Appia: formalization of its topology assignment droblem

Li-Shiuan Peh

Storage Systems Program,
Computer Systems Laboratory,
Hewlett-Packard Laboratories, Palo Alto, CA

18th September 1998

This paper documents the formalization of Appia's problem onto the integer program framework, and the multi-commodity problem. We also present our thoughts on possible future work in formalization.

1 Introduction

The topology assignment problem faced by Appia is as follows.

Given data flows between hosts and devices (through streams bound to stores which are bound to devices by the Forum solver), Appia's goal is to design an interconnect fabric between these hosts and devices which supports these data flows, whilst achieving objectives set out by the user.

For instance, an objective may be to find the minimal cost fabric which supports these data flows; or to find the fabric which results in minimal latency for these flows. Yet another possible objective could be to provide a fabric which achieves some availability goals for critical hosts and devices.

A fabric consists of elements joined together. These elements are switches, hubs, multiplexers, fibres and scsi cables which links hosts and devices together. Each of these elements have different models, each with different bandwidth, cost and other attributes. Appia thus have to take these into account in its design of the fabric.

The movitation for formalizing Appia's problem is to allow us to tap into the rich literature of theoretically-proven algorithms, and to make it easier for future extensions of the problem, such as reliability issues.

2 Formalization as graph problem

We formulate the topology assignment problem as a graph problem, as we see the natural parallels.

Hence, the fabric is a graph with nodes and links. Nodes are either terminal or internal; with hosts and devices being terminal nodes and fabric elements such as hubs, switches and multiplexers as internal ones. Links are fabric elements such as fibres and scsi cables.

The flows required between the terminal nodes are specified in the requirements to Appia, with their respective bandwidths.

Different types of internal nodes are also provided, with their cost, capacity and maximum degree (number of ports on the fabric element) detailed.

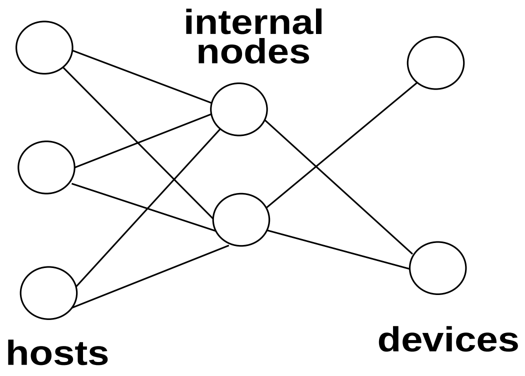
Similarly, different types of links are specified, with their cost and capacity detailed.

Hence, if the objective is to minimize cost, given attributes of flows and nodes and links of different types, Appia's goal is to produce a minimal-cost graph containing nodes and links of the given types and satisfying the given flows.

3 Formalization as an integer program

Although the general Appia problem requires the construction of a graph with the given types of nodes and links, to formulate it into an integer program, we choose to first generate a possible graph, and let the integer program prunes out unnecessary nodes and edges.

Hence, first, we guess the maximum possible number of internal nodes. This can be the higher of the number of hosts or devices. We then generate a graph which is completely connected between the hosts and internal nodes, and the internal nodes and devices.



Nodes are henceforth subscripted with i, and links with ij.

Decision variables

x_{ij} – link ij exists or not

h_i – a hub is installed at node i

s_i – a switch is installed at node i

f_{ij}^{k} – flow of commodity k over edge ij

Data variables

cp_{ij} – capacity of edge ij

c_{ij} – cost of edge ij

cp_h – capacity of a hub

cp_s – capacity of a switch

c_h – cost of a hub

c_s – cost of a switch

d_h – *degree* of a hub

d_s – *degree* of a switch

r_i^k – requirements of commodity k

Constraints

$$\sum_j f_{ij}^k - \sum_j f_{ji}^k = 0, \text{ for internal nodes} \qquad \text{(flow conservation constraint)}$$

$$= r_i^k, \text{ for terminal nodes}$$

$$\sum_j \sum_k f_{ji}^k \le cp_h h_i + cp_s s_i \qquad \text{(node capacity constraint)}$$

$$\sum_k f_{ij}^k \le cp_{ij} x_{ij} \qquad \text{(link capacity constraint)}$$

$$\sum_j x_{ij} + \sum_j x_{ji} \le d_h h_i + d_s s_i, \quad \text{for internal nodes} \qquad \text{(internal nodes' degree constraint)}$$

$$\sum_j x_{ij} + \sum_j x_{ji} \le d_i, \quad \text{for terminal nodes} \qquad \text{(terminal nodes' degree constraint)}$$

$$0 \le (h_i + s_i) \le 1 \qquad \text{(logical constraint)}$$

Objective

$$\text{Minimise cost } = \sum_i \sum_j x_{ij} c_{ij} + \sum_i (h_i c_h + s_i c_s)$$

