An efficient approach to the Network Division Problem, VLANs configuration and WLANs hosts grouping

Marcello Esposito, Antonio Pescapè and Giorgio Ventre
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II" (Italy)
{mesposit, giorgio, pescape}@unina.it

ABSTRACT

Nowadays, network design and management can be considered as a well understood topics. A thorough view of a complete project of a network infrastructure is definitely more important than its single details. Among the many innovations introduced in the IP networks, an interesting issue is represented by enhanced architectures where services like telephony and video transmission are provided on the same infrastructure used for data traffic. At the same time, actual networks are heterogeneous in terms of access network technologies, end user's device and finally operating systems. In these new scenarios innovative and efficient management's approaches are needed. This paper describes a proposal to improve the management of different network scenarios. This work presents a "*Partitioning Algorithm*" and some of its possible practical applications in the field of shared LANs, full switched LANs, VLANs (*Virtual Local Area Networks*) configuration, and in WLANs (*Wireless Local Area Networks*) environments.

I. INTRODUCTION

Most network designers have to balance the trade-off between existing implementations and up-grades due to technological innovations, having, at the same time, to cope with the problem of guaranteeing continuity in the service provided by networks. Among the main issues, the coexistence of different protocols over heterogeneous (wired and wireless) networks plays a major role; if underestimated, it can bring the whole architecture to work in sub-optimal conditions and, in the worst case, even to starvation. For these reasons a continuous management activity is needed. Configuration and management activities are frequently performed both in Local Area and Campus Networks due to intrinsic variability characterizing such networks as well as innovative services provided by means of them. These activities are both time and money consuming since they usually are under the responsibility of network administrators and managers. One of the most frequently used management activity is the *network segmentation* or *network partitioning*.

In literature many studies on the NDP (*Network Division Problem*) are present. These works are based on simulation or analytical study concerning a minimization study of a particular "*fitness function*". These minimization studies take into account only a short range of parameters: this choice permits to reach only a partial and theoretical sub-optimal solution [1] [2] [3]. On the other hand, by taking into account all needed network parameters the "optimum" itself could change. In this work a concrete approach for an efficient solution of the NDP is presented.

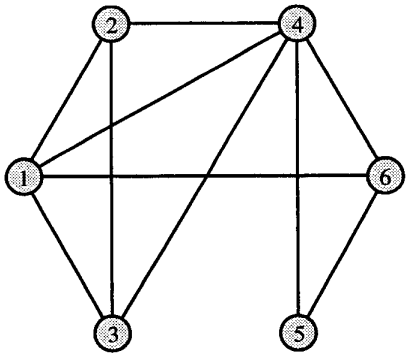
II. PARTITIONING ALGORITHM

Our segmentation approach allows gathering of all active hosts in a same group. Current tools are based on static information to assign a host to the pertaining group. Conversely, our *Partitioning Algorithm* relies on *on-line* measures and computations. This feature enables efficient and dynamic assignment of hosts to groups according to the current traffic figure by assuming that hosts transmitting data will continue their data exchange also in the future [4]. Sniffing activities, i.e. the activities related to logging the existing traffic flowing in a network infrastructure, help to build "*Traffic Table*" and "*Devices Table*". While the former contains information about "*who is talking to whom*", the latter takes into account the total amount of traffic generated by every single station. These actions are automatically and transparently performed. *Traffic Table* contains all measures concerning traffic flowing towards all network devices, included switches and routers, as well as both broadcast and multicast traffic. Due to the need to analyze communications among the hosts, a "*Discovery module*" is

introduced in order to discover all the active computers in the network. Such a module is able to distinguish devices, and it returns addresses of switches and routers that have to be deleted from the final table. During this process, a supplementary table is created: the *Address Table* . This table gives the association between IP addresses and MAC addresses. As an example, in case of 254 hosts, a square matrix with null diagonal is built. In a 192.168.128.0 network, the element $a_{12,21}$ represents the traffic between 192.168.128.12 and 192.168.128.21. The *Partitioning Algorithm module* configures groups starting from traffic measurements. In case of application on other matrixes, as added value, the Partitioning Algorithm computes the *Maximum Groups* (*Maximum Cliques*) [5][6][7] among all matrix elements.

