Load-based Clustering for Hierarchical Routing

Beat Liver
IBM Research Division
Zurich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon, Switzerland
bli@zurich.ibm.com

Abstract

A novel abstraction of the available bandwidth in a communication network is presented based on the blocking island abstraction (Frei & Faltings 1997). Path computing algorithms using this abstraction are presented for non-hierarchical and hierarchical networks. For the latter case, a composeable abstraction of subnetworks is proposed.

Introduction

Abstractions are used to reduce problem complexity and facilitate man-machine interaction (Choueiry 1994). An abstraction for the load of telecommunication networks are blocking islands and blocking island hierarchies (Frei & Faltings 1997; 1998). This paper proposes blocking island contour maps, which generalize blocking island hierarchies.[1] In addition, the application of blocking island contour maps for intra- and inter-domain routing of constant bit-rate connections is discussed. Generalizing the result to hierarchies with more levels is straight forward. Hierarchical routing is important, because scalability (ATM 1996) and multiple network operators demand hierarchies (Galis *et al.* 1996).

Problem

Intra- and inter-domain path computation are defined, where the former corresponds to allocating circuits in a communication network that is not hierarchically structured.

Intra-Domain Case

A communication network is modeled as a *network graph* $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}, \mathcal{V}_\mathcal{A} \rangle$. $\mathcal{G}$ is an undirected multi-graph without loops, i.e., edges whose end-points are the same vertices. $\mathcal{V}$ and $\mathcal{E}$ denote the set of *vertices* and *edges*, respectively. $\mathcal{V}_\mathcal{A}$ denotes the set of access nodes, where an *access node* is a network node at which network users are connected to the network, i.e., $\mathcal{V}_\mathcal{A} \subseteq \mathcal{V}$. An example network graph is depicted in Fig. 1, where access nodes are depicted as filled circles. The vertices

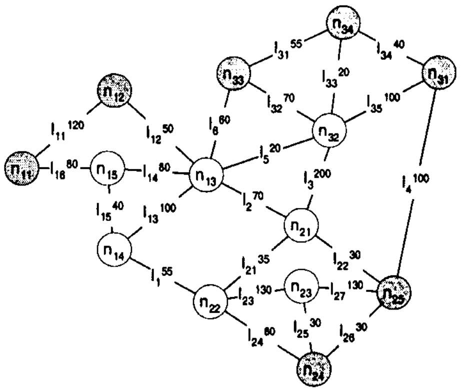

Figure 1: Example network graph.

and edges correspond to switches (e.g., ATM and SDH) and bidirectional point-to-point communication media and services. Each edge e is characterized by its ***bandwidth capacity*** β^*_e, the (currently) ***available bandwidth*** $\bar{\beta}_e$, and other quality of service properties, like delay, loss, and so on. In Fig. 1, superscripts of edge labels denote available bandwidth.

A *demand* describes the requirement on the network for transferring data between a pair of access nodes. It is formally defined by Equation 1,

$$d_k \overset{\text{def}}{=} \langle a, z, \beta_{d_k} \rangle, \text{where } a, z \in \mathcal{V}_\mathcal{A} \qquad (1)$$

where β_{d_k} denote the required amount of bandwidth. It is assumed that β_{d_k} is a constant number, i.e., *constant bit-rate* demands are modeled.

A network $\mathcal{G}$ satisfies a set of demands $\mathcal{D}$ by allocating

[1]Patents are pending for the algorithms presented in this paper.

a *circuit* for each demand. A circuit is a simple path p_i in the network graph that satisfies the bandwidth and other quality of service requirements.

Inter-Domain Case

A *domain* is a subnetwork with domain access nodes, called *border nodes.* In the inter-domain network graph, inter-domain links connect border nodes of different domains. Fig. 2 depicts the inter-domain net-

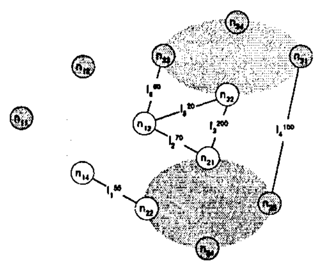

Figure 2: Inter-domain network graph.

work derived from Fig. 1 assuming that the first digit of two-digit subscripts indicates the domain.

