rede terrestre. Os *e-mail*s que são recebidos pelo servidor da costa, caso tenham como destinatário um endereço com o formato “xxx@barcoX.seamail.com”, são entregues à interface DTN na forma de *bundles* para serem transmitidos. Caso o endereço destino dos *e-mail*s não seja nenhum utilizador de domínio “seamail.com”, a gestão dos *e-mail*s é feita de forma padrão e estes são entregues através do protocolo SMTP.
- Nó “Internete.com” – A Internet foi representada na *testbed* por um nó que se encontra fora da rede DTN. O nó foi implementado com o propósito de simular um servidor de *e-mail* presente na Internet. Este nó foi configurado como sendo simultaneamente um servidor DNS e um MTA. Para efeitos de testes foi utilizado este nó para interagir com o nó da costa, de forma a simular situações onde são enviados ou recebidos *e-mail*s de e para a rede DTN.

IV. Avaliação de desempenho

Nesta secção serão expostos os resultados dos testes práticos realizados à proposta apresentada neste trabalho, bem como o contexto em que os mesmos foram efetuados.

A. Cenários de testes

Os testes experimentais consistiram no envio e receção de *e-mail*s em diferentes cenários, arquitetados de forma a aproximar as condições do ambiente controlado às condições das comunicações marítimas, nomeadamente a falta de conetividade entre os nós. Os cenários utilizados nas experiências, retratam situações que ocorreriam com frequência no ambiente marítimo.

As métricas de desempenho consideradas nos testes foram os tempos de entrega dos *e-mail*s. Os tempos foram recolhidos a partir dos ficheiros *log* presentes nos nós, e são essencialmente o tempo consumido pelas aplicações e protocolos, pois, efetivamente os tempos de transmissão entre os nós são reduzidos. Todos os testes feitos neste estudo foram realizados em ambiente controlado, com o sistema imune às caraterísticas da navegação marítima, pelo que os *e-mail*s foram sempre entregues ao seu destino com maior ou menor dificuldade.

O primeiro cenário (cenário 1) foi constituído por dois nós DTN totalmente conectados, representando um cenário, onde dois barcos permanecem algum tempo em contato um com o outro, ou um cenário onde uma embarcação estabelece um contato duradouro com o nó da costa (Fig. 5). Neste cenário pressupõe-se que as *bundles* são geradas para envio apenas quando existe conetividade DTN com outros nós. Para isso, foi estabelecida a ligação DTN entre os dois nós, antes da criação dos *e-mail*s.

Fig 5. Cenário 1 – Dois nós DTN com conexão estabelecida.

No cenário 1, foram realizados dois conjuntos de testes. Na primeira série, foram enviados *e-mail*s com algum texto, que se converteram respetivamente em *bundles* de tamanho padrão 4 Kbytes. O objetivo desta série de testes foi verificar o tempo que demora a processar um tradicional *e-mail* de texto por parte das aplicações presentes nos dois nós.

Na segunda série de testes feita no cenário 1, foram enviados *e-mail*s de diferentes tamanhos entre os dois nós. Os *e-mail*s foram agrupados em 4 tamanhos diferentes e foram concebidos através da anexação de imagens, gerando *bundles* com tamanho entre 1,1 e 8,5 MB. O objetivo nesta série de testes, foi perceber de que forma o tamanho dos *e-mail*s influencia o tempo de entrega dos mesmos.

O segundo cenário (cenário 2) foi constituído igualmente por dois nós DTN. Contudo teve algumas diferenças face ao cenário 1, especialmente nas condições em que foram realizados os testes. Os testes realizados neste cenário foram projetados de forma a simular uma situação recorrente das redes DTN, onde dois nós encontram-se e trocam entre si as mensagens que têm armazenadas há algum tempo. No cenário marítimo esta situação poderia facilmente acontecer entre embarcações, ou entre embarcações e o nó da costa (Fig. 6), pois devido às caraterísticas do meio, é pouco provável que exista conetividade entre os nós no momento em que as *bundles* são geradas para envio.

Fig 6 - Cenário 2 – Dois nós DTN sem conexão previamente estabelecida.

No conjunto de testes executados no cenário 2, antes de criar cada *e-mail*, foi desativada a conexão entre os dois nós DTN. Após o *e-mail* ser criado, a conexão foi restabelecida entre os dois nós. Nesta série foram enviados *e-mail*s contendo apenas alguns bytes de texto, originando *bundles* com tamanho padrão. O objetivo deste conjunto de testes foi analisar o tempo que dois nós DTN demoram a sincronizar-se e a transmitir os *e-mail*s que têm em buffer. Neste cenário não se justificou fazer testes com *bundles* superiores a 4 Kbytes, pois os resultados seriam semelhantes aos obtidos com *bundles* de tamanho padrão, visto que o tempo de entrega é maioritariamente composto pelo período de sincronização dos dois nós.

O cenário 3 foi constituído por 6 nós DTN parcialmente conectados. Este cenário foi projetado de forma a simular um ambiente marítimo composto por 6 nós, em que por exemplo, 5 nós são embarcações e o restante é o nó presente na costa, sendo que cada nó tem conetividade limitada, pois apenas consegue comunicar com um ou dois nós vizinhos. Num cenários marítimo destes, caso uma embarcação ou o nó da costa pretenda enviar um *e-mail* que tenha como destino um nó que não esteja imediatamente a seu alcance, esse e-mail terá de ser passado a outros barcos, que reencaminharão o e-mail na rede marítima até ao destino (Fig 7).

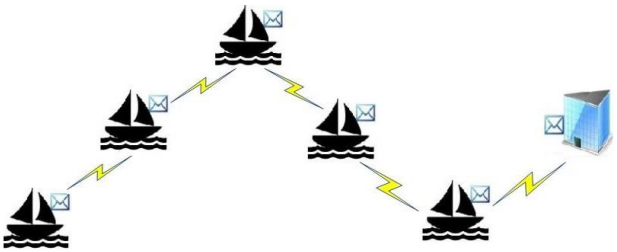


Fig 7. Cenário 3 - Seis nós DTN com conexão estabelecida.

No cenário 3 foram feitos 5 tipos de testes, cada um deles envolvendo um número de nós diferente. Os testes consistiram no envio de *e-mails* entre um número variável de nós DTN, ou seja, em alguns testes foram utilizados 2 nós, em outros testes 3 nós e assim sucessivamente até aos 6 nós, sendo que o objetivo dos testes foi avaliar o desempenho do serviço, nomeadamente a rapidez, quando o número de *hops* na entrega dos e-mails é superior a 1. Em cada conjunto de testes, excetuando os testes com apenas 2 nós, foram utilizadas mulas de transporte entre a origem e o destino, ou seja, nós DTN que apenas são responsáveis por fazer o transporte das *bundles*, não tendo qualquer papel a nível aplicacional. Neste cenário

foram enviados *e-mails* de texto, originando *bundles* de tamanho padrão.

B. Análise de desempenho no cenário 1

O estudo começou com a avaliação do desempenho do sistema num cenário onde estão dois nós DTN totalmente sincronizados.

Neste cenário, foram executadas duas séries de testes descritas no capítulo anterior. Em cada série de testes foram enviados manualmente 40 *e-mail*s entre os dois nós, sendo que na primeira série os *e-mail*s foram constituídos apenas por alguns bytes de texto, e na segunda série foram constituídos por texto e imagens. Os resultados obtidos na primeira série de testes estão ilustrados na Fig. 8.

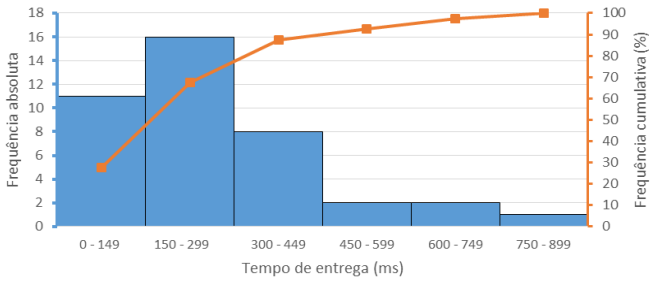


Fig 8. Tempo de Entrega de E-mails com 4 Kbytes.

Na Fig.8 é possível ver que a maior parte dos *e-mail*s demoraram menos de meio segundo a serem entregues, quando as respetivas *bundles* têm como tamanho 4 Kbytes. Nas 40 experiências realizadas verificou-se que existiu uma ligeira variação de valores nos resultados obtidos. Isto deve-se necessariamente à capacidade de processamento instantânea da máquina onde foi montada a *testbed*, aquando da realização dos testes.

O segundo conjunto de experiências realizadas no cenário 1, tiveram como resultados os valores ilustrados no gráfico da Fig. 9, onde é indicado um intervalo de confiança de 90%.

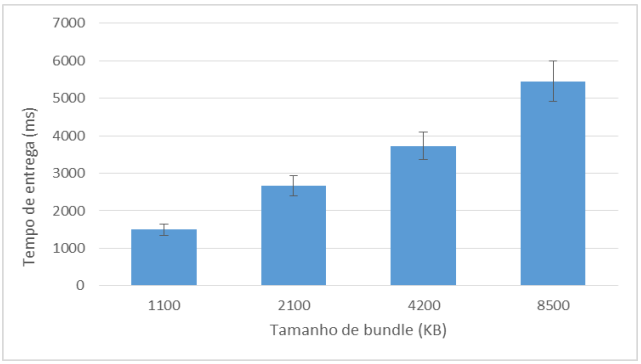


Fig 9. Tempo de Entrega de E-mails com Tamanho Variável.

A partir do gráfico anterior é possível visualizar que, apesar do aumento não ser totalmente proporcional, quanto maior for o tamanho do *e-mail* enviado, maior é o tempo despendido na sua entrega. Este facto deve-se essencialmente à quantidade de informação que é necessário processar em cada situação e naturalmente à capacidade de processamento do instrumento de trabalho utilizado nos testes. Tal como no conjunto de testes anterior, os tempos de transmissão das *bundles* entre os nós foram reduzidos, portanto, grande parte do tempo gasto na entrega dos *e-mail*s foi consumido pelas aplicações do sistema.

C. Análise de desempenho no cenário 2

O segundo cenário serviu para avaliar o desempenho do sistema, no que se refere ao tempo que dois nós DTN levam a se sincronizar e ao tempo que levam, após a sincronização, a trocar os *e-mail*s que têm em buffer. Neste cenário foram criados manualmente 40 *e-mail*s, originando respetivas *bundles* com 4 Kbytes tamanho. Os procedimentos tomados antes e durante a realização de cada teste estão especificados no capítulo anterior. Na Fig. 10 estão ilustrados os resultados obtidos.