This formulated integer program can be coded into an integer program solver such as CPLEX (Cplex), through the use of an interface language such as AMPL (Ampl), (Ampl, 1993). Even if the integer program proved too slow for practical usage, it will serve as a good benchmark to compare our heuristics against.

4 Formalization as a multi-commodity flows problem

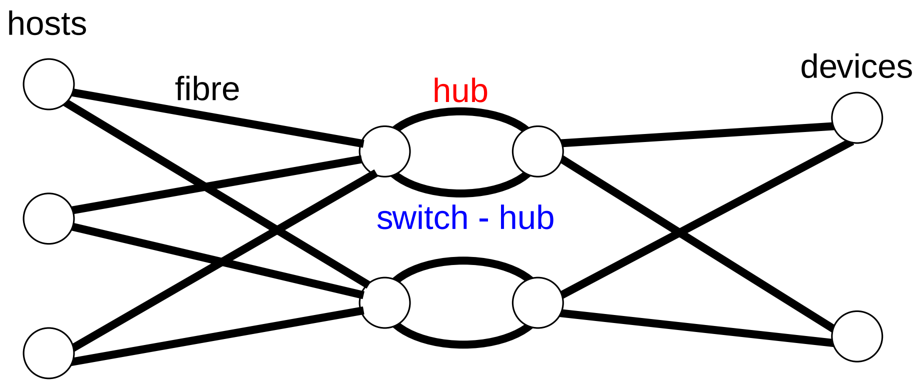
We also attempted to map Appia's problem onto the well-known theoretical problem of multi-commodity flows.

An introduction to the multi-commodity flows problem can be culled from (Bazaraa, Jarvis and Sherali, 1990). (Boesch, 1976) also gives a good survey to the wide array of existing graph problems, and a few interesting papers on multicommodity flows.

Formalization A

Multicommodity flows problem only takes in account links, not nodes. We hence need to think of a way to map the internal nodes of hubs / switches onto links. One way is to create a link for each internal node in the previous integer program formalization, with this link representing a hub or a switch, depending on the load on it. Hence, its cost function is a step cost function which is the cost of a hub when load is <= hub's capacity, and the cost of a switch when load is between a hub's capacity and a switch's. However, there isn't a variant of multicommodity flows which handles step cost functions.

We hence use another way of mapping hubs and switches onto links. This is proposed by Julie Ward, HPL. Two links are used instead of one. One link represents a hub, and the other represents a switch - a hub. Hence, when only the hub link is taken, a hub is installed, whilst when both links are taken, a switch is used. Note that this requires the algorithm used NOT to allow for the case whereby the switch-hub link is used solely.



Each host is the supplier of its own bandwidth commodity. That is, if there are 3 streams generated by the host, each of 30 MB / s each, the host then supplies 90 units of its bandwidth commodity.

Devices on the other hand are the sinks of bandwidth commodities. If a device has streams coming from hosts A and B, it will demand some units of bandwidth commodity A and some units of bandwidth commodity B.

Each of the links have associated with it a cost = cost of a fibre, or the cost of a hub, or the cost of a switch - cost of a hub. Notice that this cost function is a Fixed Charge cost function, which is a special case of a concave cost function, whereby the cost per unit decreases as the load increases.

Each link also has associated with it a capacity over all commodities. Hence, a link representing a fibre has the capacity of a fibre (say 100 units for 100MB / s), and hence, no more than 100 units of all types of commodities can go through it. The link representing a hub has the capacity of a hub and the link representing a switch has the capacity of a switch - that of a hub. So if both links are used, the capacity of a switch is available.