Path computation at the inter-domain level has to take into account the traversal costs of the domains for computing globally optimal paths. These traversal costs can either be requested during the computation of a particular path (Karali *et al.* 1998; Chatzaki & Sartzetakis 1998) or be determined independently of a route to be computed (Karali *et al.* 1998; ATM 1996). For a comparison, see Section "Discussion". I present a novel abstraction, because this approach can be expected to be more efficient if an appropriate abstraction can be found.

The BICM

Blocking Island Contour Map (BICM), a novel abstraction of network graphs, is introduced to address the hierarchical path computation problem. It is based on the Blocking Island concept (Frei & Faltings 1998) and generalizes the Blocking Island Hierarchy (Frei & Faltings 1998), so that the necessary definitions from (Frei & Faltings 1997; 1998) are recalled first. Thereafter, the BICM itself and its construction are presented.

Definitions

Blocking Island (Frei). A β *Blocking Island (β-BI)* for a vertex v is defined as the set of all vertices of the network graph that are reachable from v using edges with at least β available bandwidth (Frei & Faltings 1997). It also includes the edges connecting its vertices. Note that that a β-BI might contain edges with less than β available bandwidth. The usefulness of blocking islands results from the fact that there exists only one β-BI for each vertex contained in it, so that blocking islands define equivalence classes over vertices.

Blocking Island Graph (Frei). A *β-Blocking Island Graph (β-BIG)* clusters each β-BI into a single *abstract vertex* (Frei & Faltings 1997). Edges between two different β-BIs are clustered into a single ***abstract edge*** between the two β-BIs. These abstract edges represent critical edges, since their available bandwidth is lower than β. The available capacity of an abstract edge is the maximum of the available bandwidth of the edges clustered by this abstract edge. If the two end-points of a demand are not in the same vertex of a β-BIG, there exists no route for this demand.

Blocking Island Hierarchy(Frei). A *Blocking Island Hierarchy (BIH)* is defined as a hierarchy of β-BIGs, where $\mathcal{B}$ denotes the set of ordered bandwidth levels (Frei & Faltings 1997). That is $\mathcal{B} = \{\beta_1, \ldots, \beta_b\}$ and $0 < \beta_1 < \ldots < \beta_b$.[2] The highest and lowest level are the 0-BIG and the β_b-BIG, respectively. The latter is equivalent to the network graph, if $\forall e \in \mathcal{E}, \beta(e) < \beta_b$.

BICM Data Structure

Visualization. A BICM can be visualized as a contour map, where contour lines enclose the network nodes belonging to a blocking island. A contour line is drawn for each abstract node that contains at least one network graph node or two abstract nodes. The contour map of the BICM of the network graph Fig. 1 is depicted in Fig. 3.

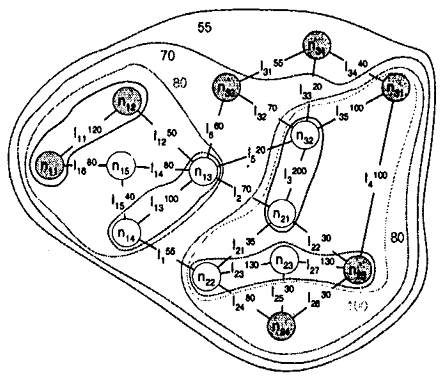

Figure 3: Contour map of overall network graph.

[2]For explanatory purposes, the labeling of the levels is different from (Frei & Faltings 1997).

Approximate β Blocking Island (ABI). An ABI is defined like a β-BI, except that edges with less available capacity than β are not included in the ANH-subtree representing the ABI. These edges are irrelevant with respect to the important properties of a β-BI. Furthermore, an ABI behaves like a β-BI with respect to visualization and path computation (see ABI Graphs). For instance, l_{12} does not belong to the subtree with the root node N_1^{80} (Fig. 4), but it is included in the visualization of this ABI (Fig. 3).

