ROUTING IN A HIERARCHICAL STRUCTURE

Piet Van Mieghem

AReNA
Alcatel Corporate Research

Francis Wellesplein 1, B-2018 Antwerp (BELGIUM)

mieghemp@rc.bel.alcatel.be

ABSTRACT

Routing in a given hierarchical structure is discussed. The work is inspired by the ATM Forum's PNNI specification. As a general conclusion, we argue that any routing problem that has a solution in a single (flat) network, is also tractable and feasible in a hierarchy.

INTRODUCTION

Let us assume that we are able to construct a hierarchical structure of a network topology (see Van Mieghem, 1998b). A hierarchical structure is just another topological representation of the original network with some additional features.

The most important property of a hierarchy is the level-structure that draws on information condensation (node and link aggregation) to yield scalablility. Each level k in a hierarchical structure represents the complete original network, however, in a different degree of detail. Hence, if the hierarchy contains N levels, then the hierarchical structure exhibits in some sense a N-fold redundancy. The lowest level ($k = 0$) is but the original network, already divided into subnets (or in the language of PNNI (ATMF, 1996) peer groups). The subsequent levels ($k>0$) are condensed forms of the underlying levels. In particular, each subnet on a lower level is represented by a single item, that is called a complex (or logical) node in PNNI.

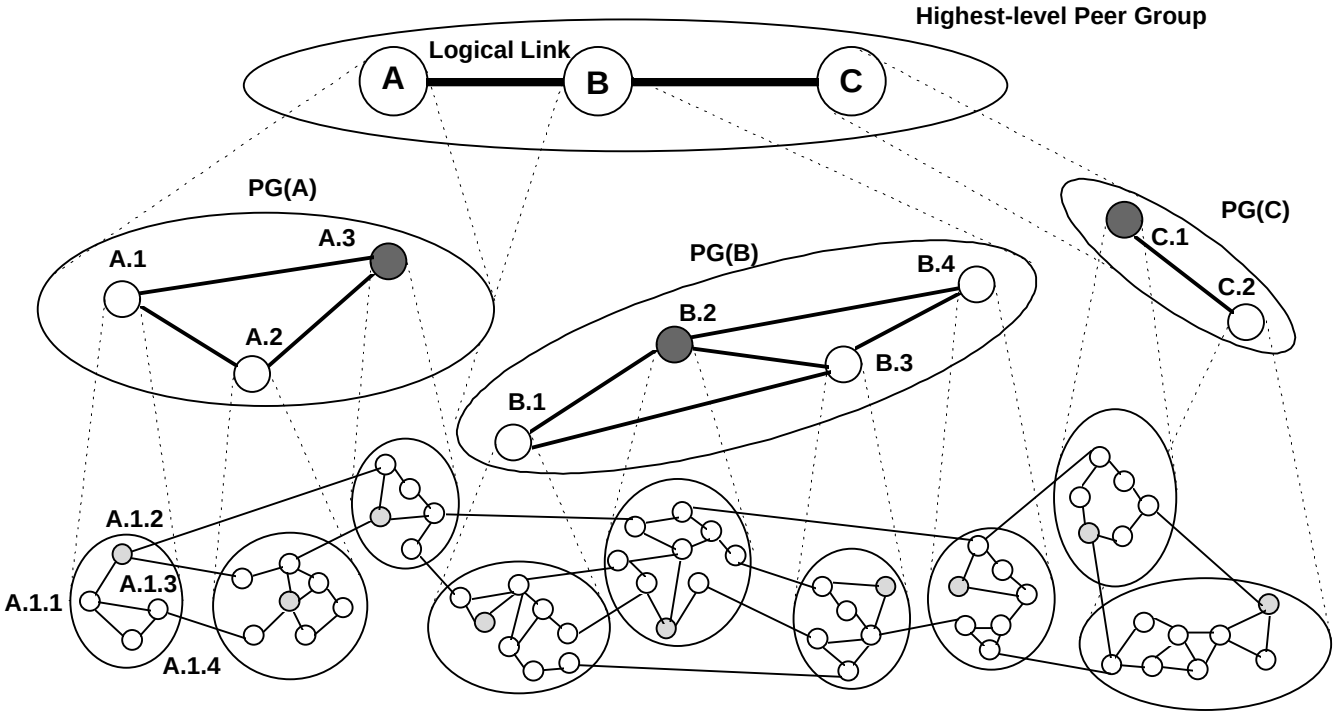

Figure 1. A hierarchical structure based on the specifications of PNNI where a subnet is coined a peer group, denoted as PG(.).