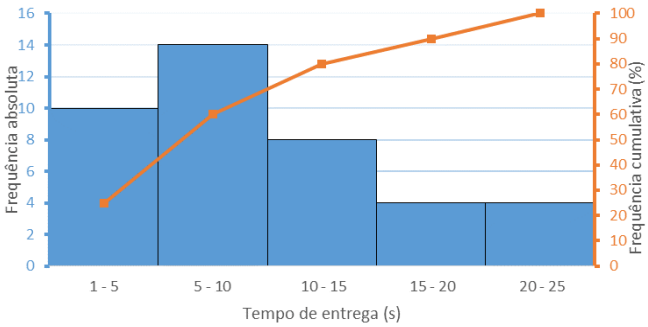


Fig 10. Tempo Total de Sincronização e Entrega de E-mails.

A partir do gráfico da Fig. 10 é possível verificar que nos 40 *e-mail*s enviados, existiu uma grande disparidade de valores nos tempos de entrega dos *e-mail*s, visto que 60% deles levaram menos de 10 segundos a serem entregues e 10% deles demoraram mais de 20 segundos. Isto deve-se essencialmente ao tempo gasto pelos nós na sua sincronização. Os agentes de descoberta presentes nos nós DTN estão configurados para enviar datagramas UDP a cada 5 segundos. Contudo a sincronização com um vizinho pode demorar mais algum tempo, até porque, geralmente a ligação DTN entre os nós apesar de estar criada e registada, não fica ativa por alguns segundos.

D. Análise de desempenho no cenário 3

O terceiro cenário foi elaborado com o objetivo de avaliar a capacidade do sistema, no que se refere à transmissão de *bundles* com a ajuda de nós intermédios. Para isso foram enviados 25 *e-mails* em cada contexto presente no cenário 3. Os resultados obtidos estão presentes graficamente na Fig. 11, com um intervalo de confiança de 90%.

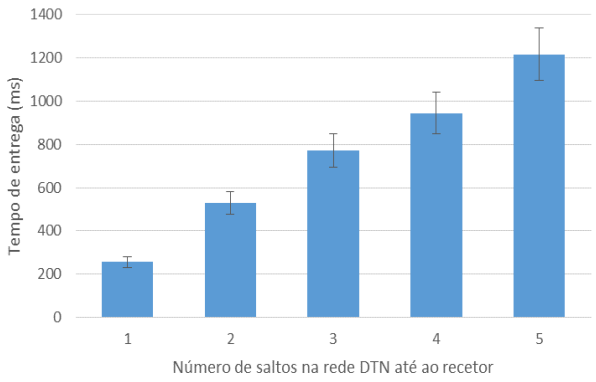


Fig 11. Tempo de Entrega de E-mails com Número de Saltos Variável.

O gráfico presente na Fig. 11 demonstra que o tempo de entrega dos e-mails foi praticamente proporcional ao número de saltos realizados na rede DTN até ao recetor, sendo que o atraso médio introduzido por cada salto foi 243,4 ms. Esta proporcionalidade acontece porque todos os nós estão configurados de forma semelhante e são constituídos pelas mesmas arquiteturas. Contudo, o facto de neste cenário terem sido utilizadas várias máquinas virtuais em simultâneo no mesmo computador, pode ter tido alguma influência nos resultados obtidos.

V. Conclusões e Trabalho Futuro

Neste artigo, foi exposta a implementação e avaliação de uma proposta baseada em DTN, que possibilita a utilização do serviço de *e-mail* no cenário marítimo. A proposta implementada foi avaliada a nível do desempenho do serviço em dois cenários. Contudo, mais importante do que os valores obtidos nas experiencias, é constatar a eficiência do sistema proposto ao executar o seu principal propósito, transmissão de *e-mail*s entre dois nós sem existir obrigatoriamente uma conetividade *extremo-a-extremo*.

O sistema foi totalmente implementado num computador sobre uma plataforma de virtualização. Este facto tem alguma preponderância nos valores obtidos a partir dos testes de desempenho, pois a capacidade de processamento do instrumento de trabalho, foi o fator que determinou a rapidez de execução na entrega dos *e-mail*s.

A validação da solução proposta neste artigo, indica que o serviço de *e-mail* que foi implementado em ambiente controlado, pode ser transportado para o cenário real, visto que o protocolo Bundle está perfeitamente adaptado às condições impostas pelo ambiente de comunicações marítimo e a parte aplicacional do sistema está totalmente funcional.

Agradecimentos

Este trabalho é financiado pela FCT – Fundação para a Ciência e a Tecnologia no âmbito do projeto UID/EEA/50014/2013.

Bibliografia

[1] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, e H. Weiss, “Delay-Tolerant Network Architecture”, *RFC4838*, 2007.

[2] F. Warthman, “Delay-Tolerant Networks (DTNs) A Tutorial v2.0”, Disponivel:http://ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.04.pdf, 2012.

[3] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, e P. McDonald, “When TCP Breaks: Delay- and Disruption-Tolerant *Networking,”IEEE Internet Computing*, pág. 72-78, Julho-Agosto 2006.

[4] E. Husni, A. Wibowo, "*E-mail* system for Delay Tolerant Network", *2012 International Conference on System Engineering and Technology* (ICSET), Bandung, Indonésia, 2012.

[5] E. Husni, A. Sumarmo, "Delay Tolerant Network utilizing train for news portal and email services", *2010 International Conference on Information and Communication Technology for the Muslim World* (ICT4M), Jakarta, Indonésia, 2010.

[6] Projeto N4C, http://www.n4c.eu/.

[7] A. Azfar, J. Jiang, L. Shan, M.J.P. Marval, R. Yanggratoke, e S. Ahmed, ByteWalla: Delay Tolerant Networks on Android phones, CSD Labs, The Royal Institute of Technology, Stockholm, Final Report, 2010.

[8] H.-M. Lin, Y. Ge, A.-C. Pang e J. Pathmasuntharam, “Performance Study on Delay Tolerant Networks in Maritime Communication Environments”, *OCEANS 2010 IEEE*, Sydney, Austrália, 2010.

[9] J.A. Dias, J.N Isento, V.N.G.J. Soares, F. Farahmand e J.J.P.C. Rodrigues, "Testbed-based performance evaluation of routing protocols for vehicular delay-tolerant networks", *2011 IEEE GLOBECOM Workshops* (GC Wkshps), Houston, E.U.A, 2011.

Mecanismo de Deteção de Consumos Anómalos em Redes Energéticas Inteligentes

Hugo C. Costa[1], Cristiano Cabrita[1,2], Jânio Monteiro[1,3], Jorge Semião[1,3]
(1) Instituto Superior de Engenharia, Universidade do Algarve, Faro, Portugal
(2) IDMEC-IST, Lisboa, Portugal
(3) INESC-ID, Lisboa, Portugal

{hugomiguelccosta@hotmail.com; ccabrita@ualg.pt; jmmontei@ualg.pt; jsemiao@ualg.pt}

***Abstract*— As redes de sensores permitem hoje medir, comunicar, armazenar, monitorizar e controlar uma gama alargada de variáveis e dispositivos. A aplicação dessas redes às redes energéticas inteligentes permite, entre outras funcionalidades, a monitorização de consumos de equipamentos e instalações. No entanto, hoje em dia, na maioria dos casos em que esses consumos já são monitorizados, a deteção de anormalidades faz-se com recurso à intervenção humana, através de técnicos que, com maior ou menor experiência, analisam as curvas de consumo para perceber onde e quando os respetivos valores ficaram acima ou abaixo de valores ditos normais. Com a complexidade de uma estimação deste nível a crescer com o número de variáveis envolvidas, tais como dia da semana e hora em que o consumo se verifica, temperatura exterior, radiação solar, ou mesmo níveis de ocupação do edifício, importa implementar métodos automáticos de deteção de consumos anómalos. Para alcançar este objetivo, comparamos neste artigo dois métodos, um recorrendo a redes Neuronais e outro recorrendo a uma rede Bayesiana Dinâmica, que permitem estimar o consumo de uma instalação em função do seu histórico, gerando alarmes sempre que os valores reais ficam acima ou abaixo dos intervalos normais.**

***Index Terms*— Previsão de Consumo, Redes Energéticas Inteligentes, Redes Bayesianas, Redes Neuronais.**

I. Introdução

As redes de sensores são hoje utilizadas num espectro alargado de aplicações industriais e habitacionais, que resultam da disponibilidade, a custos reduzidos, de um conjunto de dispositivos capazes de efetuar a recolha de dados, de os processar e de comunicá-los via redes IP para servidores locais ou na Internet. Globalmente esses dispositivos constituem o que é hoje chamado de Internet das Coisas (*Internet of Things*).

Uma das aplicações das redes de sensores que tem ganho relevância nos últimos anos encontra-se nas redes energéticas inteligentes. Os objetivos das mesmas passam, por um lado, pela monitorização e controlo de dispositivos, por forma a promover o equilíbrio entre a produção e o consumo (veja-se por exemplo [1]), mas também pela implementação de medidas ativas de eficiência energética.

No âmbito da Eficiência Energética de edifícios, para além das medidas de controlo de equipamentos, importa desenvolver métodos que, a partir dos dados obtidos de vários sensores, consigam perceber se os consumos da instalação estão dentro doa padrões normais de funcionamento, gerando notificações caso tal não se verifique. Ora, se por um lado a monitorização de consumos é já uma realidade em algumas empresas nacionais e internacionais, é também verdade que a deteção de consumos anómalos continua a ser feita por pessoal técnico que, com maior ou menor experiência, analisa os gráficos de consumo e os interpreta em face do que foram os consumos históricos.

Neste contexto, importa encontrar métodos automáticos de análise dos dados de consumo que permitam efetuar esse mesmo procedimento permanentemente, gerando alertas de consumo anormal de uma forma rápida e evitando a sua tardia deteção e consequente ação. Para que se alcance esse objetivo, vários desafios têm que ser ultrapassados, como a seguir se explica. Um desses desafios consiste no facto dos consumos globais dependerem de uma série de fatores como o estado de ocupação do edifício, da respetiva eficiência energética e em particular dos seus equipamentos, dos hábitos de consumo dos utilizadores, da temperatura exterior, da radiação solar, ou mesmo do dia da semana em que nos encontramos. Sendo esses fatores determinantes no consumo, qualquer mecanismo de deteção de consumos anómalos deve considerá-los antes de gerar um alarme de consumo excessivo, ou inferior ao que é expectável. A solução pode passar pelas Redes Energéticas Inteligentes, que ao integrarem várias tecnologias envolvendo as Redes de Sensores, Internet das Coisas e as Tecnologias de Informação e Comunicação, combina-as com algoritmos de apoio à decisão que suportem medidas ativas de eficiência energética. O objetivo desses algoritmos é o de, em função dos consumos passados, prever o nível de consumo normal da instalação, gerando alarmes sempre que esse consumo fique acima ou abaixo dos níveis considerados normais. Para efetuar esta previsão podem ser usados métodos estatísticos ou de otimização que, correlacionando o consumo energético com os fatores determinantes para esse consumo, consigam perceber quando o consumo real se afastou do previsto.