Abstract Node Hierarchy (ANH). In contrast to BIH (Frei & Faltings 1997), each subtree of an ANH represents an ABI. An example ANH is shown in Fig. 4. Edges and abstract nodes are depicted as rectangles

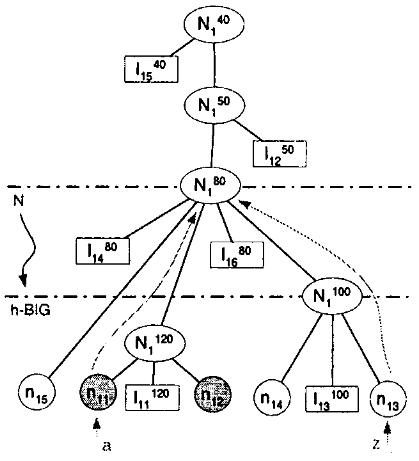

Figure 4: Abstract node tree for domain labeled "1".

and ellipses, respectively. An *abstract node* is defined such that it contains:

- a set of network links that have all the same available capacity b and form a connected component;
- all endpoints of above network links, of which the available bandwidth of the link associated with an endpoint is larger than the one of any other link associated with the endpoint; and
- all abstract nodes that contain recursively an endpoint of the above network links.

The value b defines the *capacity* $\beta(N)$ of the abstract node N, which is denoted by the superscripts of abstract node labels.

Abstract Link Hierarchy. *Abstract links* are defined like abstract edges. Fig. 5 depicts the hierarchy of abstract links of the node hierarchy given in Fig. 4. L_1^{80} results from the fact that n_{14} and n_{13} are contained

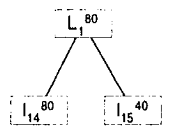

Figure 5: Abstract links for domain labeled "1".

in the same abstract node N_1^{100} and that l_{14} and l_{15} — which are associated with n_{14} and n_{13}, respectively — share the endpoint n_{15}.

ABI Graphs (β-BIG). *ABI Graphs* are β-BIGs due to their definition, which is explained next. The *degree* of an abstract node N is defined as the union of the degrees of the nodes contained in N, where edges with endpoints in N are excluded and edges that share the same endpoint outside N are replaced by an abstract link containing them.[3] For instance, the degrees of N_1^{120} and N_1^{100} are $\{l_{16}, l_{12}\}$ and $\{L_1^{80}, l_{12}\}$, respectively.

The 100-BIG consists of the nodes N_1^{100}, N_1^{120}, and n_{15} (see Fig 4). Assuming network nodes have infinite capacity, N_1^{120} and n_{15} imply nodes at the level $h = 100$ — these nodes are called *implied nodes*, because they are projections of nodes at a higher level than 100. The set of links of a BIG is defined by the links associated with the nodes of the BIG. In our example, it is $\{l_{16}, l_{12}, L_1^{80}\}$. Note that the degree of network and abstract nodes depends on the considered level.

BIH and BICM. In contrast to a BIH, a BICM does not require the specification of a set $\mathcal{B}$. Therefore, a BIH corresponds to a discrete BICM. Such a discretized BICM is computed by using the discretized available capacity $\hat{\beta}()$ instead of the available capacity of the edges for computing the BICM. $\hat{\beta}()$ is defined as

$$\hat{\beta}(e) \overset{\text{def}}{=} f(0, e)$$

$$f(i, e) \overset{\text{def}}{=} \begin{cases} f(i+1, e) & \text{if } \mathcal{B}[i] < \beta(e) \\ \mathcal{B}[i] & \text{else} \end{cases}$$

where $\mathcal{B}$ is represented as an array $\mathcal{B}[]$ and $\mathcal{B}[0] = 0$ and $\forall e \in \mathcal{E}, \beta(e) < \beta_b$.

Computing a BICM

A BICM Ξ for a network graph $\mathcal{G}$ is computed from scratch initially. Then, the BICM Ξ is incrementally

[3]Note that endpoints denote network vertices and abstract nodes.

updated in response to modifications of $\mathcal{G}$. Only the former is presented here due to the limited space.