For degree, the hub link has a degree constraint = number of ports on the hub, and the switch link has a degree constraint = number of ports on the switch. Note that this assumes that a switch has more ports than a hub.

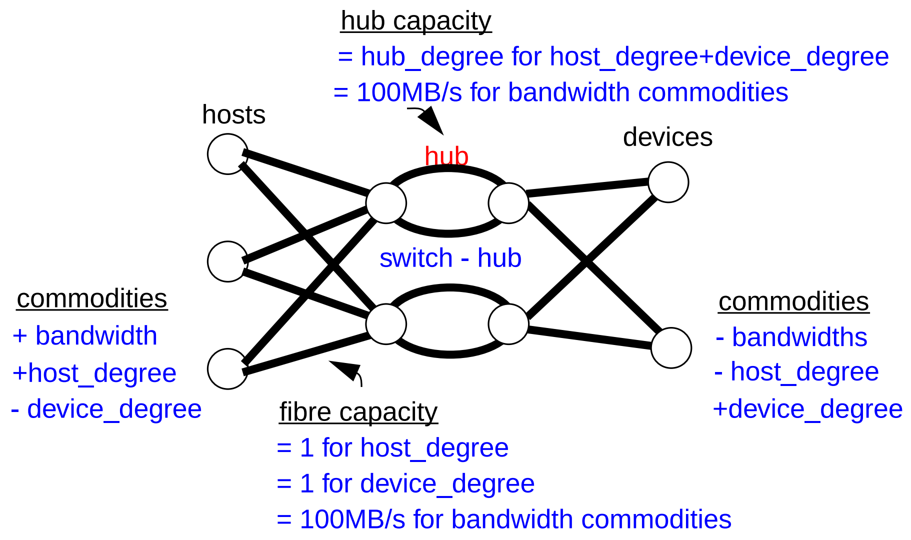
This formalization captures Appia's problem, but requires a variant of multicommodity flows which is capacitated, have links with concave-cost functions, and is degree-constrained.

Unfortunately, no such variant has been proposed in literature yet.

Formalization B

To remove the degree constraints, we came up with another way of mapping Appia's problem onto multicommodity flows. The gist of this idea was proposed by Guillermo Alvarez, HPL.

Instead of having degree-constraints, new degree commodities are added. These commodities correspond to degree, i.e., the number of ports on hosts, devices and interconnects.



Hosts, in this formalization, supply degree commodities in addition to bandwidth commodities. So, they supply N units of host degree commodity corresponding to the number of ports on it, and demands N units of device degree commodity corresponding to the number of ports on it. This can be explained as it being able to send and receive N fibres through its N ports.

Devices, similarly, supplies M units of device degree commodity corresponding to its number of portsm and demands M units of host degree commodity. Also, they are the sinks of bandwidth commodities of the hosts.

For capacities, fibres have a capacity of 1 for host degree commodity, and 1 for device degree commodity since it's bi-directional. Hubs have a capacity equal to the number of ports it has for host and device degree commodities, and (Switch-Hub) have a capacity equal to the difference between the number of ports a switch has and the number of ports on a hub. The bandwidth capacities follow that of the previous formalization.

This formalization is again able to capture Appia's topology assignment problem.

However, it requires a multi-capacity multicommodity flows problem, whereby there are multiple capacities on each link, associated with groups of commodities, instead of aggregate commodities. Again, there is no literature on this variant.

Relevant Literature

The variant of multi-commodity flows problem which Appia's Formalization A can map into is a concave-cost, capacitated, degree-constrained multi-commodity flows. For Formalization B, the variant is a concave-cost, multi-capacitated multi-commodity flows. If we choose to represent the choice of hubs and switches as a single link, we need a multi-commodity flows variant which handles step-cost functions too, or resort to an approximation of the step-cost function with a concave cost function.

There exists algorithms which handles concave-cost multi-commodity flows, such as Yaged's algortihm (Yaged, 1971). A survey of the algorithms for this variant of multi-commodity flows can be found in (Ward, 1998).

For multi-commodity flows problem with capacity constraints, this paper by Erickson, Monma and Veinott (1987) proposes a way to transform capacitated flows into uncapacitated ones, through the modifications of the requirements variable.

Degree-constrained multi-commodity flows are yet to be found in literature.

