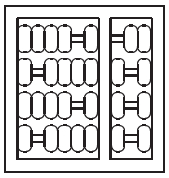

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Resource Allocation in Switchlets Networks

Nelson L. S. da Fonseca *Antonio Pires Castro*
Alexandre Teotonio Rios

Technical Report - IC-02-009 - Relatório Técnico

September - 2002 - Setembro

A Procedure for Resource Allocation in Switchlet Networks

Nelson L. S. da Fonseca, Antônio P. Castro Jr and Alexandre T. Rios

State University of Campinas

Institute of Computing

Campinas, Brazil

Abstract[1] - In this paper, a procedure for resource allocation method in switchlet networks is introduced. The procedure is based on restricting the search space for the solution of a multi-commodity flow problem The proposed approach is accurate, and amenable to real time implementation.

I. INTRODUCTION

The ability to rapidly deploy new services in response to market demand will be a key factor for the survivability of network service providers. In line with that, network programmability has been a focus of interest of the research community. Programmable network techniques [1]-[3] includes the programming of signalling, as well as the allocation of the resources of switches. Switchlet [4]-[7], a programmable network technique, allows the allocation of a subset of the resources of a switch and the co-existence of multiple control architectures in a single switch.

Virtual Private Networks (VPNs), or overlay networks, are networks of resources built on the top of an existing network infrastructure, and are used for traffic segregation. VPNs resemble private leased-line networks, however, the allocation of permanent resources is not required. Programmable networks take the VPN concept a step further. Signalling and any aspect of the underlying network resources, which can be programmed, are under control of VPNs in programmable networks.

The arrival of requests for VPN establishment can be highly dynamic and resources can be frequently allocated/deallocated. Therefore, it is of paramount importance to maximize network resource utilization. Moreover, resource allocation decisions have to be made in real time.

In this paper, an approach for resource allocation in switchlet networks is presented. The proposed solution falls into the category of the multicommodity flow type of solution [7]-[8], which is a NP-hard one [9]. To decrease the computational complexity of the multicommodity flow solution, and consequently, to be able to obtain a procedure amenable to real time implementation, the *k* shortest paths between a pair of nodes are taken into account. Only those paths which satisfy the QoS requirement of a VPN are considered. Furthermore, the feasibility of a real time implementation is demonstrated.

1. This work was partially sponsored by CNPq and CAPES
e-mails: {nfonseca, apcastro, alexrios}@ic.unicamp.br

This paper is organized as follows. Section II describes the switchlet concept. Section III introduces a method for resource allocation in switchet networks. Section IV analyses the effectiveness of the proposed method, and conclusions are drawn in section V.

II. SWITCHLETS

Switchlet is an open signalling concept [3]-[5]. The set of resources and signalling of a switch can be decomposed in "mini switches", called switchlets, which can be individually allocated. The functionality of a switch is encapsulated in an open control interface. Therefore, a fine grained control of the switch resources can be achieved. Although the switchlet technique was originally implemented using ATM technology, it is extensive to other network technologies.

A set of switchlets on different switches can be combined to form a virtual network. Each virtual network can potentially use different control and management mechanisms, which are collectively called a control architecture. Therefore, different control architectures can be operational on the same physical network at the same time.

In switchlet networks, new control architectures can be introduced into a network without disrupting existing services and applications. Control architectures can be created on demand to allow the dynamic establishment of virtual private networks. The action of creating switchlets and combine them into VPNs can be automated, allowing the implementation of an open network, in which 'any user' can construct a network, and become a service provider.

III. THE RESOURCE ALLOCATION PROBLEM

To establish a virtual private network, resources need to be allocated along the network. The allocation decision can be done either in a centralized or in a decentralized way. In a decentralized mode, either one of the end points of a VPN carries out the resource allocation. If on one hand, the decentralized scheme may present low signalling overhead. On the other hand, it may lead to non-optimal allocation of resources, In a centralized scheme, however, optimal solutions can be achieved.

To minimize the cost of furnishing service is the goal of every network provider. To do so, in the centralized scheme, a traffic management agent should collect, from time to time,

requests for VPN establishment, and decide which resources should be allocated to a VPN. In other words, it should determine which routes the flow of each VPN should follow. Such kind of problem falls into the category of the multicommodity flow type of problem.

Before showing the multicommodity flow solution, the node splitting transformation and an approach to find the *k* shortest paths between a pair of nodes are introduced.

A. The Node Splitting Transformation

In networks where switching elements are monolithic signalling blocks, bandwidth is the major resource to be allocated. However, in switchlet networks, the resources of a switch can be decomposed into several logical switchlets. Moreover, the input and the output bandwidth of a switch is shared among all switchlets of a switch. To represent the allocation of switchlets and to state the network flow problem in a standard "arc flow" form, the node splitting transformation is used.

The node splitting transformation splits each node *i* into two nodes *i"* and *i'*, which correspond to the node input and output, respectively. It replaces each original arc (*i*, *j*) by an arc (*i'*, *j"*) with the same cost and capacity, and adds an arc (*i"*, *i'*) of zero cost and with infinite capacity for each node *i*. Node *i"* receives the node inflow. From node *i'*, the output of node *i* departures. Arc (*i"*, *i'*) carries flow from the input to the output. Figure 1 illustrates the node splitting transformation.