That complex node is the parent of each node in the underlying lower level subnet (or peer group). More general, complex nodes on higher levels are ancestors of the nodes on different levels they represent. Earlier (Van Mieghem, 1997, 1998b), we have given estimates how to divide an original network into peer groups and how many hierarchical levels a network is about to have.

Here, we will concentrate on how to route in a hierarchical structure. We have already shown (Van Mieghem, 1997) that important savings in computational effort can be achieved when we dispose of a hierarchical structure of the original network. These earlier estimates motivate a deeper and more concrete discussion.

THE PRINCIPLE

Suppose a path is to be found between a source *A* and a destination *B*. The *basic routing principle* in a hierarchy consists of *top-down level routing* as follows:

(1) Search for that level k where the ancestor of *A* and the ancestor of *B* belong to one logical subnet.

(2) Construct at that level k the path, P_k, over complex nodes from *A*'s ancestor to *B*'s ancestor.

(3) Specify path P_k in ever greater detail by descending in the hierarchy to lower levels until the physical level (k=0) is reached.

We will demonstrate that this principle holds in general. Whatever routing problem, i.e. QoS (source) routing, multicast routing or hop by hop routing, is considered, the above recipe is applicable. Before concentrating to its details, we present a different approach inspired by the way the PNNI specification determines the designated transfer list (DTL), which is a hierarchical representation of the path from A→B. In particular, the routing is established using the view point of a level *k=0* node in a hierarchy.

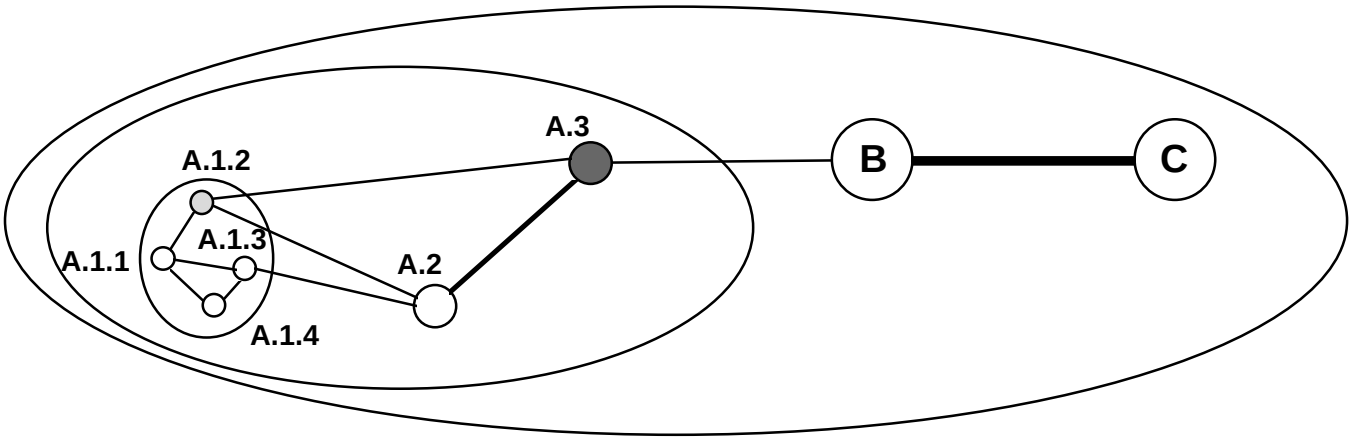

Figure 2. The world view according to physical node A.1.4 is called the 'projected' topology of A.1.4.

The world according to such a physical node consists of all its own ancestors on all levels. For instance, in Figure 1, the view point of the physical node A.1.4 is the union of its own peer group A.1, that of its parent peer group PG(A) and the highest level peer group that overviews - albeit condensed - the complete topology.