Para análise desses dados pode recorrer-se a modelos de regressão [2] ou modelos de inteligência artificial, incluindo redes neuronais [3][4], *support vector machines* [5], redes Bayesianas [6] [7], entre outros.

Uma das abordagens que emprega redes neuronais para a deteção de anomalias é a que utiliza uma rede treinada em paralelo com o sistema. A saída da rede é então comparada com a do sistema físico, identificando as anomalias [8]. Neste

âmbito, foi dedicada especial atenção à deteção de anomalias em unidades de tratamento de ar nos edifícios, usando metodologias estatísticas como *Principal Component Analysis* [9], classificadores difusos [10][11] e redes do tipo perceptrão determinístico recorrente [12].

Por outro lado, as redes Bayesianas Dinâmicas têm vindo a ganhar importância como método de estimação. Refira-se por exemplo o trabalho desenvolvido em [13] na previsão de consumo agregado em zonas residenciais.

Assim, no âmbito do presente artigo, pretende-se analisar e comparar modelos para previsão de consumo, com base em modelos auto-regressivos com entradas exógenas, de redes Bayesianas Dinâmicas e de redes Neuronais. Em qualquer dos modelos assume-se que através de uma rede de sensores medimos três grandezas: temperatura exterior, radiação solar e energia consumida (em kWh). Com base nessas grandezas, pretende-se que os consumos anómalos sejam determinados pela diferença entre os valores de estimação obtidos pelos modelos e as medições reais. Mas, para que tal aconteça, os resultados dos modelos propostos têm que demostrar uma elevada correlação com os dados de consumo reais.

O resto do artigo tem a seguinte estrutura. No capítulo II descrevem-se os modelos implementados. Os resultados dos modelos são descritos no Capítulo III e no Capítulo IV conclui-se o artigo, apresentando os resultados da comparação entre os diversos modelos.

II. Modelos Implementados

Em seguida apresentam-se os três modelos utilizados neste estudo.

A. Rede Bayesiana Dinâmica

As redes Bayesianas convencionais representam as probabilidades de uma forma estática, não variante no tempo. Para os modelos em que as probabilidades variam, têm que se utilizar as Redes Bayesianas Dinâmicas (RBD).

As RBD foram apresentadas pela primeira vez por Dean e Kanazawa em 1988 [7], como uma extensão das redes Bayesianas [6]. Neste sentido, uma RBD permite utilizar um modelo gráfico probabilístico para descrever o nível de incerteza numa variedade de aplicações. As RBD permitem também a análise de fenómenos complexos e o apoio à decisão em situações altamente variáveis e interligadas [7] ou quando os dados são ambíguos.

Nas RBD, o momento temporal das variáveis é apresentado por um conjunto de variáveis aleatórias $X^t = X_1^t ... X_n^t$. Se o sistema depender apenas do estado imediatamente anterior, ele é denominado de cadeia de Markov de 1ª ordem, com a sua distribuição dada por [7]:

$$P(X^t|X^{t-1}) = \prod_{i-1}^{n} P(X_i^t|X_{pai}(X_i^t)) \quad (1)$$

onde $X_{pai}(X_i^t)$ representa o nó antecessor de X_i^t. Neste artigo iremos utilizar este tipo de RBD, representado por uma cadeia de Markov de 1ª ordem, apresentada na Fig. 1. Os nós mais escuros indicam os dados X_i^t e os restantes os estimadores.

A deteção de consumo anómalo faz-se pela comparação entre o consumo efetivamente medido (no período de um quarto de hora) com aquele que é obtido pelo modelo (i.e. o estimado). O consumo estimado é obtido a partir de dois estimadores, identificados na Fig. 1 como Estimadores de Consumo A e B.

O Estimador A utiliza como dados de entrada o dia da semana e hora atuais, não entrando em linha de conta com a temperatura exterior e com a radiação solar. No caso do Estimador B, para prever o consumo, utilizam-se os dados: do dia da semana, da hora do dia, da temperatura exterior e da radiação solar. A previsão de consumo faz-se com base na distribuição de probabilidades de consumo verificadas até então, para as mesmas condições de entrada (dia da semana e hora do dia; temperatura exterior e radiação solar).

O resultado apresentado pelo Estimador Final dá preferência ao estimador B, pois este considera mais variáveis. No entanto, e uma vez que para efetuar a estimação de consumo se utilizam dados do passado, nem sempre o estimador B tem todos os dados que necessita. Sempre que tal acontece recorre-se ao Estimador A que, sendo menos esparso, consegue fazer previsões ao final de uma semana.

Segue-se a descrição das variáveis utilizadas no modelo:

- Dia da Semana: um valor inteiro a variar entre 1 e 7 que traduz o dia da semana em que nos encontramos, onde o valor 1 corresponde ao domingo.
- Hora do Dia: um valor inteiro entre 1 e 24 que traduz a hora do dia em que nos encontramos, onde o valor 1 corresponde ao período entre as 0 e a 1 hora.
- Temperatura: um valor discreto que reflete a temperatura exterior, fragmentado em intervalos de 5ºC, onde o valor 1 representa o intervalo entre -10ºC a -5ºC, o valor 2 representa o intervalo entre -5ºC a 0ºC, etc.
- Radiação: um valor discreto que reflete a radiação solar, fragmentado em intervalos de 500 W/m^2, em que o valor 1 representa a radiação entre 0 e 500 W/m^2, etc.

Neste caso, para que o histograma de consumos traduza uma distribuição de probabilidades normal, o processo de

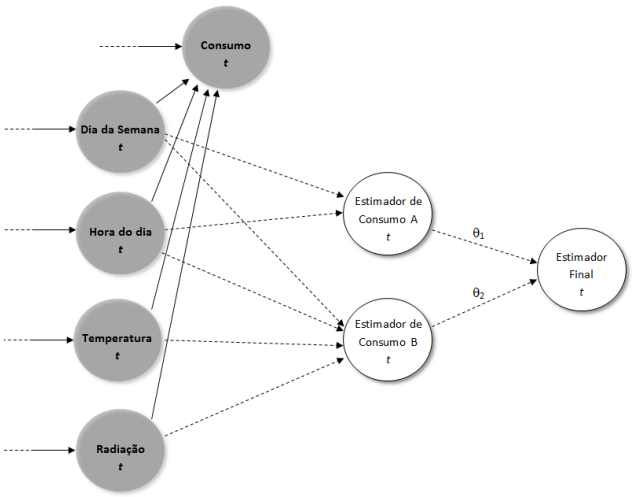


Fig. 1. Esquema da Rede Bayesiana Dinâmica utilizada neste artigo.

discretização de consumos é acompanhado por uma conversão de escala, traduzido pela equação (2).

$$Índice\ de\ Consumo = 1 + round(Pconsumida^{1/3}) \quad (2)$$

A aproximação do histograma de consumos a uma distribuição normal permite utilizar a média aritmética dos valores medidos como método de *maximum likelihood estimation*. A equação (2) introduz também um processo de discretização não linear, que permite dar uma maior precisão a valores baixos de consumo, traduzindo-se por isso no final em menores erros percentuais quando as grandezas medidas são baixas.

De acordo com as características da distribuição normal, 95,4% dos valores medidos devem estar a uma distância de dois desvios padrão (σ) da média. Assim, dado um nível de consumo medido, produz-se a geração de um alarme sempre que o mesmo esteja a mais que δ vezes o desvio padrão dos consumos anteriores, em que δ pode ser ajustado ao longo do funcionamento do equipamento.

Ao fim de uma semana de operação, o Estimador de Consumo A começa a reportar dados de estimação, pois tem em consideração apenas os mesmos dias da semana e o mesmo intervalo horário da(s) semana(s) anterior(es). Em relação ao estimador B, este só efetua estimações quando verificou no passado as mesmas condições de radiação e temperatura exterior, de acordo com os intervalos acima descritos.

B. Modelo auto-regressivo com entradas exógenas

Como alternativa ao modelo sustentado pela rede Bayesiana, é discutida a aplicação de um modelo que também se insere na categoria de sistemas inteligentes, dada a sua capacidade em resolver problemas complexos, quer aplicando os princípios relacionados com a capacidade de raciocínio humano, quer aplicando os fundamentos de ordem biológica que estão na base do seu funcionamento.

Para obter uma eficiência aceitável, estes modelos dependem essencialmente da seleção das variáveis de entrada e da representação do problema que os dados recolhidos constituem para a fase de aprendizagem (fase a partir da qual se determina o modelo final). Os dados deverão então ser em quantidade suficiente para que a capacidade de generalização não seja afetada com a aprendizagem. Por conseguinte, é habitual separar os dados em dois ou três conjuntos. Ao primeiro conjunto designa-se de treino e é usado para estimar os parâmetros do modelo, contemplando a maior parte dos dados. Um segundo conjunto é usado para validar o modelo durante a fase de aprendizagem (conjunto de validação), ao passo que um terceiro conjunto é útil para, por um lado testar a qualidade do modelo, e, por outro, auxiliar na seleção do modelo a manter no final.

Acrescentar instantes passados das variáveis de entrada traduz-se num maior número de variáveis de entrada no modelo e, consequentemente, num aumento da sua estrutura, tempo e carga computacional. Neste sentido, foi aplicado um modelo de regressão linear Auto-Regressivo com entradas eXógenas (ARX) com o objetivo de aferir da importância dos instantes passados no modelo de estimação do consumo energético.

O consumo estimado pelo modelo ARX é descrito pela equação seguinte, para *n* entradas exógenas:

$$p(t) = \sum_{i=1}^{n_p} -a_i p(t - i\times\tau) + \sum_{j=1}^{n}\sum_{i=1}^{n_p} b_{j\times(n_p-1)+i} x_j(t-(i-1)\times\tau) \quad (3)$$

onde n_p representa o número de parâmetros total para a componente auto-regressiva (definidos por a_i), τ é o atraso (em número de amostras) entre as entradas e a saída e, $b_{j\times(n_p-1)+i}$ denota o parâmetro linear de índice $j\times(n_p-1)+i$ associado à entrada exógena *v*. Note-se que o número total de parâmetros a estimar é definido por $n_p\times(n+1)-1$.