A BICM Ξ is computed for a network graph $\mathcal{G}$ and a discretisation function $\hat{\beta}()$ — often the identity function — by carrying out the following steps:

1. An ordered set $\mathcal{W}$ is computed by sorting the edges $\mathcal{E}$ in decreasing order according to their available bandwidth $\hat{\beta}(\bar{\beta}_e)$. A set $\mathcal{S}$ is initialized as the empty set.
2. From $\mathcal{W}$, the first edge e is removed and processed:
   (a) If one of the endpoints of e belongs to an abstract node N of the capacity equal to $\hat{\beta}(\bar{\beta}_e)$, e is added to N. If two such nodes exists, they are merged into a single one (called N). Otherwise, an abstract node N that contains e is created.
   (b) N is added to $\mathcal{S}$, if not already contained.
   (c) For all endpoints n of e, the following step is carried out: If n is contained in an abstract node N' with a capacity that is larger than $\hat{\beta}(\bar{\beta}_e)$, then N' is added to N. Otherwise, n itself is added to N.
3. If $\mathcal{W}$ is empty or if e and the first edge of $\mathcal{W}$ have a different amount of available bandwidth, the degree for each abstract node M in $\mathcal{S}$ is computed in the following steps:
   (a) The union $\mathcal{X}$ of the degrees of all nodes contained in M is computed.
   (b) All links of which both endpoints are in M are removed from $\mathcal{X}$.
   (c) $\mathcal{X}$ is partitioned into subsets, of which each contains all links that share an endpoint at the capacity level of M.
   (d) An abstract link L is created for each subset that contains more than one element. L or the single element is added to the degree of M.

   $\mathcal{S}$ is set to the empty set.
4. The algorithm continues with Step 2, if $\mathcal{W}$ is not empty. Otherwise, the algorithms terminates and returns the computed BICM.

The computational complexity of this algorithm is $O(|\mathcal{E}| \log |\mathcal{E}|)$, because the network edges have to be sorted and processed once. The worst-case space complexity is $O(|\mathcal{E}|)$, which results from some single-ring topologies only. Consequently, a BICM is a compact data structure that is efficiently computable.

Path Computation

For explaining the algorithms, a path is computed for a demand $\langle n_{11}, n_{13}, 20\rangle$ for the network graph of Fig. 1.

Route Existence Test

For determining whether a route exists, it is necessary to determine whether the available bandwidth in the ABI that contains both endpoints of the demand d_k is equal or larger than β_{d_k}. This is done by determining the lowest abstract node containing both endpoints by traversing the ANH starting with a and z (as illustrated in Fig. 4). This computation is a very simple and efficient.

Hierarchical Routing

An ABI might encompass a large subnetwork graph, in which case hierarchical path computation can be employed to reduce the search space. The following algorithm uses the BICM:

1. The common abstract node N of the two endpoints of the input demand is determined using above route existence test.
2. N is mapped to the h-BIG that balances the trade-off between subnetwork size and the number of iterations to compute a path at the level of the network graph (as specified). In Fig. 4, h is the minimum over the capacity of the abstract nodes contained in N as immediate children.
3. Within the h-BIG, the shortest abstract path $\mathcal{P}_h$ is computed as described in the next subsection.
4. Steps 2 to 4 are recursively applied for each abstract node on the abstract path $\mathcal{P}_h$ to compute a path at the level of the network graph. Note that an abstract link is also associated and processed with each abstract node, except the last one.

In the example given in Fig. 4, the abstract node N is N_1^{80} after performing the first step. N contains the abstract nodes N_1^{100} and N_1^{120}, so that it is mapped to a 100-BIG. This BIG consists of the nodes n_{15}, N_1^{100} and N_1^{120}. Then the algorithm is recursively applied on N_1^{120} and N_1^{100}. It is important to understand that, in case of implied abstract nodes, the algorithm operates on the implying ones (e.g., N_1^{120}). This reduces the number of mappings and hence the run-time of the algorithm.

Shortest Path

The well known Dijkstra algorithm (Leeuwen 1990) is extended for computing shortest paths with BICMs. In particular, the costs function is extended by a *Node-Traversal Cost (NTC)* $\xi(n)$ and other terms (not discussed here). NTC denotes the costs for traversing a node n. Here, I present two different functions $\xi()$, where one of them is an estimate. Note that it is necessary to define appropriate weights for $\xi(n)$ in general.