The idea then is to route in the 'projected' topology that is obtained by placing all physical and complex ancestor nodes into one topology. The 'projected' topology of the world view of A.1.4 is drawn in Figure 2 which exhibits the projection of the higher hierarchical levels onto the physical level *k=0*. The uplink information is useful for the construction of this 'projected' topology. This method of routing, which I call *'projected' topology routing* focuses on how each node individually perceives the network (in a hierarchically distributed environment). Two disadvantages are immediate.

The routing is computed based on a 'heterogeneous' topology where (complex) nodes of a different level have different meaning and accuracy. The routing does not rely on the quality of the information condensation (Van Mieghem, 1998a) and it ignores the hierarchical structuring of the topology information. Second, routing is again performed in a large topology while it was shown previously (Van Mieghem, 1997) that hierarchy substantially enhances scalability in routing. Hence the routing complexity will be larger than in the top-down level routing outlined above. Finally, since the idea of using the 'projected' topology is found of a less conceptual beauty and since it is far from obvious how to use this method in multicast routing, we will not dwell on it in the sequel.

UNICAST ROUTING

The above principle is invoked to compute the path between source *A* and a destination *B* in a hierarchical structure with a single metric, say delay for example (see Figure 1).

The first action is to determine that level where both ancestors of *A* and *B* are peers. This brings us to the identifier representation in a hierarchical structure. The identifiers of (logical) nodes are not necessarily the same as the addresses of the (logical) nodes. Just as in the original network, there is a one-to-one relation between a (logical) node and its identifier