Os parâmetros são estimados a partir da solução dos mínimos quadrados, aplicando:

$$\hat{\theta} = \left(A^T A\right)^{-1} A^T y \quad (4)$$

onde

$$\theta = \left[a_1 a_2 \ \ldots\ a_{n_p} b_1 b_2 \ldots b_{n_p\times n}\right]^T$$

sendo a matriz de regressão *A* descrita por:

$$A = \begin{bmatrix} p_1(t-\tau)\ p_1(t-2\tau)\ \ldots p_1(t-n_p\tau)\ D_{1,1}\ D_{1,2}\ \ldots D_{1,n} \\ p_2(t-\tau)\ p_2(t-2\tau)\ \ldots p_2(t-n_p\tau)\ D_{2,1}\ D_{2,2}\ \ldots D_{2,n} \\ \ldots \\ p_N(t-\tau)\ p_N(t-2\tau)\ \ldots p_N(t-n_p\tau)\ D_{N,1}\ D_{N,2}\ \ldots D_{N,n} \end{bmatrix} \quad (5)$$

com $D_{i,j} = \left[x_{i,j}(t)\ x_{i,j}(t-\tau)\ \ldots x_{i,j}(t-n_p\tau)\right]$, considerando um total de *N* dados.

De salientar que os dados de entrada dos modelos inteligentes da subsecção seguinte, usam a definição para a matriz de regressão anterior para definir o conjunto de dados entregue aos mesmos. Por conseguinte, quer n_p quer τ determinam o quanto o modelo depende dos instantes de tempo anteriores das variáveis usadas como entradas.

As metodologias de aprendizagem dos sistemas aqui referidos inserem-se na categoria de aprendizagem supervisionada e, como tal, é habitual definir um critério de treino que determina a seleção do modelo. Tipicamente, o critério usado é a soma dos erros quadráticos (SSE):

$$SSE = \sum_{i=1}^{N} \left(\hat{y} - y\right)^2 \quad (6)$$

Não obstante, e por se verificar que o critério SSE não é propriamente adequado para avaliar a qualidade do modelo, à luz do objetivo de deteção de anomalias, também o erro relativo em percentagem (MRE%) foi tomado em conta assim como a média da soma dos erros quadráticos (MSE):

$$MRE\% = \frac{100}{N}\sum_{i=1}^{N}\frac{\left|\hat{y} - y\right|}{y} \quad (7)$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y} - y\right)^2 \quad (8)$$

C. Modelo baseado em Redes Neuronais

Para além dos dois modelos anteriores em seguida utiliza-se um modelo baseado em Redes Neuronais. A rede neuronal empregue é o caso particular de uma rede multi-camada (MLP) com apenas uma saída, a qual corresponde ao consumo estimado. Uma MLP é uma rede neuronal artificial constituída por um conjunto de neurónios distribuídos por várias camadas, todos eles interligados por um conjunto de ligações ou pesos (vide Fig. 2). O sinal que alimenta esta arquitetura de redes propaga-se entre a camada de entrada e a camada de saída, atravessando a camada escondida. Esta última pode conter um número indeterminado de camadas, embora seja habitual o uso de até duas camadas. Às ligações entre os diversos neurónios designam-se pesos e a capacidade de eficiência destas redes depende dos valores que são atribuídos aos pesos assim como à quantidade de pesos por cada camada (por outras palavras, a topologia usada). Para além dos pesos, a cada neurónio está associado um viés.

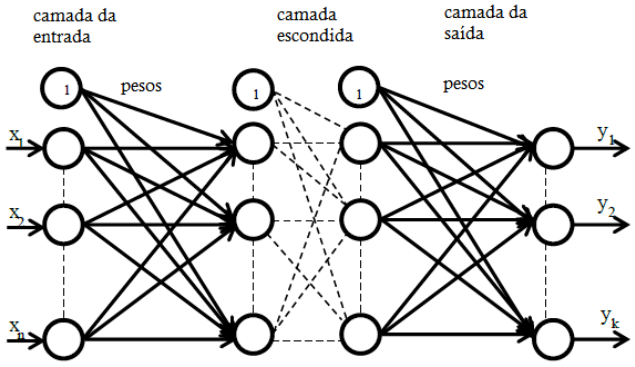


Fig. 2. Esquema de Rede Neuronal Perceptrão Multi-Camadas (MLP)

O esquema da rede neuronal MLP genérica com *k* saídas é mostrado na figura seguinte. A saída *r* desta rede, considerando apenas uma camada escondida, é uma combinação linear entre os pesos da camada da saída, ie.:

$$y_r(X) = \sum_{i=1}^{n_h} w^{(2)}{}_{i,r} I_r(X) + w^{(2)}{}_{0,r} \qquad (9)$$

onde

$$I_j(X) = \frac{1}{1+e^{-\sum_{p=1}^{n} w^{(1)}{}_{p,j} x_p + w^{(1)}{}_{0,j}}}, X = x_1, x_2, \ldots x_n . \qquad (10)$$

Na equação (9) y_r é a saída para o neurónio *r*, $w^{(2)}{}_{i,j}$ é o peso da camada da saída entre o neurónio da camada escondida *i* e o neurónio da camada da saída *j* e x_p é o valor para a entrada *p*. A relação não linear entre a saída e as entradas, ilustrada pela equação (10), mostra que o problema, para encontrar os valores dos parâmetros da rede, é um problema de otimização não linear. O método mais comum estima os parâmetros por meio da retro-propagação do erro na saída em direção à camada da entrada [14]. No entanto este método exibe uma fraca taxa de convergência, pelo que no presente estudo se optou pelo método de Levenberg-Marquardt [15], que exibe uma taxa de convergência bastante mais rápida, para além de melhor explorar o espaço de procura (em consequência de ser um método de 2ª ordem). Esta metodologia foi implementada em Matlab, usando-se por isso, as funções correspondentes à implementação das MLPs.

Como foi referido anteriormente, para além de especificar que variáveis de entrada são usadas, é necessário definir a topologia, indicando a quantidade de neurónios por camada. A fase de aprendizagem só se inicia depois de se fixarem os valores iniciais para os parâmetros da rede (pesos e viés). Para esse efeito, é utilizado o algoritmo de inicialização de Nguyen-Widrow [16], que devolve diferentes soluções de cada vez que é usado para a mesma topologia, permitindo obter diferentes desempenhos com a mesma rede e, desse modo, convergindo para uma solução mais próxima da solução global. Por esse motivo é habitual executar a mesma topologia diversas vezes, para selecionar a rede que apresente o melhor critério de avaliação.

Neste artigo foram consideradas 10 execuções por cada topologia testada e, não pretendendo proceder a uma procura exaustiva, apenas diferentes topologias foram adotadas. Os dados foram divididos em dois conjuntos, dos quais o primeiro foi usado para treinar o modelo e compreende as datas entre 8/Abr./2014 e 31/Out./2014. De modo a empregar a estratégia *early-stopping* (caraterizada pela paragem assim que ocorra um aumento no critério de avaliação dos dados de validação) estes dados foram subsequentemente divididos em dois conjuntos na proporção {85%, 15%}. O conjunto de dados correspondente a 85% (dados de treino) são usados para estimar os parâmetros do modelo, fazendo-se a validação com os restantes 15%. O terceiro conjunto refere-se aos dados entre 01/Nov./2014 e 31/Jan./2015, e apenas são usados depois de escolhido o modelo final para determinada topologia. Não obstante a existência do conjunto de validação, foi estabelecido um número máximo de iterações por execução de 200.

III. Avaliação de Resultados

Para efetuar a avaliação dos resultados utilizaram-se dados de medição de consumo do edifício da Biblioteca Municipal de Olhão, obtidos no período compreendido entre 8 de abril de 2014 e 31 de Janeiro de 2015, intercalados por períodos de 15 minutos. Em seguida descrevemos cada um desses resultados para os modelos apresentados anteriormente.

A Tabela I apresenta os resultados obtidos para o modelo da rede Bayesiana Dinâmica, apresentada na Fig. 1.

Tabela I Resultados obtidos para a Rede Bayesiana Dinâmica

	Dados de 8/Abr/2014 a 31/Out/2014		Dados de 01/Nov./2014 a 31/Jan./ 2015	
Estimador de Consumo	**MSE (Wh)**	**MRE %**	**MSE (Wh)**	**MRE%**
A	$7{,}8\text{x}10^{10}$	27,6	$4{,}5\text{x}10^{6}$	32,7
B	$5{,}8\text{x}10^{10}$	28,3	$2{,}0\text{x}10^{6}$	21,9
Final	$8{,}1\text{x}10^{10}$	27,8	$2{,}5\text{x}10^{6}$	22,7

Os resultados obtidos mostram que este modelo consegue estimar com um erro final de 22,7% o consumo da instalação. Quando comparamos os Estimadores de Consumo A e B, verificamos que o B consegue obter um erro menor (21,9%).

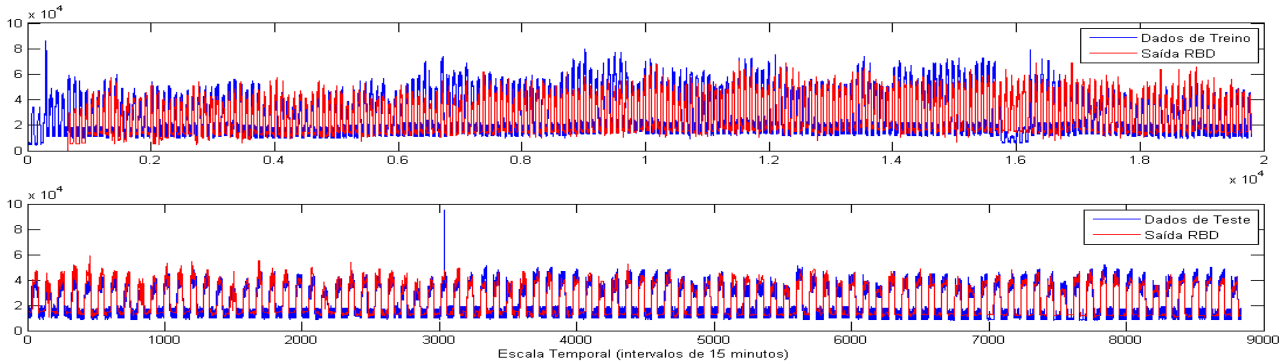


Fig. 3. Comparação entre consumo real e estimado, utilizando a rede Bayesiana Dinâmica, em que o gráfico superior corresponde à fase anterior a novembro de 2014 e a parte inferior à fase posterior.