Estimated NTC. $\xi()$ is defined to be inverse proportional to the available bandwidth in the domain. If N_q is the root of ANH representing the ABI covering all vertices of a domain q, Equation 2 defines $\xi()$.

$$\xi(N_q) = \frac{1}{\beta(N_q)} \tag{2}$$

In a h-BIG, the such defined $\xi()$ returns different values, because implied nodes have a larger $\beta()$ value than h. This definition balances the network load, because traversing an island with a lot of available capacity is cheap.

ABI NTC. The Estimated NTC does not take into account the incoming and outgoing edge of traversed nodes. In the ABI NTC, this information is taken into account. For this purpose, a BICM Ξ_q is computed for each domain q independently of the other domains and the inter-domain edges. To hide the internal topology of the domain, only the border nodes and their blocking islands are published in the form of an *Abstract BICM (AB)*. Therefore, an AB $\tilde{\Xi}_q$ is a hierarchy of abstract nodes containing only border nodes. The AB of domain "1" is given in Fig. 6. This abstraction reveals

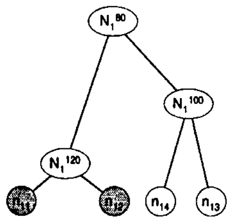

Figure 6: Abstract BICM of Domain "1".

the amount of bandwidth available among the border nodes. Based on the AB, the costs $c(a,z,b)$ for traversing a domain q from border node a to z with bandwidth b can be calculated, e.g., according to Equation 3.

$$\begin{aligned} c(a,z,b) &= \frac{\beta(N_q)+1}{\nu(a,z,b)} \\ \nu(a,z,b) &= \begin{cases} \beta(M) & \text{if } \rho(a,z,b) = M \\ \frac{1}{\infty} & \text{if } \rho(a,z,b) = nil \end{cases} \end{aligned} \tag{3}$$

where N_q denotes the lowest abstract node containing all border nodes of domain q. $\nu(a,z,b)$ returns the capacity of M, where M is the abstract node returned by the above route existence test $\rho()$ applied on the AB (instead of the BICM) of domain q for a and z. Note that $\rho()$ returns *nil*, if there exists no path.

Discussion

A BICM is concerned only with available bandwidth or restrictive costs in general. Therefore, it can act only as a first filter in QoS routing in general.

A blocking island is a property of a network instead of a link. In contrast to (Winter, Busuioc, & Titmus 1995; Magedanz 1996), it is possible to differentiate in load-based pricing between critical and other links. An edge is *critical*, if the edge connects two different blocking islands (that are located in an area in which a demand exists). Hence, heavily loaded edges within a blocking island are uncritical.

In the remainder, the application of Estimated and ABI NTC to the intra- and inter-domain routing problem is discussed. For these applications and pricing, BICM is superior to BIH, because it does not require to specify a priori the precision in the form of a set of bandwidth capacity levels $\mathcal{B}$.

Use of Estimated NTC

The BICM for the whole network given in Fig. 3 shows that there exists not always an ABI (and hence a blocking island) that contains all vertices of a domain and no other vertices. It is the case for domain "1", but not for the two other domains. If this this condition is not satisfied, the Estimated NTC does not reflect correctly the available bandwidth in the network.

Consequently, Estimated NTC is computational efficient but the boundaries of domains and blocking island have to be aligned. This is the case if the hierarchy is determined by a BICM. In case that the network is also structured based on other criteria (e.g., administrative domains), Estimated NTC is inappropriate.

Use of ABI NTC

Fig. 7 depicts the ABs computed for each domain independently. The BICM for the inter-domain network graph can be composed out of the inter-domain links and the ABs of the domains. This computation corresponds to adding links to a network consisting of multiple connected components and updating the BICM. The resulting *inter-domain BICM* is shown in Fig. 8. It corresponds to the BICM of the network graph that is not structured into a hierarchy (Fig. 3). These two BICMs result also into the same end-to-end paths, but the topology of the domains are not revealed for computing the inter-domain BICM. An AB is therefore an

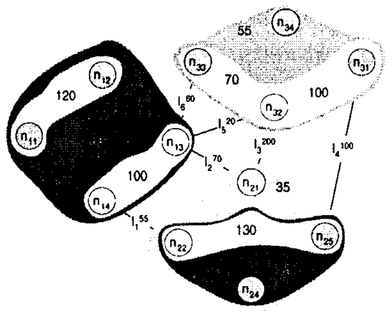

Figure 7: ABs of all domains.