The algorithm creates groups on the basis of traffic flowing among network stations belonging to a Campus or Local network. This requires the knowledge of the status of all networked stations, in particular whether each pair of stations belongs to the same group: *Threshold Traffic* (and its calculus) is the parameter that gives the information concerning the correlation between two stations. Under normal circumstances this parameter is automatically calculated (e.g. as mean value), but it is also possible to set it in a manual way or even to let it come out from a closed-loop algorithm. In this way, a more flexible approach can be taken, since the network manager has full control on the threshold whose value has a great impact on the overall algorithm results. We added this possibility for our experimentation, too. Stemming from the analysis of measures carried out in laboratory, we choose the mean value of traffic matrix elements as automatic parameter. It is important to notice that, during computation, a variation in the threshold causes a variation in the number of groups, but there is not a proportional correlation between them. A bigger threshold does not roughly imply a lower number of groups. This behavior can be explained by observing the variation in communication matrix with respect to the variation in threshold, which causes a modification in *Maximum Groups*. Different criteria have been adopted for the threshold. While in a first time a multiplier factor has been chosen, in the following a new simpler and more correct method has been preferred: after the automatic calculus of mean value, the sorted vector containing traffic values is inspected until the number of groups fits the desired value. The first step in the algorithm is the computation of *Traffic Matrix*. For instance, for C class addresses, the matrix is 254x254, in case of B class addresses the matrix will be 65534x65534. *Traffic Matrixes* are usually sparse and suggests the use of *Sparse Matrix* and of *Three Vectors Method with pointers* implementations of the necessary data structures. After the Matrix acquisition, the threshold is computed. Hence, elements that overtake threshold are considered and the matrix is modified in a *Triangular Boolean Matrix*.

In the following, the algorithm will be described with reference to the example graph shown in Figure 1a. The depicted configuration involves six hosts. For instance, host 1 is sending (and/or receiving) a significant amount of data to hosts 2, 3, 4, and 6 (which is represented by edges in the graph); host 5 talks with just hosts 4 and 6. An alternative representation of the same graph is the incidence matrix shown in Figure 1b. The generic element $e_{i,j}$ is *1* if host *i* is talking with host *j*, *0* elsewhere. Since we are considering full-duplex connections, edges in the graph are undirected and the matrix is triangular.



a)

	1	2	3	4	5	6
1	-	1	1	1	0	1
2		-	1	1	0	0
3			-	1	0	0
4				-	1	1
5					-	1
6						-

b)

Figure 1: An example graph and the related Boolean matrix.

The algorithm proceeds creating a tree by means of the shown recursive procedure. Each node represents a set of hosts talking to one each other. In the first step, the algorithm finds the largest groups containing host 1 (anyway, it might start with any other node in the graph): a new node containing host 1 is created as a child of the root.

Starting from it, new children are added, one for each host talking to host 1. Hence, nodes 12, 13, 14, and 16 are created. Node 15 is not created because host 1 is not connected with host 5 and, thus, no group containing these two hosts can exist. As far as node 12, new children are added for each host talking with both host 1 and host 2. Hence, nodes 123 and 124 are created. In the same way, node 1234 is added as a child of node 123. Node 1234 cannot be further expanded with nodes 12345 and 12346 since neither host 5 nor host 6 talk to all of the nodes in the set {1,2,3,4}. Similarly, node 124 does not allow the creation of nodes 1245 or 1246. Furthermore, trying to add node 3 to node 124 is unnecessary since groups containing nodes {1,2,3} have already been considered starting from node 123. This is a general rule that can be resumed as follows: during expansion of the generic node containing hosts {i, j,...,k} (with i<j<...<k), only nodes whose index is greater than k have to be considered. Nodes 1234 and 124, thus, cannot be further expanded and the recursion recedes as far as considering node 13. Only node 134 can be appended to 13. Similarly, node 146 can be appended to 14. This completes the growth of tree descending from node 1. The algorithm continues creating node 2 and, because of the above rule, it tries to include only the hosts having index greater than 2. The recursion comes to an end when no more children can be added to the nodes in the tree. The resulting tree is shown in Figure 2. The last step of the algorithm consists in iterating through the leaves in the tree from left to right and selecting them according to the following rule: a leaf is selected if no other larger leaves have already been selected during the iteration. This process brings to select the following leaves: 1234, 146, 234, and 456 (shown with a double border in the tree). Leaves 124 and 134 have not been selected because they represent a subset of leaf 1234. The same applies to the groups containing two nodes (e.g. group 16 is a subset of group 146).