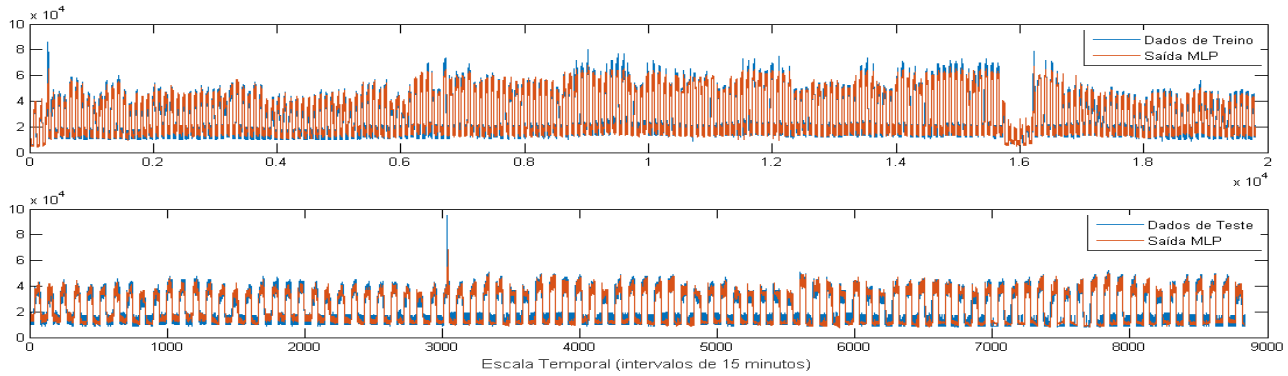


Fig. 4. Comparação entre consumo real e estimado pelo melhor modelo MLP, com topologia [8 8 1], em que o gráfico superior corresponde à fase de treino e a parte inferior à fase de funcionamento.

No entanto, apesar de ser mais preciso, para efetuar previsões requer que condições semelhantes de temperatura exterior e radiação se tenham verificado no passado, o que nem sempre acontece. Já o estimador de consumo A, não sendo tão preciso, consegue sempre devolver estimativas após uma semana de monitorização de consumos.

A fim de determinar o melhor modelo ARX, compararam-se diferentes modelos ARX resultantes da geração de todas as combinações para um intervalo de valores de n_p e τ. O valor de n_p variou no intervalo [1,80] enquanto o atraso τ ficou compreendido no intervalo [1,5]. A TABELA II apresenta os resultados obtidos com o modelo ARX, indicando na primeira linha o modelo com menor valor de MSE para os dados de validação (dados entre 12 e 31 de Outubro) e, na segunda linha o modelo com menor MSE para os dados de treino (dados entre Abril e 11 Outubro).

TABELA II ESPECIFICAÇÕES DE DESEMPENHO DO MODELO ARX

Modelo	**Dados de 8/Abr/2014 a 11/Out/2014**		**Dados de 12/Out/2014 a 31/Out/2014**		**Dados de 01/Nov./2014 a 31/Jan./ 2015**	
Parâmetros { n_p,τ}	**MSE (Wh)**	**MRE %**	**MSE (Wh)**	**MRE %**	**MSE (Wh)**	**MRE %**
{50,2} (#par=149)	2,1x10^7	14,8	1,6x10^7	16,1	3,7x10^7	23,3
{79,2} (#par=236)	2,0x10^7	14,7	1,6x10^7	16,1	3,7x10^7	23,0

Como se pode observar, as soluções obtidas nos dois casos são distintas na quantidade de parâmetros que usam, mas idênticas em termos de especificações de avaliação. A diferença relevante está na complexidade dos modelos e nos instantes passados que são necessários guardar em memória para implementar tais modelos. Note-se que o pior caso obriga a usar 79 parâmetros por entrada e um atraso de 30 minutos ($\tau = 2$) entre amostras. Em relação ao valor de MRE% dos dados de teste, de sublinhar que é um valor idêntico ao obtido pelos melhores modelos das restantes abordagens em discussão.

Apesar dos resultados obtidos pelo modelo ARX indicarem a importância dos instantes passados na estimação do consumo, a vantagem de usar um modelo regressivo não linear, como é o caso de uma rede neuronal, está no facto de se poder reduzir significativamente a quantidade de variáveis de entrada, mantendo o desempenho a níveis idênticos.

Na TABELA III são listados os resultados obtidos para 3 diferentes topologias de rede MLP e os valores para os critérios de desempenho adotados neste trabalho. Tendo em conta os resultados obtidos pelo modelo ARX, testaram-se dois conjuntos de entradas, ambos com $\tau = 1$. O primeiro (5 entradas) recorre ao instante de tempo passado mais recente para a radiação solar, temperatura do ar e consumo, assim como instantes atuais para a radiação e temperatura. O segundo (2 entradas) admite apenas a disponibilidade dos dados da radiação e temperatura do ar atuais.

TABELA III ESPECIFICAÇÕES DE DESEMPENHO DA REDE NEURONAL COM MELHOR CRITÉRIO DE TREINO

Número de variáveis de entrada	Modelo	Dados de 8/Abr/2014 a 31/Out/2014		Dados de 01/Nov./2014 a 31/Jan./ 2015	
	Topologia	MSE (Wh)	MRE %	MSE (Wh)	MRE %
5	**[4 4 1] (#par=49)**	$4{,}2x10^7$	21,5	$4{,}1x10^7$	21,7
5	**[4 6 1] #par=61**	$4{,}6x10^7$	26,5	$4{,}2x10^7$	26,0
5	**[8 8 1] (#par=129)**	$3{,}7x10^7$	21,5	$4{,}2x10^7$	**20,6**
2	**[4 4 1] (#par=37)**	$9{,}1x10^7$	30,7	$1{,}2x10^8$	34,8
2	**[4 6 1] (#par=49)**	$8{,}5x10^7$	29,5	$8{,}6x10^7$	30,3
2	**[8 8 1] (#par=105)**	$8{,}2x10^7$	28,7	$8{,}9x10^7$	27,8

Na TABELA III é possível confirmar que, com apenas 5 variáveis de entrada (1 instante de tempo passado para a radiação e temperatura), se obtém resultados de erro de estimação do consumo ligeiramente melhores, embora o valor do critério de aprendizagem MSE seja ligeiramente maior. Os resultados indicam que usando a rede MLP e a tipologia com oito neurónios em ambas camadas escondidas, é expectável obter erros relativos da ordem dos 20%.

Nas figuras 3 e 4 são apresentados as comparações entre o consumo real e o consumo estimado, respetivamente para a rede Bayesiana Dinâmica e para o modelo MLP com tipologia [8 8 1] e 5 entradas. Em relação à Fig. 4, pode observar-se que o comportamento exibido pela rede MLP é semelhante, quer para os dados de treino, quer para os dados de teste.

As maiores diferenças entre os modelos e correspondentes resultados apresentados nas figuras 3 e 4, residem no facto da rede Bayesiana Dinâmica estimar o consumo atual com base no histórico de consumo de há mais de uma semana, enquanto que no caso do modelo MLP aqui considerado se entrarem em linha de conta dados de consumo ocorridos 15 minutos antes.

IV. CONCLUSÕES

Os resultados obtidos evidenciam que no âmbito dos sistemas inteligentes o modelo de desempenho da rede neuronal consegue estimar com 20,6% de erro o consumo da instalação utilizando a topologia [8 8 1]. Já no caso da rede Bayesiana Dinâmica alcançou-se um erro um pouco superior a 22,7%. Estes resultados comparam com o desvio padrão do consumo diário, que se verificou ser de 31.4% face à média de consumo. Considera-se por isso que a capacidade de previsão dos modelos aqui descritos pode ser utilizada com vantagem na deteção de consumos anómalos.

Em termos de trabalho futuro, a partir dos resultados aqui obtidos, pretende-se desenvolver mecanismos de alerta de consumos anómalos, que serão utilizados para determinar a sensibilidade do modelo a desvios entre estimativa e consumo efetivo. A introdução no modelo de informação sobre a ocupação do edifício é também um dos objetivos futuros.

AGRADECIMENTOS

Os autores gostariam de agradecer à Agência Regional de Energia e Ambiente do Algarve, pelos dados disponibilizados, no âmbito do projeto PV-NET Metering, co-financiado pelo FEDER através do programa de cooperação transnacional MED. Os autores agradecem também a análise dos revisores, fator determinante na melhoria da qualidade deste trabalho.

REFERÊNCIAS

[1] Jânio Monteiro, Jorge Eduardo, Pedro J.S. Cardoso, Jorge Semião, "A Distributed Load Scheduling Mechanism for Micro Grids", 2014 IEEE International Conference on, 3-6 Nov. 2014.

[2] F. Lei, P. Hu, A baseline model for office building energy consumption in hot summer and cold winter region, in: International Conference on Management and Service Science (MASS'09), September 20–22, Beijing, China, 2009.

[3] J.F. Kreider, D.E. Claridge, P. Curtiss, R. Dodier, J.S. Haberl, M. Krarti, Building energy use prediction and system identification using recurrent neural networks, Journal of Solar Energy Engineering 117 (3) (1995) 161–166.

[4] B.B. Ekici, U.T. Aksoy, Prediction of building energy consumption by using artificial neural networks, Advances in Engineering Software 40 (5) (2009)356–362.

[5] B. Dong, C. Cao, S.E. Lee, Applying support vector machines to predict building energy consumption in tropical region, Energy and Buildings 37 (5) (2005)545–553.

[6] J. Pearl, Probabilistic reasoning in intelligent systems: networks of plausible inference: Morgan Kaufmann Publishers Inc., 1988.

[7] T. Dean and K. Kanazawa, "Probabilistic temporal reasoning," 1988.

[8] Hush, D. R., Abdallah, C., Heileman, G. L., Docampo, D.,"Neural networks in fault detection: a case study", Proceedings of the American Control Conference, July 1997

[9] Wang, S.W., and F. Xiao. AHU sensor fault diagnosis using principal component analysis method. Energy and Buildings (36): 147–160, 2004.

[10] Parvaresh, A., et al. Fault Detection and Diagnosis in HVAC System Based on Soft Computing Approach.

[11] Du, Juan, Meng Joo Er, and Leszek Rutkowski. Fault diagnosis of an air-handling unit system using a dynamic fuzzy-neural approach.

[12] Magoulès, Frédéric, Hai-xiang Zhao, and David Elizondo. Development of an RDP neural network for building energy consumption fault detection and diagnosis. Energy and Buildings, 62 (2013): 133-138.

[13] Vlachopoulou, M.; Chin, G.; Fuller, J.; Shuai Lu, "Aggregated residential load modeling using dynamic Bayesian networks," Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on , pp.818,823, 3-6 Nov. 2014.

[14] Z. Zainuddin, N. Mahat, and Y. Abu Hassan, Improving the Convergence of the Backpropagation Algorithm Using Local Adaptive Techniques, World Academy of Science, Engineering and Technology 1, 2005.

[15] K. Levenberg, A Method for the Solution of Certain Problems in Least Squares, Quarterly Applied Mathematics, (2) 164-8, 1944.

