Minimizing Switching Overhead in Central Offices

Carl Nuzman, *Lucent Technologies, Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974, USA*
Nachi Nithi, *Lucent Technologies, Bell Labs, 600 Mountain Ave., Murray Hill, NJ 07974, USA*

March 5, 2004

Keywords: network topology, graph partitions, optimization

1. Summary

In a generic communication network, switching equipment in a central office is used to route information flows that pass between external trunks. In practice, the number of trunks and the aggregate traffic load are higher than the capacity of any single switch, and thus the central office must contain multiple switches connected to each other by internal trunks.

If the traffic patterns are dynamic and unpredictable, multiple switches can be arranged into non-blocking configuration capable of handling arbitrary permutations of demands. In many cases however, the aggregate flows between external trunks are static, or are expected to satisfy fairly tight static bounds. In such cases, it is sufficient and much less expensive to design the topology of a central office only to satisfy a given static demand pattern.

Although the physical topology of a wide area network is constrained by geography and rights of way, and is expensive to change, the internal topology of a central office may be designed relatively freely and inexpensively.

In this paper, we discuss optimization problems related to the design of internal office topologies, and some algorithms for solving them.

The central office configuration problem may have a number of optimization objectives. These may include capital expenses, operating expenses, and switching load. In this paper we focus on the problem of minimizing switching load, that is, the total amount of traffic observed by all switches in the office. This load consists of a fundamental load, consisting of the set of flows between external trunks, and an overhead, which is twice the sum of all traffic flowing between local switches on internal trunks. In practice, reducing the overhead naturally tends to reduce the cost of equipment required.

A very simple form of this problem is equivalent to the min k-cut problem. Suppose that there are N external trunks, let $t_{i,j} \geq 0$ represent the aggregate demand between trunks i and j, and suppose that we know that k switches should be used. We wish to assign trunks having much traffic in common onto the same switch, in order to minimize the flow between switches. We identify the trunks with the nodes of a fully connected graph, where a cost $t_{i,j}$ is associated with edge (i, j). The min k-cut problem seeks to cut the graph into k parts, in such a way that the total weight of the edges in the cut is minimized.

The min k-cut problem is NP-hard with respect to k and N, but if k is fixed, it is polynomial in N [1]. There are also fairly simple 2-approximations to the problem for fixed k [2].

This model leaves out switch constraints, which are the reason that multiple switches are needed in the first place. In the simplest useful constraint is to limit the number of trunks that can be assigned to any switch, or equivalently to limit the cardinality of the connected components of the k-cut. An efficient "swap" heuristic for this problem was developed in [3].

For several reasons, the constraints on allowable switch configurations can be much more complex than just a limit on the number of trunks.

1. In addition to the external trunks, the switch must have ports available for the internal trunks used to carry traffic between local switches.
2. There are typically many different kinds of trunks, distinguished by capacity and protocol.
3. The switches may have a hierarchical structure: for example, trunks plug into various physical interface cards, which in turn plug into various processor cards which plug into slots.

We discuss a general class of switch constraints that meets the above considerations, and show that for such constraints, the entire problem can be formulated as an integer program. If the switch constraints are reasonably simple and the problem scale is not too large, such programs can be solved to optimality in reasonable time. When the number of trunks and switches is large, or when the switch constraints cannot be described by simple linear constraints, swap-based heuristics provide an effective alternative to the integer programming approach. The sub-problem of determining whether or not a given set of trunks may be assigned to a switch is used many times within the heuristic approach, and hence it is crucial to be able to make this decision quickly.

References

[1] Goldschmidt, O. and Hochbaum, D. S., "A Polynomial Algorithm for the Kappa-Cut Problem for Fixed Kappa," *Mathematics of Operations Research*, pp. 24–37, 1994.

[2] Saran, H. and Vazirani, V. V., "Finding k Cuts within Twice the Optimal," *SIAM J. Computing*, pp. 101-108, 1995.

[3] Kernighan, B. W. and Lin, S., "An Efficient Heuristic for Partitioning Graphs," *Bell Systems Tech. J.*, pp. 291-308, 1970.