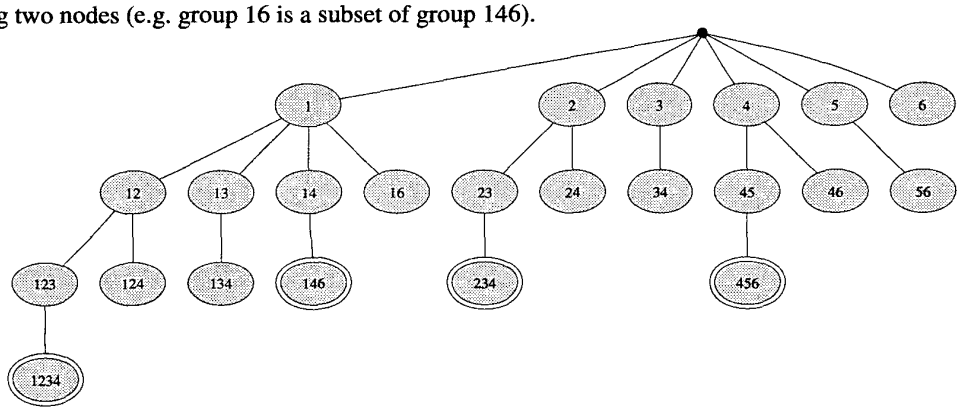


Figure 2: The tree created by the algorithm

In figure 3 the description in *Pascal like* notation is reported. The algorithm *FindMaxGroups* recursively add nodes to a given set of nodes (i.e. a potentially maximum group). At the first step of recursion, it is invoked as follows:

FindMaxGroups(ϕ , firstNode);

A node is successfully added to a group if such a node is connected to each and every node in the group. In this case, the search continues starting from this new group and tries to further add nodes. When a step of recursion corresponds to a leaf (i.e. no further nodes can be added to a group) the group happens to be maximum and it is passed to the storing procedure. This procedure stores the group if it contains at least two nodes and the group is not contained inside some of the already stored groups.

III. NETWORK DIVISION PROBLEM: FROM A FLAT NETWORK TO A FULL-SWITCHED NETWORK

Among all techniques for the *"network performance management"* one of the most spread used is the *segmentation"*: the key concept of this methodology is the division of an entire network in smaller host aggregates (figure 4). The segmentation provides better performance for the entire network, optimizing overall communications. In this field, the Partitioning Algorithm (PA) is positioned in the framework of *Network Division Problem (NDP)* theory.

The PA is useful whenever is needed to move from a flat LAN (where a large number of hubs are present) to a full-switched LAN.

```
procedure FindMaxGroups(nodes: Nodes, currentNode: Node);
var
   leaf: Boolean;
   _nodes: Nodes;
   n: Node;
begin
   leaf = TRUE;
   for n := currentNode to lastNode do
   begin
   if (nodes ∪ {n} is a valid group) then { true if n is connected to any node in nodes }
   begin
   leaf = FALSE; { node n has been added: the current step of recursion does not correspond to a leaf }
   _nodes = nodes ∪ {n}; { create the new group called _nodes }
   FindMaxGroups(_nodes, n+1); { recursively search starting from _nodes = nodes ∪ n }
   end;
   end;

   if (leaf) then
     StoreGroup(nodes); { store group if necessary }
end;
```

Figure 3: Solution Procedure in a Pascal-like language

The PA represents an efficient criterion of segmentation with respect to inter-switch traffic and communications and in general for network performances like end-to-end delay, jitter and throughput.

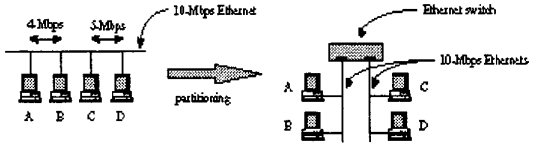


Figure 4: Example of LAN segmentation

In this case the PA is applied with the following association: on the matrix rows and columns there are the host addresses present in the flat LAN; matrix elements identify communications between hosts; the max number of groups is the number of usable switches. The same approach is applicable when instead of hosts there are entire network segments (hubs or switches). In this last case it is possible to refer to hub or switch addresses.

IV. VLANs CONFIGURATION

The previous solution presents the only disadvantage of a static configuration. By using PA in a VLAN configuration problem it is possible to provide a dynamic connection to several groups (VLANs). Segmenting a corporate network in VLANs (figure 5, where different colors represent different VLANs) allows reducing the broadcast traffic generated by devices on the network [8]. The VLAN paradigm represents a mean for the provision of many important features, such as security, virtual working groups and controlled resource sharing. In this case the PA is applied with the following association: on the matrix rows and columns there are the addresses of hosts belonging to the switched LAN; matrix elements are the communications between the hosts; the max number of groups is the number of needed VLANs. In [9] is presented PA applied to a VLANs scenario over a real integrated Campus Network.