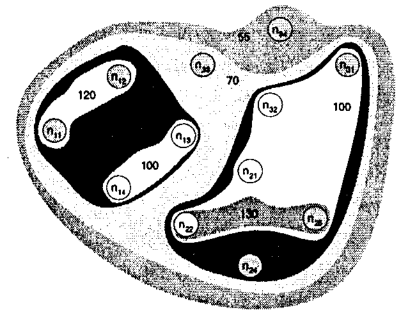

Figure 8: Composition of ABs.

appropriate abstraction of a subnetwork of a hierarchical network.

With respect to a restrictive costs, AB is superior in comparison to the abstraction proposed in (Karali *et al.* 1998): it consumes less space and acts as an initial filter for QoS routing. Furthermore, a overall BICM can be composed from ABs. Clearly, further research is necessary to handle multiple QoS parameters.

Conclusions

Blocking island contour maps generalize the blocking island hierarchy introduced in (Frei & Faltings 1997). The proposed abstraction improves intra-domain QoS routing, where inter-domain QoS routing requires more research. The proposed abstractions and algorithms are implemented in Java, where only a preliminary graphical interface exists at the moment. An example screen shot is given in Fig. 9.

Acknowledgments

I thank Roger Griessen for implementing the graphical interface and the anonymous reviewers, Christian Frei, and Kai Cieliebak for their help to improve this paper.

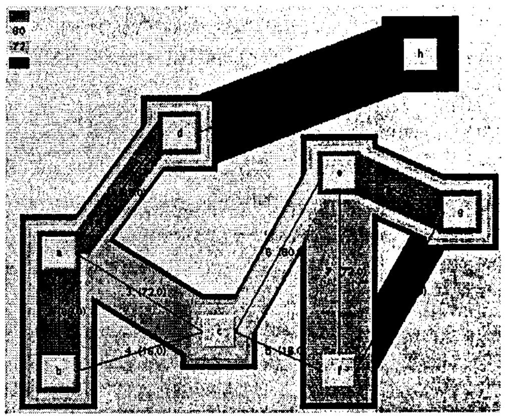

Figure 9: Example screen shoot.

References

The ATM Forum. 1996. *Private Network-Network Interface Specification (P-NNI) Version 1.0.*

Chatzaki, M., and Sartzetakis, S. 1998. Multilevel QoS-policy based routing management architecture appropriate for heterogeneous network environments. In *Proc. Syben'98.*

Choueiry, B. Y. 1994. *Abstraction Methods for Resource Allocation.* Ph.D. Dissertation, Swiss Federal Institute of Technology in Lausanne.

Frei, C., and Faltings, B. 1997. A dynamic hierarchy of intelligent agents for network management. In *Working notes of IJCAI-97 workshop on AI in Distributed Information Networks.* IJCAI.

Frei, C., and Faltings, B. 1998. A dynamic hierarchy of intelligent agents for network management. In *Second International Workshop on Intelligent Agents for Telecom Applications (IATA'98).*

Galis, A.; Gantenbein, D.; Covaci, S.; Brianza, C.; Karayannis, F.; and Mykoniatis, G. 1996. Towards multi-domain integrated network management for ATM and SDH networks. In *Proc. Int. Symp. on Advanced Imaging and Network Technologies - Conf. on Broadband Strategies and Technologies for Wide Area and Local Access Networks (BSTW'96).* Berlin, Germany: IEEE.

Karali, D.; Karayannis, F.; Berdekas, K.; Reilly, J.; and Romano-Critchley, D. 1998. QoS-based multi-domain routing in public broadband networks. In *IS&N'98.*

Leeuwen, J. v. 1990. *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity. Elsevier. chapter Graph Algorithms.

Magedanz, T. 1996. Mobile agent-based service provision in intelligent networks. In *Proceedings of the ECAI'96 workshop on Intelligent Agents for Telecoms Applications.* Budapest, Hungary: ECAI.

Winter, C.; Busuioc, M.; and Titmus, R. 1995. Intelligent agents for service management in integrated fixed and mobile networks. In *Proceedings of the IJCAI'95 workshop on AI in Distributed Information Networks.* Montréal, Québec: IJCAI.