5 Future Work

5.1 Multi-commodity Flows

5.1.1 K-Shortest Paths Tweak to Yaged's Algorithm

Yaged's algorithm (Yaged, 1991) currently uses the all-pairs shortest paths algorithm (Cormen, Leiserson and Rivest, 1997) to obtain the loads on links, given the derivatives of costs. We propose to use the k-shortest paths algorithm (Eppstein,1997) instead, and prune through these k shortest paths in order, until an all-pairs shortest path which satisfies degree constraints is arrived.

It needs to be further investigated and analyzed if this proposal will work for Formalization A.

5.1.2 Multi-capacities

Formalization B requires capacities which are associated with groups of commodities, and not just with aggregate of all commodities. This variant of multi-commodity flows has not yet been proposed in literature. If we are able to solve this variant, Appia's problem will be solved.

5.2 Constraint Programming

The main disadvantage of using integer programming is the time it takes to find the optimal solution. Constraint programming may be a viable alternative, since it prunes the search space more efficiently.

5.3 Steiner Trees

Steiner trees may be another well-known problem which we can map Appia problem to. A good starting point for exploring steiner trees is (Hwang,Richards and Winter, 1992) and (Winter, 1987). Analysis and algorithms for degree-constrained steiner treees are available, but they do not fully address Appia's problem either. It may be fruitful to pursue this.

5.4 Degree-constrained multicasting

This is an idea I came upon, but had no time to pursue. The idea is that it may be possible to uncover similarities between the multicast routing problem and Appia's problem. A paper detailing the degree-constrained multicast problem is (Bauer and Varma, 1995).

5.5 Multi-layer hubs and switches

Currently, the formalization allows only for single layer of internal nodes between hosts and devices. This need not be true in real life, where we can have cascaded hubs and more importantly, switch fabrics. There is hence a need to extend our formalization to account for multiple layers of internal nodes.

5.6 Availability

An important issue in the practical problem is reliability. There are a few dimensions to this. Firstly, the fabric ought to be able to absorb failures of its own components. Secondly, if the terminal nodes of the fabric fails (its hosts or devices), flows may be redirected to another host, and the fabric may need to handle that.

Preliminary thoughts on this is the use of k-paths in network flows literature (Gross and Saccoman, 1997), i.e., assuring that at least k distinct paths exist between pairs of terminal nodes, and thus, should any of these links fail, connectivity is still assured.

6 Acknowledgements

I will like to thank my mentor John Wilkes for giving me guidance through this summer internship. Thanks too to Julie Ward and Guillermo Alvarez of HP Laboratories for helping me along with this formalization.

References

(Ampl) Ampl Documentation.

(Ampl, 1993) Ampl, A Modeling Language for Mathematical Programming, 1993.

(Bauer and Varma, 1995) Fred Bauer and Anujan Varma, “Degree-constrained Multicasting in Point-to-Point Networks”, IEEE, 1995.

(Bazaraa, Jarvis and Sherali, 1990) Mokhtar S. Bazaraa, John J. Jarvis and Hanif D. Sherali, “Linear programming and Network flows”, Chapter 12, John Wiley and Sons, 1990.

(Boesch, 1976) Francis T. Boesch, “Large-Scale Networks: Theory and Design”, IEEE Press, 1976.

(Cplex) The Cplex manual .

(Eppstein, 1997) David Eppstein, “Finding the k shortest paths”, Technical Report, UC Irvine, http://www.ics.uci.edu/~eppstein/, 1997.

(Gross and Saccoman, 1997) D. Gross and J. T. Saccoman, “Uniformly Optimally Reliable Graphs”, Networks 31, pp. 217-225, 1998.

(Hwang, Richards and Winter, 1992) F. K. Hwang, D. S. Richards, P. Winter, “The Steiner Tree Problem”, Annals of Discrete Mathematics 53, North-Holland, 1992.

(Ward, 1998) Julie Ward, “Minimum-aggregate-concave-cost multicommodity flows in strong-series-parallel-networks”, manuscript.

(Winter, 1987) Pawel Winter, “Steiner Problems in Networks: A Survey”, Networks, Vol. 17, pg. 129-167.

(Yaged, 1971) B. Yaged, Jr., “Minimum Cost Routing for Static Network Models”, Networks, Vol. 1, pp. 139-172, 1971.