[16] D. Neguyen and B. Widrow, The truck backer-upper: an example of self-learning in neural networks. In Proceedings of the international Joint Conference On Neural Networks, pages II-357-363. IEEE, June 1989

Localização de Precisão Usando Redes Sem Fios Enterradas

José Venâncio, José Oliveira*, Rui Campos*, Jorge Mamede'
Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto
1080490@isep.ipp.pt
*INESC TEC e Faculdade de Engenharia, Universidade do Porto, FEUP
jcpo@inesctec.pt
rcampos@inesctec.pt
'INESC TEC e Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto
JBM@isep.ipp.pt

***Resumo*—As Redes Sem Fios Enterradas são formadas por nós que comunicam entre si através de ligações sem fios e têm como meio de propagação o solo. Os sistemas de localização mais utilizados atualmente têm desvantagens ao nível de precisão e custo.**

Neste artigo é proposta uma solução de localização de precisão que recorre à utilização de redes sem fios enterradas e um algoritmo de posicionamento baseados em Wi-Fi. O objetivo é estimar a localização de objetos, utilizando dispositivos de baixo custo. Os resultados experimentais obtidos demonstram que o erro de localização é inferior a 0,40 m, e que esta solução é viável para, por exemplo, localizar jogadores num campo de futebol ou localizar um objeto num campo agrícola.

Palavras-Chave—Wireless Underground Networks (WUN); Posicionamento; Wi-Fi; Search Nearest Neighbor

I. INTRODUÇÃO

As redes sem fios enterradas (*Wireless Underground Networks* –WUN) são formadas por nós que estão tipicamente localizados debaixo de terra e que utilizam tecnologias sem fios para comunicar com outros nós enterrados ou à superfície. Numa WUN há a particularidade de o meio de comunicação ser o solo, podendo esta comunicação ser *underground-to-underground* (U2U), quando os dois nós estão enterrados e o único meio de propagação é o solo, *underground-to-aboveground* (U2A), quando a informação é enviada a partir do nó enterrado para o nó que está à superfície e *aboveground-to-underground* (A2U), quando o transmissor está à superfície e o recetor está enterrado. Uma vez que o meio de propagação é o solo, a propagação rádio tem características distintas da propagação no ar, sobretudo no que diz respeito à atenuação da potência de sinal em função da distância. As implementações existentes de WUN são tipicamente baseadas em redes de sensores sem fios[1]. As WUN têm diversas aplicações [2]: na agricultura, em campos de golfe ou em campos de futebol, podendo ser usadas para monitorar parâmetros como o teor de água, o teor de minerais e a temperatura, através de nós enterrados que transmitem a informação em tempo real para um ou mais dispositivos acima do solo.

Os sistemas de localização têm ganho importância na última década para a localização de pessoas, veículos, animais e objetos. O sistema GPS (*Global Positioning System*) é hoje o mais utilizado, no entanto não é adequado para todas as aplicações. Para além de funcionar apenas em ambientes exteriores, possui precisão limitada e é sensível a variações súbitas das condições de propagação do sinal, que se traduzem em informação de localização errada. Devido a estas limitações, têm sido desenvolvidos sistemas de localização complementares sendo que as classes mais utilizadas são as seguintes: localização baseada em vídeo e localização baseado em Rádio Frequência (RF). A primeira classe tem sido nos últimos anos utilizada, por exemplo, para seguimento de jogadores nos jogos da UEFA *Champions League*, a segunda tem sido principalmente utilizada em ambientes interiores para seguimento de pessoas, tipicamente recorrendo a Wi-Fi. Ambos os sistemas têm ainda assim desvantagens. Os sistemas de localização de vídeo têm custo elevado e são computacionalmente exigentes, os sistemas de posicionamento RF têm tradicionalmente baixa precisão.

A medição de uma distância e a localização são dois conceitos que estão relacionados. A localização é um ponto no espaço que é descrito geometricamente ou geograficamente como um conjunto de coordenadas definidas como distâncias e ângulos em relação a outro ponto local ou global. As distâncias e ângulos podem ser utilizadas para calcular a localização de um determinado objeto [3].

Existem quatro técnicas que permitem a medição da distância e a localização a partir da análise das características físicas específicas de sinais rádio: a intensidade do sinal recebido (RSS – *Received Signal Strenght*), o tempo de chegada do sinal (TOA – *Time of Arrival*), a diferença de tempo de chegada do sinal (TDOA – *Time Difference of Arrival*) e o ângulo de chegada (AOA – *Angle of Arrival*), como se pode ver na Figura 1 [3] [4][5] .

O posicionamento utilizando o RSS tem várias vantagens sobre as técnicas de TOA, TDOA e AOA. A grande vantagem de técnicas baseadas em RSS deve-se ao facto de utilizar infraestruturas já existentes para implementar um sistema de

posicionamento. É mais simples obter informações RSS do que do tipo TOF ou AOA que exigem um processamento de sinal avançado e *hardware* especial [6]. As infraestruturas necessitam apenas de ter capacidade de leitura do RSS, algo que está disponível em praticamente todos os recetores, e ter *software* com capacidade de leitura para que seja possível estimar a posição. A capacidade de localização pode ser adicionada a um sistema sem fios por um custo reduzido.

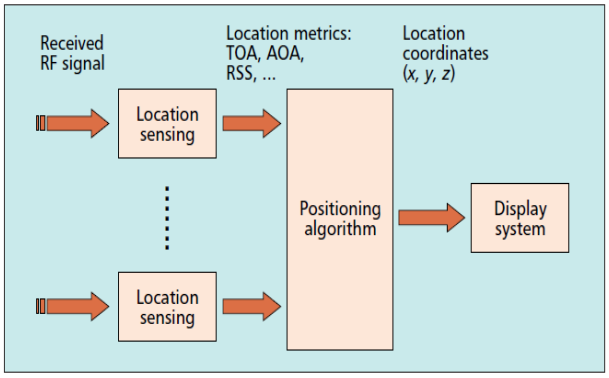


Figura 1 - Diagrama de blocos de um sistema de posicionamento wireless [4]

Existem alguns algoritmos baseados em RF que implementam as técnicas acima apresentadas: método de Fourier, método do atraso e soma, *beamformer* clássico e triangulação. No entanto, os algoritmos referidos apresentam pouca resolução e precisão quando os sinais estão correlacionados entre si [5][7]. Os algoritmos de classificação de múltiplos sinais (MUSIC – *Multiple Signal Classification*) e de estimativa dos parâmetros dos sinais via invariância rotacional (ESPIRIT – *Estimation of Signal Parameters via Rotational Invariance Techniques*) são mais precisos que os anteriores, mas que têm um custo computacional mais elevado, baseando-se na técnica AOA [5][8][9]. Por fim, existem ainda os métodos baseados em bases de dados, também chamados de *fingerprinting*, como o método *Search Nearest Neighbor*, e o método estatístico utilizando a interferência *Bayesian* [3][5][10][11].

Neste trabalho pretendeu-se desenvolver um sistema de localização de precisão recorrendo a uma rede WUN. Numa primeira fase foi feito um estudo do desempenho da tecnologia Wi-Fi em ambiente *underground*. Numa segunda fase desenvolveu-se um sistema de localização com base no estudo realizado na primeira fase e avaliou-se a sua precisão utilizando equipamento Wi-Fi *off-the-shelf*.

Ao longo deste artigo é detalhado o conceito de WUN, e apresentado o funcionamento dos algoritmos de posicionamento que utilizam base de dados, com maior destaque para o *Search Nearest Neighbor*, que foi usado com base na solução proposta. É também explicada a solução proposta para a resolução do problema, e descrito o ambiente experimental usado para a avaliação da solução proposta. Por fim são apresentadas as principais conclusões com base nos resultados experimentais obtidos.

II. WUN UTILIZANDO ONDAS ELECTROMAGNÉTICAS

As ondas eletromagnéticas (OE) são a tecnologia mais utilizada para estabelecer comunicações sem fios. A propagação de ondas eletromagnéticas em espaço livre pode ser aproximada pela seguinte expressão a que se dá o nome de equação de Friis:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi r}\right)^2 \quad \textbf{(1)}$$

em que P_r é a potência recebida, P_t a potência transmitida, G_t e G_r são os ganhos das antenas transmissoras e das antenas recetoras respetivamente, o λ é o comprimento de onda e o r a distância entre a fonte e o recetor.

Quando as comunicações são feitas através do solo, as OE têm outro tipo de comportamento, uma vez que as propriedades de propagação são diferentes. A propagação de uma OE através do solo irá depender das características do meio, como a permissividade, permeabilidade e a condutividade elétrica, sendo que estas dependem do teor de água existente no solo [12].

Nas comunicações subterrâneas U2U, U2A e A2U os meios de propagação são o solo e o ar. Os nós que se encontram à superfície ou que se encontram enterrados podem ter funcionalidades como aquisição de dados, gestão de redes e retransmissão de dados.

A superfície do solo, ou seja, a interface entre o solo e o ar, pode ter bastante impacto nas WUN. Devido a este impacto as comunicações U2U têm dois caminhos principais. O primeiro é o caminho direto entre os dois nós enterrados, que é por onde é feita a comunicação. O segundo caminho é a refração na superfície do solo, que apenas afeta comunicações com pouca profundidade. No caso das comunicações U2A e A2U o sinal terá de atravessar a superfície do solo, o que poderá provocar níveis elevados de refração e de atenuação, prejudicando a comunicação. Este tipo de efeitos irá depender da direção da comunicação, isto é, no caso de uma comunicação U2A, o sinal irá atravessar em primeiro lugar o solo e depois o ar, ou seja irá passar de um meio com um índice de refração mais alto, para um meio com um índice de refração mais baixo, o que irá ter uma menor atenuação do que uma comunicação em sentido contrário que é o caso da comunicação A2U.

III. ALGORITMOS DE LOCALIZAÇÃO COM UTILIZAÇÃO DE BASE DE DADOS

A. *Localização usando base de dados*

Existem dois tipos básicos de sistemas que usam RSS para calcular o posicionamento [3]: 1) os que se baseiam em relações analíticas de propagação de rádio; e 2) aqueles que envolvem pesquisa numa base de dados que é pré carregada com medidas de RSS (técnica de *fingerprinting*). Esta técnica baseada em RSS tem uma maior precisão, do que a primeira [3] [6].

A técnica de *fingerprinting* consiste em comparar um conjunto de valores de RSS entre o objeto e pontos de referência, em tempo real, com medições RSS obtidas

anteriormente em toda a área de cobertura de um local específico, em que as características de sinal são dependentes desse local. As leituras dos valores de RSS de diferentes localizações são armazenadas numa base de dados e combinadas com as medidas de RSS da atual localização de um determinado objeto. Esta base de dados apenas é aplicável para o sítio específico para onde foi criada, e as mudanças físicas, como mudanças climatéricas, afetam a propagação de rádio no local e podem exigir a criação de uma nova base de dados. A técnica de localização com comparação com a base de dados pode ser aplicada tanto em ambientes interiores como em exteriores. De acordo com [3] e [6], as propriedades físicas e limitações tecnológicas associadas a outras técnicas, fazem com que os esquemas de localização *fingerprinting* sejam a solução mais viável para a localização.