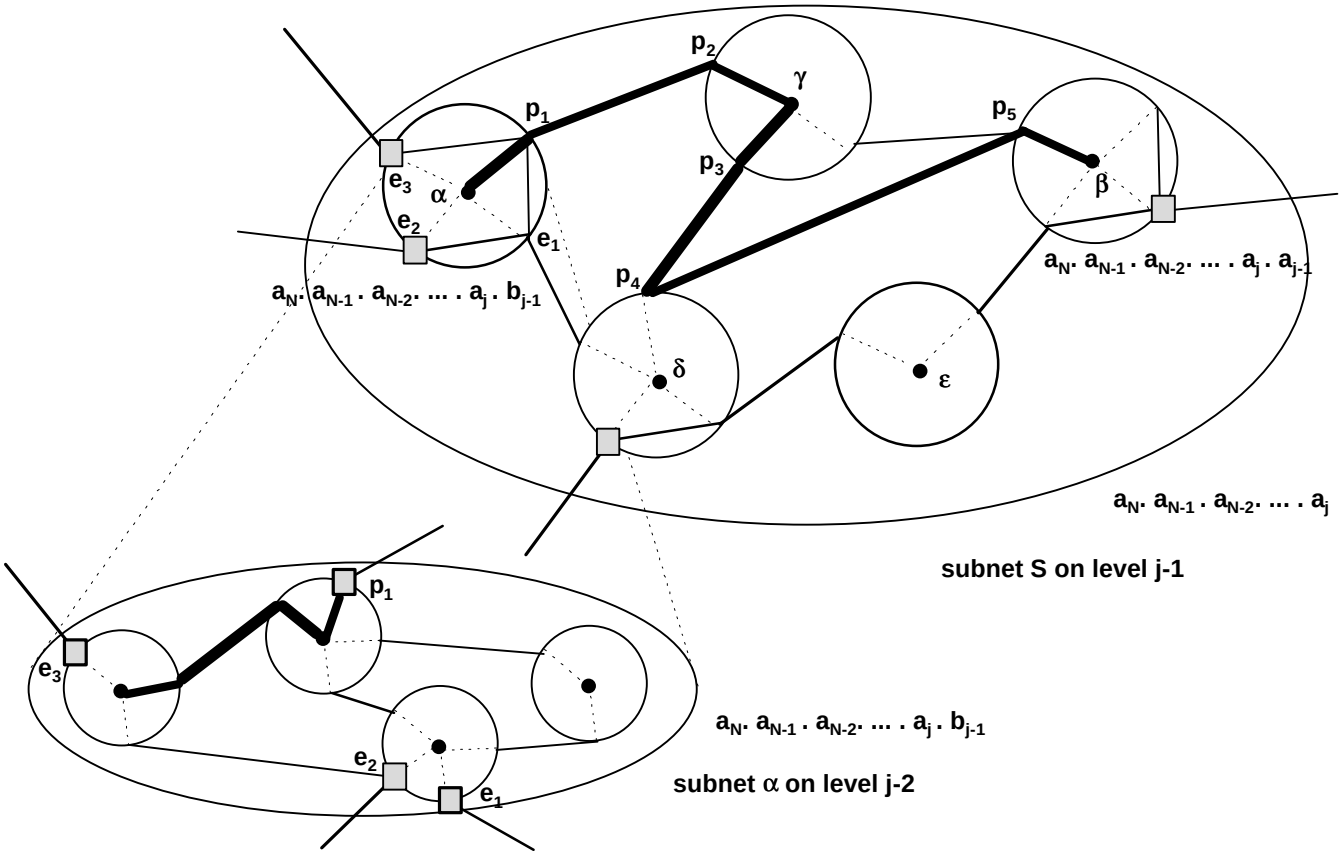

Figure 3. The subnet S on level *j-1* containing the ancestor of *A* and *B* , denoted as *a* and *b*. Further, the other peer nodes are denoted by Greek letters and the port numbers along the path *P* are represented by the set $\{p_j\}$. The squares denote in- or egresses of the subnet under consideration. Not all port numbers are still in- or egresses when viewed on a next higher level. The shortest path *P* between their nuclei is $P = \{(a, p_1), (p_2,g, p_3), (p_4,d, p_{43}), (p_5,b)\}$. These port numbers are physical nodes (already defined on level *k=0* as illustrated in the specification of the complex node α).

.

Identifiers.

In order to obtain an efficient identifier representation, the properties of the hierarchy must be exploited. As illustrated in Figure 1 (and conform to PNNI), a good identifier scheme reflects immediately the place of the logical node in the hierarchy. We refer to this scheme as the PNNI identifier scheme. For a physical node (on level *k=0*), A_0, we have precisely *N+1* identifier coefficients or we can say that A_0 is decomposed (denoted as decomp(A_0)) as

$$\text{decomp}(A_0) = a_N \,.\, a_{N-1} \,.\, a_{N-2} \,.\, ..a_2 \,.\, a_1 \,.\, a_0$$

In general, a (logical) node on level *k*, A_k, is decomposed as

$$\text{decomp}(A_k) = a_N \,.\, a_{N-1} \,.\, a_{N-2} \,.\, ..a_{k+2} \,.\, a_{k+1} \,.\, a_k$$

where all identifier coefficients a_i for $i<k$ are zero (and not shown in the decomposition).

With this identifier scheme, we find the level where both ancestors of *A* and *B* are peers as follows. Since *A* and *B* are physical nodes, their address decomposition is decomp(A) = $a_N \,.\, a_{N-1} \,.\, a_{N-2} \,.\, ..a_2 \,.\, a_1 \,.\, a_0$ and decomp(B) = $b_N \,.\, b_{N-1} \,.\, b_{N-2} \,.\, ..b_2 \,.\, b_1 \,.\, b_0$ respectively, with of course, $a_N = b_N$ because at the highest level there is but one complex node.

The level of interest appears as the lowest index j of the identifier coefficients where holds that $a_j = b_j$ and $a_{j-1} \neq b_{j-1}$. Although simple, this identifier scheme is clearly not that efficient because for each node in the hierarchy, we need to store $N+1$ integers. An alternative scheme based on a polynomial transform of the identifier coefficients into one identifier is proposed in the appendix.

In summary, the first step in the routing principle has an efficient and relatively simple solution.

The path on level j-1.

From the knowledge of j determined above or via (1) in the appendix, the particular subnet, say S, in which we have to consider the routing between the ancestor of A and B is found via the identifier coefficient $a_j = b_j = [V^*/ x^j]$. Recall that the identifier of a subnet equals the identifier of its complex node on the next higher level. Thus, the polynomial transform of the identifier (see appendix) of that subnet equals $S^* = a_N x^N + a_{N-1} x^{N-1} + a_{N-2} x^{N-2} + ... + a_{j+1} x^{j+1} + a_j x^j$. This means that the routing must be performed on level $j-1$ in subnet S between the ancestor nodes of A and B distinguished by their identifier coefficient $a_{j-1} = [A^*/ x^{j-1}]$ and $b_{j-1} = [B^*/ x^{j-1}]$ with the respective identifiers $S^* + a_{j-1} x^{j-1}$ and $S^* + b_{j-1} x^{j-1}$.