Note that there is a one-to-one correspondence between flows in the original network and flows in the transformed network. Besides that, flows in both networks have the same cost.

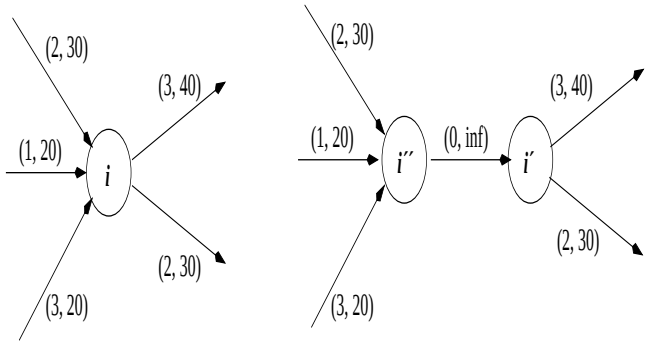

Fig. 1: The Node Splitting Transformation.

B. Finding the k Shortest Paths

A multicommodity flow solution takes into account all possible paths between a pair of nodes which leads to computational unfeasible solutions, except for small networks. To reduce the computational complexity from exponential to linear, at most *k* shortest paths between a pair of nodes are used in the multicommodity flow problem. Only those paths which satisfy the QoS requirements of a VPN are taken into account.

A generalization of the Dijkstra algorithm was used to find the *k* shortest paths between a pair of nodes [10]. In Dijkstra's classical algorithm, nodes are inserted into a heap with the distance to the source node used as a key. At each step, the minimum distance node is extracted from the heap and the distances to its neighbors are updated. Each node is visited only once. When the heap is empty the distances from the source node to all the other nodes of the network are determined. In the generalized Dijkstra algorithm, nodes may be visited more than once, since a node may be used in more than one path. An "element" is created each time a node is visited, and it is tied to the node. Each element has a distance to the source node. Elements are inserted into a heap with their distances as keys. At each step, the minimum distance element is extracted. The time complexity of the generalized Dijkstra algorithm is *O(kmn)*, and the total memory usage complexity is *O(km+kn)*, where *k* is the number of shortest paths, *m* the number of links, and *n* the number of nodes.

C. Resource Allocation via Multicommodity Flow Formulation

In a multicommodity flow problem, each commodity corresponds to a VPN. The problem is formulated as a problem of minimizing network costs. Such formulation can be easily changed to a problem of maximizing revenue if a revenue value is associated to each commodity.

Let *G* be a directed network with *N* nodes, *A* arcs and *K* commodities. *P(k)* is a set which contains all source-destination paths, for $k \in K$. The cost of a path is denoted c_p. The required amount of flow for commodity *k* is denoted by q^k. The total cost of assigning commodity *k* to the p^{th} path, for $p \in P(k)$ is, thus, $q^k c_p$. The revenue of commodity *k* is denoted by w_k.

The *integer* multicommodity flow problem can be formulated as follows:

$$\textit{Maximize} \quad \sum_{k \in K} \sum_{p \in P(k)} c_p q^k y_p^k$$

$$\textit{Subject to} \quad \sum_{k \in K} \sum_{p \in P(k)} q^k y_p^k \delta_a^p \le d_a, \forall a \in A \qquad , (1)$$

$$\sum_{p \in P(k)} y_p^k \le 1, \forall k \in K \qquad , (2)$$

$$y_p^k \in \{0, 1\}, \forall p \in P(k), \forall k \in K \qquad . (3)$$

The value of the decision variable y_p^k is 1 if the commodity k is assigned to p^{th} path. d_a, $a \in A$, is the a^{th} arc capacities. The value of δ_a^p, $a \in A$, $p \in P(k)$, $k \in K$, is 1 if the p^{th} path contains the a^{th} arc. Otherwise δ_a^p equals zero.

The objective function is the sum of the cost of the paths, c_p, multiplied by the required flow q^k of the k^{th} commodity. Constraint (1) establishes that for each arc a the sum of the flows of all commodities using it cannot exceed its capacity d_a. Constraint (2) and (3) state that the flow of a commodity should be assigned to a unique path.

To solve efficiently the integer multicommodity flow problem, an NP-hard problem, a branch-and-bound algorithm is used. A branch-and-bound algorithm analyzes the solution space as a tree structure where each node corresponds to a decision variable in the problem, each branch is an assignment of a value to the decision variable, and leaves are possible solutions of the problem.

Brand-and-bound algorithms reduce the number of nodes to be searched by pruning the search tree. This is done by calculating a bound for each node of the tree and by comparing it to the best solution. When calculating the bound, the values of the already assigned variables are used. The method assigns a path to a commodity at each step, and then tests the lower bound of the decision variable assignment. If the lower bound is greater or equal to the best solution found so far, the search tree is pruned, otherwise the algorithm continues to the next step. The lower bound is calculated using the column generation method that consists in generating columns to solve the linear multicommodity problem when necessary. The algorithm searches the solution space in order to find an optimal solution. It maintains the best solution found at each step of the search.