O principal objetivo de um sistema de localização é estimar as coordenadas (*x*, *y*) de um objeto. O processo de estimativa de localização baseado em *fingerprinting* tem duas fases, a *off-line* e a *on-line* [13] que podem ser vistas na Figura 2.

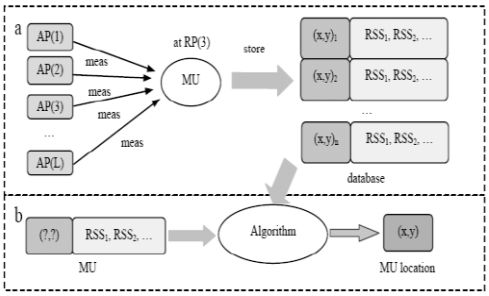


Figura 2 - As duas fases de localização fingerprinting (a) fase *off-line* (b) fase *on-line* [13]

Na fase *off-line* (Figura 2 (a)), ou fase de pesquisa, é criada uma base de dados. As medições dos valores de RSS dos pontos de referência são feitas através de vários APs (*Access Point*). A intensidade do sinal varia ao longo do tempo, devido a interferências que possam existir no local onde estão a ser realizadas as medições. Por esse motivo é necessário realizar várias medições para o mesmo ponto de referência.

A natureza da base de dados que é criada a partir dos dados recolhidos depende do método de comparação, que será abordado de seguida. As componentes de informação na base de dados são identificadas como posições de referência.

A segunda fase do processo de localização, ou fase *on-line* (Figura 2 (b)) trata da medição do RSS em tempo real. Nesta fase, pretende-se saber o posicionamento de um certo objeto que se encontra na área de cobertura. Os RSS do objeto medido pelos APs são registados. O conjunto de RSS adquirido nesta fase é comparado com a base de dados, associando-se a todos os pontos de referência ou aos pontos de melhor correspondência para indicar a estimativa da localização do objeto.

A maior desvantagem do método que utiliza a base de dados é o facto de não poder ser reutilizado num ambiente diferente daquele para o qual foi criado. Existem métodos diferentes de comparar medições de dados em tempo real com a base de dados, incluindo: 1) com base na distância euclidiana mínima, referido como o método *Search Nearest Neighbor*, que será o utilizado neste trabalho; 2) o método estatístico utilizando a interferência *Bayesian* [3][10][11].

B. Algoritmo Search Nearest Neighbor

Alguns estudos têm incidido neste algoritmo, também denominado por método das distâncias euclidianas [3][10][11][14][15]. Primeiro a base de dados é criada, na fase *off-line* onde é avaliado o local onde será feita instalação do sistema de localização. Aqui são efetuadas várias medições de RSS em cada ponto de referência que serão posterioremente inseridos na base de dados. Desta forma, cada elemento da base de dados é representado pelo seguinte vetor:

$$V_n = (x, y, p, s_1, s_2, s_k, \ldots s_K) \quad \textbf{(2)}$$

em que as componentes são as coordenadas de localização *x*, *y*, a orientação *p*, e as intensidades de sinal em s_k, *k* representa o AP em que é executada a medição, *K* é o número de APs e o *n* é o índice do ponto de referência.

Durante a fase de medição em tempo real, o objeto a localizar transmite e cada AP mede o RSS, ou em sentido oposto, os APs transmitem e o objeto mede os RSS. É feita uma média de um número de leituras de RSS de um objeto vinda de cada um dos APs. As leituras são normalizadas de modo a que os RSS dos diferentes pontos de acesso sejam mantidos, permitindo a comparação com a base de dados. Esta normalização é realizada para reduzir a redundância de dados e de forma a tornar os dados consistentes Sendo assim a normalização será feita em relação a um dos APs do sistema. As leituras de RSS de cada um dos APs irá formar um vetor ($s_1 \ldots s_k$), cujos componentes são a média normalizada do RSS do objeto em cada um dos APs. De seguida é verificada a base de dados previamente preparada, e verificados os RSS que se aproximam mais dos valores das medições em tempo real, neste caso utilizando o algoritmo da distância euclidiana mínima. Para cada entrada na base de dados um valor D_n é calculado da seguinte forma:

$$D_n = \sqrt{\sum_{i=1}^{K} \left(S_{Ti} - S_{i,n} \right)^2} \quad \textbf{(3)}$$

onde S_T é o RSS *on-line* do objeto, *S* é o vetor da intensidade do sinal da base de dados, *i* é um índice do AP, e *n* é o índice da posição de referência. O menor *D* corresponde a uma estimativa da posição onde o objeto se encontra mais próximo. Uma maior precisão da posição do objeto pode ser obtida ao escolher mais que um ponto de referência da base de dados e com a média das suas coordenadas, que já são conhecidas, obtém-se uma estimativa da localização do objeto. Para isso utiliza-se as seguintes expressões:

$$x = \frac{1}{L}\sum_{l=1}^{L} x_l \quad \textbf{(4)}$$

$$y = \frac{1}{L}\sum_{l=1}^{L} y_l \quad \textbf{(5)}$$

em que L representa o número de vizinhos, ou pontos de referência da base de dados mais próximos, x_l as coordenadas x desses vizinhos e y_l as coordenadas y desses vizinhos.

IV. Avaliação experimental da solução proposta

A. Solução Proposta

O algoritmo de localização *Search Nearest Neighbor* foi escolhido dado que, como já foi referido, consegue ser mais preciso que os outros, não sendo necessário equipamento com custos elevados para realizar as experiências, sendo este um dos objetivos destas experiências. Foi utilizada uma comunicação U2A, uma vez que este tipo de comunicação tem uma menor atenuação do RSS, em relação à comunicação A2U, pois o sinal tem origem no meio com maior índice de refração (solo), para o meio com menor índice de refração (ar). Para realizar este tipo de comunicação foram usados 4 nós, enterrados a 20 cm de profundidade, a emitir, e à superfície um nó a receber essa informação dentro da área estipulada. Esta mesma área tinha 4 m de lado, com 25 pontos de referência com 1 m de distância entre eles.

B. Ambiente experimental

Em relação ao *hardware* foram necessários 4 nós enterrados e um nó móvel acima do solo cerca de 90 cm. O nó usado à superficie tratou-se de um adaptador USB wireless igual aos enterrados. Os 4 adaptadores USB wireless dos 4 nós enterrados foram ligados a quarto Raspberry Pi.

Em termos de *software* o nó móvel utilizou uma distribuição Linux, onde um *script*, utilizando a ferramenta *iwlist*, era executado e guardava num ficheiro o ESSID (*Extended Service Set Identification*) e o RSS de cada nó enterrado de 2 em 2 segundos. Em relação aos nós enterrados, uma vez que estes estavam ligados a Raspberry Pis para arrancarem como AP, foi utilizado o Raspbian, uma distribuição Linux para Raspberry Pi.

Para execução da experiência após a seleção do local marcou-se uma grelha com fios espaçados por 1 m de comprimento e 1 m de largura de forma a saber-se a localização dos pontos de referência em que seriam recolhidos os RSS para inserir na base de dados, como pode ver-se na Figura 3. Na área selecionada o solo era argiloso, um pouco rochoso, e não uniforme típico de um terreno agrícola. Cada nó estava localizado em cada um dos vértices do quadrado, mais precisamente, o AP1 perto do ponto de referência 1, o AP2 perto do ponto de referência 21, o AP3 perto do ponto de referência 5 e por ultimo o AP4 perto do ponto de referência 25.

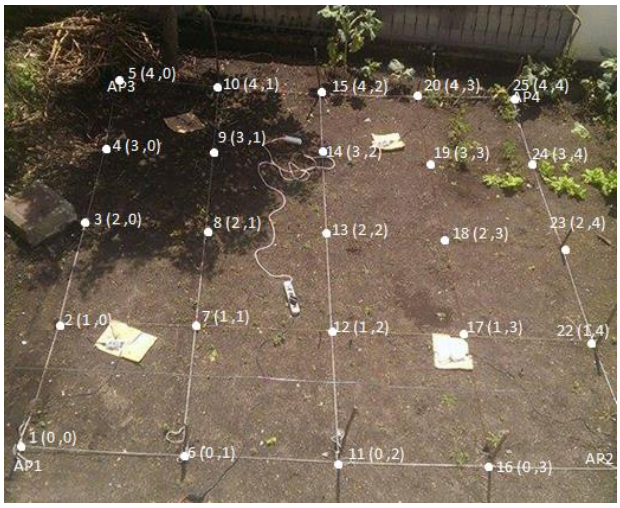


Figura 3 - Local de realização das experiências com os respetivos pontos de referência e suas coordenadas

Numa primeira fase foi realizada a medição dos pontos de referência. Em cada ponto de referência foram lidas e guardadas as medição de 50 amostras de RSS, e a sua média foi inserida na base de dados. Note-se que para esses pontos de referência já eram conhecidas as suas coordenadas. Na base de dados foi feita uma normalização em relação a um dos APs, em todos os pontos de referência. Neste caso a normalização foi feita sempre em relação ao AP1.

Depois de terem sido realizadas as medições dos pontos de referência e serem colocados na base de dados foram escolhidos alguns pontos aleatórios nesta mesma área. Nos pontos escolhidos foi feita uma leitura de 50 amostras tal como nos pontos de referência, e calculada a média. Depois de ter recolhido os RSSs nesses pontos foi então usado o algoritmo *Search Nearest Neighbor* para se estimar a localização desses pontos. De seguida, é apresentado um exemplo de estimação de posição de um dos pontos aleatórios da experiência, para melhor se perceber o processo utilizado. Com a média das leituras de RSS de cada AP neste ponto aleatório foi feita uma normalização em relação ao AP1, como se pode ver na seguinte Tabela 1. A normalização consiste no cálculo da diferença dos RSSs de todos os APs em relação ao AP1.