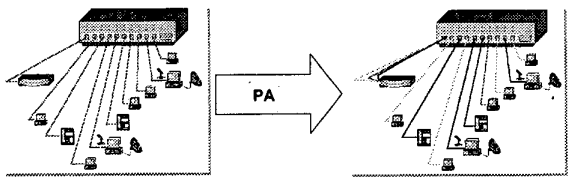


Figure 5: Example of VLAN configuration on a single switch

V. WLANs HOSTS GROUPING

When a new station is connecting to an 802.11 WLAN where several Access Points (APs) are present, the choice of the AP is made by the user on the base of his access information. When more APs are available, the choice is made, for instance, by evaluating signal powers. By using the PA it is possible to define which station must be assigned to which AP. In this scenario the first hosts attachment could be performed in a random fashion and after a transitory time (during this time a *traffic sniffing* phase is performed and the *traffic matrix* is built) the assignment follows the PA results. This *modus operandi* provides a network performance improvement and a load balancing between APs. In addition there is a traffic reduction between APs. In this case the PA is applied with the following association: on the matrix rows and columns there are the addresses of hosts belonging to the WLAN; matrix elements are the communications between the hosts; the max number of groups is the number of usable APs.

VI. DISCUSSION, CONCLUSIONS AND ISSUES FOR RESEARCH

This paper presents a solution that allows the configuration of network devices in an automatic fashion in order to facilitate traffic management and network performance. The solution is based on a Partitioning Algorithm. The application of the Partitioning Algorithm in different fields has been presented with reference to several scenarios: shared LAN, full-switched LAN, VLAN and WLAN.

In order to show the benefits of our proposed approach we discuss qualitatively the performance of the schemes described above. In the case of the first two schemes (Network Division Problem and VLANs configuration) there is an obvious reduction of broadcast traffic. In the case of WLANs host grouping there is a comprehensible reduction of the traffic between different Access Points. Besides providing qualitative consideration, using experimentations on real networks, it will possible to demonstrate the validity of our approach with end-to-end delay, jitter and throughput as the analyzed performance parameters. In our experimentations we ll use a multiservice network where both data and voice traffic will be in place. In order to study the benefit of our proposal we envisage the use of delay measurement in different network configurations and, in the case of voice traffic, the use of the MOS (*Mean Opinion Score*) parameter. Currently, we are experimenting some preliminary configurations: first output results demonstrate the correctness of our approach.

As far as future work, it is clear that a fundamental issue is related to the behavior of the proposed approach in large-scale scenarios. Indeed, scalability represents one of the most important aspects that should be emphasized when implementing network architectures. Until now, our approach has been tested on an experimental testbed. A detailed analysis of potential bottlenecks of our proposal on different network architectures is needed, with respect to both the single modules and their orchestrated operation. We aim to evaluate the relationship between the network configuration time and the network size. Thus, our future goal is to setup a certain number of networks of different size (in the different shown fields) so to measure the time needed to carry out the application of our Partitioning Algorithm.

VII. REFERENCES

[1] R. Elbaum and M. Sidi, Topological Design of Local-Area Networks Using Genetic Algorithms, Proceedings of IEEE INFOCOM 95, pp. 64 - 71, 1995.

[2] M. Nusekabel, Switched Network Partitioning Using Tabu Search, Thesis for Master of Science in Computer Science, University of South Florida, August 1997.

[3] C. Farrell, J. Simpson, M. Schulze, and D. Kieronska, Analyser: A Genetic Algorithm Based Network Divider, URL: http://www.cs.curtin.edu.au/~netman/analyser.html.

[4] S. Crosby, I. Leslie, H. Huggard, J. Lewis, B. McGurk e R. Russel, Predicting Bandwidth Requirements of ATM and Ethernet Traffic , IEEE 13th UK Teletraffic Symposium, Mach 1996

[5] E. Balas, C. S. Yu, Finding a Maximum Clique in an Arbitrary Graph , SIAM J. Computing, Vol. 15, No 4, 1986.

[6] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, The Maximum Clique Problem , Handbook of Combinatorial Optimization, vol. 4, Kluver Academic Publisher, Boston Pattern MA, 1999.

[7] C. Bron and J. Kerbosch, Finding All the Cliques in an Undirected Graph , Communication of the Association for Computing Machinery 16 (1973), 575-577.

[8] S. Rooney, C. Hörtnagl, J. Krause, "Automatic VLAN creation based on on-line measurement", ACM SIGCOMM 1999, pp: 50 - 57, Volume 29 - Issue 3 (July 1999), ISSN: 0146-4833
[9] A. Pescapè, S. D'Antonio, M. D'Arienzo, G. Ventre, "An Architecture for Automatic Configuration of Integrated Networks", accepted for publication at IFIP/IEEE NOMS 2004, April 2004