IV. THE EFFECTIVENESS OF THE PROPOSED METHOD

To assess the feasibility of a real-time implementation of the proposed method, a set of networks was generated by varying the number of nodes, links, and commodities. Table 1 describes the topology of these networks.

ILOG CPLEX Mixed Integer Optimizer, version 6.5, was used to solve the multicommodity flow problem. Software pieces were developed using C ANSI and were executed in a Pentium III machine running LINUX, kernel 2.2.12-20, with 450 MHz, 512 KB cache, 384 MB of main memory and 72 MB of swap memory.

Solutions using 100, 30 and 10 shortest paths were generated. These solutions were compared to the optimal one. The near optimal solutions obtained when the number of paths were restricted deviate at most 2% from the optimal value. Checking the accuracy of non-optimal results was possible only up to the network with label 11. The exact solution of networks 12 to 18 could not be derived due to memory limitation. Results for these networks are reported in order to have an estimate of the execution time.

The accuracy of the near-optimum solutions did not significantly differ when 100, 30 or 10 paths were used. As can be seen, the execution time of the solutions with 10 paths are considerably lower that the ones with 100 paths. Moreover, data support that the proposed method is adequate for real time implementation.

V. CONCLUSIONS

In this paper, it was introduced a new approach for resource allocation in switchlet networks. The proposed method reduces the complexity of a multicommodity flow solution by considering only a subset of the paths which support the desired Quality of Service. Results were within 2% of the optimal one, and the procedure is amenable to real time implementation. The method introduced here is not scalable to large networks, such as the Internet. It is currently under investigation the partitionning of large networks so that the proposed analysis can be carried out in large networks scenarios.

REFERENCES

[1] A. T. Campbell, H. G. De Meer, M. E. Kounavis, J. Vicente K. Miki and D. A. Villela. A Survey of Programmable Networks. Computer Communication Review (CCR), 29(2):7-23, April, 1999.

[2] A. Lazar. Programming Telecommunication Networks. IEEE Networks, 11(5):8-18, September/October, 1997.

[3] J. E. van der Merwe and I. M. Leslie. Switchlets and Dynamic Virtual ATM Networks. Proc Integrated Network Management V, pages 355-368, May, 1997.

[4] R. Isaacs. Lightweight, Dynamic and Programmable Virtual Private Networks. in Proc of OPENARCH, 2000.

[5] J. E. van der Merwe and I. M. Leslie. Service-Specific Control Architecture for ATM. IEEE Journal, 16(3):424-436, April, 1998.

[6] J-P. Redlich, M. Suzuki and S. Weinstein. Virtual Networks in the Internet. Seconde IEEE International Conference on Open Architecture and Network Programming, pages 108-114, 1999.

[7] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. Networking Flow - Theory, Algorithms and Application. Prentice Hall, 1993.

[8] M. Sig and M. Sigurd. MultiCommodity Flow with Integer Solutions. Technical report, Department of Computer Science, University of Copenhagen, May 1999.

[9] T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms. McGraw-Hill Book Company, 1990.

[10] E.Q.V. Martins, M.M.B. Pascoal, J.L.E. Santos. The k shortest paths problem, Tech Rep, Department of Mathematics, University of Coimbra, Portugal, June 1998.

[11] D. Mitra and K. G. Ramakrishnan. A Case Study of Multiservice, Multipriority Traffic Engineering Design of Data Networks. Proc. IEEE, GLOBECOM99, pages 1071-1083, 1999.

Table 1: DESCRIPTION OF NETWORK TOPOLOGIES IN THE STUDY.

Problem	Nodes	Links	Commodities	K = 100	K = 30	K = 10
1	8	25	21	00:00:02.03	00:00:01.79	00:00:01.72
2	8	25	33	00:00:03.18	00:00:02.73	00:00:02.65
3	10	15	28	00:00:02.31	00:00:02.76	00:00:02.28
4	10	15	41	00:00:03.34	00:00:03.36	00:00:03.30
5	10	45	28	00:00:02.85	00:00:02.38	00:00:02.26
6	10	45	41	00:00:04.01	00:00:03.45	00:00:03.31
7	12	50	65	00:00:06.41	00:00:05.42	00:00:05.18
8	20	30	42	00:00:04.65	00:00:03.73	00:00:03.45
9	20	30	90	00:00:12.74	00:00:08.30	00:00:07.36
10	30	40	43	00:00:05.80	00:00:03.90	00:00:03.50
11	30	40	97	00:00:17.98	00:00:09.53	00:00:08.04
12	30	43	43	00:00:06.22	00:00:03.90	00:00:03.53
13	30	43	97	00:00:18.71	00:00:09.88	00:00:08.18
14	30	45	97	00:00:19.07	00:00:09.35	00:00:07.96
15	30	50	97	00:00:15.61	00:00:09.14	00:00:07.93
16	30	100	43	00:00:06.15	00:00:03.96	00:00:03.53
17	30	300	43	00:00:07.35	00:00:04.49	00:00:03.72
18	30	300	97	00:00:15.63	00:00:09.62	00:00:08.11