Tabela 1 - Normalização do Ponto Aleatório em dBm

	RSS	APk-AP1
AP1	-55,98	0
AP2	-83,43	-26,45
AP3	-78,38	-21,4
AP4	-77,3	-20,33

Utilizando a Equação 3, com os dados dos pontos de referência normalizados cálculou-se a distância métrica que pode ser vista num excerto da Tabela 2. De seguida também é possível verificar um exemplo, para o cálculo, da distância D, utilizando a Equação 3 para o ponto de referência 1.

$$D_n = \sqrt{\sum_{i=1}^{4}(0-0)^2 + (-26,45-(-25,73))^2 + (-21,4-(-23,45))^2 + (-20,33-(-21,8))^2} = 2,63 \quad \textbf{(6)}$$

Tabela 2 - Distância métrica entre o RSS normalizado da base de dados com o RSS normalizado do Ponto Aleatório

PR	1	2	3	4	5	6	7
D	2,63	6,96	16,46	34,76	49,91	16,01	22,72

Os valores que estão sombreados a azul são as distâncias menores obtidas utilizando o algoritmo, o que significa que foram os pontos de referência que estavam mais próximos do ponto aleatório cuja posição se pretendia estimar. Como já se conheciam as coordenadas destes pontos de referência e usando a Equações 4 e 5 para estimar as coordenadas *x* e *y* respetivamente, foi possível então obter uma posição para esse ponto aleatório. Um dos aspetos em que este estudo incidiu foi na análise do número de pontos de referência vizinhos da base de dados com que se obteria um menor erro de posição em relação à posição real desse ponto aleatório. Neste estudo foram utilizados então 2, 3, 4 e 5 vizinhos. Na Tabela 3 é possível ver as coordenadas estimadas deste ponto aleatório para 2, 3, 4 e 5 vizinhos e o respectivo erro em relação à posição real. A título de exemplo é mostrado o cálculo das coordenadas *x* e *y* para 3 vizinhos.

$$x = \frac{1}{3}(0+1+0) = 0,33m \quad \textbf{(7)}$$

$$y = \frac{1}{3}(0+0+1) = 0,33m \quad \textbf{(8)}$$

Tabela 3 - Coordenadas e erros em metros (m) utilizando os vários números de vizinhos

	2 vizinhos	3 vizinhos	4 vizinhos	5 vizinhos	Valor real (m)
x(m)	0,5	0,33	0,75	0,8	0,41
y(m)	0	0,33	0,25	0,4	0,44
Erro (m)	0,45	0,14	0,39	0,4	

O valor real foi obtido com recurso a uma fita métrica aquando da leitura do RSS nesse ponto. Pela Tabela 2 é possível verificar que com a utilização de 2 vizinhos os pontos de referência menor *D* seriam, 1 e o 2, com 3 seria incluído o ponto de referência 6, uma vez que é o 3º ponto com menor D, com 4 seria incluído o 3 e com 5 seria incluído o 7, uma vez que seriam os 4º e 5º pontos de referência com menor D. Em relação ao erro, este é calculado da seguinte forma:

$$Erro = \sqrt{(x_{med} - x_{real})^2 + (y_{med} - y_{real})^2} \quad \textbf{(9)}$$

em que x_{med} se trata da coordenada *x* obtida pelo algoritmo, x_{real} trata-se da coordenada *x* real, do ponto resultante da medição, y_{med} a coordenada *y* obtida pelo algoritmo e y_{real} a coordenada *y* real do ponto. . Por exemplo, o cálculo do erro para o caso de 3 vizinhos com base nos valores estimados e medidos na realidade, dado por:

$$Erro = \sqrt{(0,33-0,41)^2 + (0,33-0,44)^2} = 0,14m \quad \textbf{(10)}$$

C. Resultados experimentais

Nesta experiência foi considerada uma amostra de 26 pontos aleatórios com 50 leituras de RSS em cada um dos pontos aleatórios. Depois de ter sido feita a recolha das leituras de cada um dos pontos foi feita a média de cada um e usado o algoritmo *Serach Nearest Neighbor* como já foi referido. Utilizando as Equações 4 e 5 variando o número de pontos vizinhos, *L*, com 2, 3, 4 e 5 vizinhos mais próximos, obteve-se os seguintes erros em cada um dos pontos.

Figura 4 - Erro em cada um dos pontos aleatórios utilizando 2 vizinhos

Com base nos resultados da Figura 4 é possível verificar que com 2 vizinhos o menor erro obtido foi de cerca de 3 cm no ponto aleatório 23 e o maior erro de cerca de 94 cm, no ponto aleatório 18. Deste conjunto de 26 amostras a mediana do erro é de cerca de 46 cm.

Figura 5 - Erro em cada um dos pontos aleatórios utilizando 3 vizinhos

Como se pode verificar pela Figura 5, com 3 vizinhos, o ponto aleatório com menor erro foi o 1 com cerca de 13 cm e o maior erro de cerca de 86 cm no ponto aleatório 11. Neste conjunto de 26 amostras a mediana do erro é de cerca de 41,8 cm.

Figura 6 - Erro em cada um dos pontos aleatórios utilizando 4 vizinhos

Utilizando 4 vizinhos, pode-se concluir pela Figura 6 que o erro mais baixo acontece no ponto aleatório 16 com cerca de 22 cm enquanto que o erro máximo acontece no ponto aleatório 15 com cerca de 1,12 m. A mediana do erro neste caso é de cerca de 50 cm.

Figura 7 - Erro em cada um dos pontos aleatórios utilizando 5 vizinhos

Com a utilização de 5 vizinhos, como se pode verificar pela Figura 7, o erro mínimo foi de 15 cm no ponto aleatório 8 e o máximo no ponto 15, com cerca de 1,28 m. A mediana do erro com 5 vizinhos foi de cerca de 59,7 cm.

Figura 8 - Mediana do erro em função do nº de vizinhos

Pelo gráfico da Figura 8 é possível verificar que com a utilização de 3 vizinhos é possível ter uma maior precisão. Com os resultados obtidos, neste caso, na maior parte das vezes o algoritmo identificou os pontos de referência reais como sendo os vizinhos que estavam mais próximos do ponto aleatório. Por outro lado o menos vantajoso será utilizar 5 vizinhos uma vez que é o que tem uma mediana maior de erro. Isto acontece pelo facto do algoritmo assumir como seus vizinhos mais próximos alguns pontos de referência que estão mais distantes, de onde foram feitas as medições dos pontos aleatórios, não sendo esses os reais vizinhos. Existem várias causas para isto acontecer. Por exemplo, o facto do meio não ser uniforme, podendo haver rochas, humidade e outras substâncias no solo que possam condicionar o valor do RSS medido em determinada direção. Em relação às outras duas experiências para *L* igual a 2 e 4, obteve-se um erro superior ao de 3 vizinhos, mas inferior a 4 e a 5 enquanto que com 4 obteve apenas um erro inferior a 5 vizinhos.

V. AGRADECIMENTOS

Este trabalho é financiado pela FCT – Fundação para a Ciência e a Tecnologia no âmbito do projeto UID/EEA/50014/2013.

VI. CONCLUSÕES

O trabalho aqui apresentado teve como objetivo avaliar técnicas de localização de objetos utilizando uma WUN e o algoritmo *Search Nearest Neighbor*. Recorrendo à solução proposta consegue-se uma mediana de erro que não ultrapassa os 41,8 cm.

O facto do solo ter as características já referidas provoca variações significativas na propagação rádio, o que afetou diretamente os resultados finais das experiências realizadas. O solo à superfície tem irregularidades ao longo de toda a área de teste, sendo mais um fator que influenciou os resultados.

Como trabalhos futuros, considerer-se-ão testes noutros tipos de solos, por exemplo em areia, para aplicação em futebol de praia por exemplo, utilização de mais nós enterrados de modo a verificar em quanto diminuiria o erro e o desenvolvimento de uma aplicação de simulação de posicionamento neste contexto, com diferentes tipos de solo.

REFERENCIAS

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless Sensor Networks: A Survey”, 2002.

[2] E.P. Stuntebeck, D. Pompili, T. Melodia, “Wireless Underground Sensor Networks using Comodity Terrestrial Motes”, Wireless Mesh Networks, 2006. WIMesh 2006, 2nd IEEE workshop, Setembro 2006.

[3] A. Bensky, “Wireless Positioning Technologies and Applications”, Artech House, 2004.

[4] K. Pahlavan, X. Li, “Indoor Geolocation Sciense and Technology”, Next-Generation BroadBand Wireless Networks and Navigation Services, 2002.

[5] E. V. Pereira, “Desenvolvimento de um sistema de localização de fontes Rádio Frequência para aplicações indoor”, 2011.

[6] P. Prasithsangaree, P. Krishnamurthy, P.K. Chrysanthis, “On Indoor Position Location With Wireless LANs”

[7] A. Savvides, H. Park, M. Srivastava, “The bits and flops of the n-hop multilateration primitive for node localization problems”, in First ACM International Workshop on Wireless Sensor Networks and Applications, Setembro 2002.

[8] S.V. Schell, W.A. Gardner, “High-resolution direction finding”, Proceedings of the IEEE, 74 (7), 1986.

[9] R. Roy, T. Kailath, “Espirit-estimation of signal Parameters via rotational invariance techniques”, IEEE Transactions on Acoustics, Speech, and Signal Processing, September 1986.

[10] S. Khodayari, M. Maleki, E. Hamedi, “ A RSS-based Fingerprinting Method for Positioning based on Historical Data”

[11] N. Chang, R. Rashidzadeh, “Robust Indoor Positioning using Differencial Wi-fi Acess Points”, IEEE Transactions on Consumer Electronics, Vol. 56, No 3, August 2010.

[12] S. Yoon, L. Cheng, E. Ghazanfari, S. Pamucku, M.T. Suleiman “A radio propagation model for wireless underground sensor networks”, IEEE Globecom URPM, 2011.

[13] O.M. Bodawy, M.A.B. Hasan, “Decision Tree Approach to Estimate User Location in WLAN Based on Location Fingerprinting”, 2007, 24th National Radio Science Conference.

Índice de Autores

A
Aguiar,Rui 44
Antunes,Mário 19
Azevedo,Filipe 38

C
Cabrita,Cristiano 63
Campos,Nelson 32
Campos,Rui 69
Carvalho,Paulo 52
Corujo,Daniel 44
Costa,António 25
Costa,Hugo 63
Cunha,João 7

D
Dias,Jaime 57

F
Fernandes,André 46

G
Gama,óscar 52
Gomes,Diogo 19
Guimarães,Carlos 44

L
Lima,Emanuel 52
Lima,Solange 7
Lopes,António 25

M
Macedo,Joaquim 25
Mamed,Jorge 46
Mamede,Jorge 57, 69
Martins,José 13, 32
Meneses,Flávio 44
Monteiro,Jânio 63

N
Nicolau,Maria 25

O
Oliveira,José 69

P
Pinto,António 46

R
Rato,Luís 13
Rodrigues,João 57

S
Santos,Alexandre 25
Semião,Jorge 63
Silva,Afonso 19
Silva,Fernando 38
Silva,João 7
Silva,Luís 38
Silva,Ricardo 7
Souto, André 1

V
Venâncio,José 69