In the simple case with a single metric as considered here, the path is readily found invoking a shortest path algorithm (e.g. Dijkstra, see Cormen et al., 1995). The presentation of that path follows from Figure 3. The path starts in the nucleus of the ancestor node of A and travels to the egress port of that complex node. From thereon, the path moves over the non-ancestor nodes and specifies both the complex node ID (or address) and the corresponding ingress and egress port ID. Finally, the path ends in the nucleus of the ancestor node of B which it has penetrated via the appropriate ingress port.

Further Specifying the path.

In order to further specifying the path found on level $j-1$, we descend one level. Clearly, the ancestor subnets with the identifier given above need special treatment. Let us focus on the ancestor subnet of A with identifier $S^* + a_{j-1} x^{j-1}$. In that subnet we search for the path from the node with identifier $S^* + a_{j-1} x^{j-1} + a_{j-2} x^{j-2}$ to the egress port node determined at the previous higher level (e.g. port 1 in Figure 3). The situation for the ancestor subnet is analogous. For the intermediate subnets on level $j-2$, we specify the path between ingress port and egress port, also determined above (e.g. (p_2,g, p_3), (p_4,d, p_{43}) in Figure 3).

Notice that the port ID's are always physical node numbers as follows from the construction of a hierarchy (Van Mieghem, 1998a). Again, the result of these path searches are of the same form as explained above. Hence, the cycle is completely described and the process stops on level $k=0$.

MULTICAST ROUTING

Although routing in a hierarchy has been explained only for unicast routing, other types of routing are fairly analogous. For example, let us briefly concentrate on an instance of static multicast routing (see Figure 4).

We are to find a suitable spanning tree for the multicast session between t (known) terminals T_i on level $k=0$. The common ancestor subnet on level j that contains each ancestor of terminal T_i is readily found by pairwise comparison of the result of (1) and by maintaining the maximum. This demands $t(t-1)/2$ times an execution of (1) leading to the desired level j .

In the common ancestor subnet, say S, a spanning tree algorithm provides us with a tree over complex nodes and their corresponding ingress and egress ports. For the detailing to lower levels, two cases must be distinguished. First, if all terminals have distinct ancestor complex nodes in the common ancestor subnet S on level $j-1$, then the further specifications of the branches of this tree proceeds precisely as in the unicast case because each tree can be regarded as a path from terminal to branch point (e.g. subnet ε in Figure 4). Second, if two or more terminals (e.g. $\alpha.1$ and $\alpha.2$ in Figure 4) have a same ancestor complex node (e.g. α in Figure 4) in the common ancestor subnet S at level $j-1$, the spanning tree at that level $j-1$ cannot distinguish between these terminals. Therefore, (and further referring to Figure 4), the subnet with identifier corresponding to the ancestor complex node α must be consulted at level $j-2$ to clarify the relations between these terminals $\alpha.1$ and $\alpha.2$ and how they are linked to the rest of the multicast group (typically via ports such as p_1 and e_1, that are determined by the spanning tree computed at level $j-1$). This implies that in subnet α at level $j-2$, again a spanning tree must be computed. From then on, the process can be iterated until the level $k=0$ is reached. In summary, the route computation of a multicast tree in a hierarchy, needs both a spanning tree and a shortest path algorithm. Both of them may be alternatively used depending on the topology and set of terminals, as explained above.

In actual (non-static) multicast sessions, the number of terminals may vary over time and in most of the current multicast protocols, the terminals (sources or receivers) have no knowlegde about all members in a multicast group. Terminals are added to the tree usually not in an optimal fashion (because (1) the minimal Steiner tree (mst) problem is NP-complete and (2) the mst may fundamentally change by adding of removing one terminal leading to instability on short time scales).

In most multicast protocols, the problem of multicast routing even reduces to a simpler problem of finding the shortest path from the new terminal to some node (determined via the specifics of the protocol) that already belongs to the multicast tree.

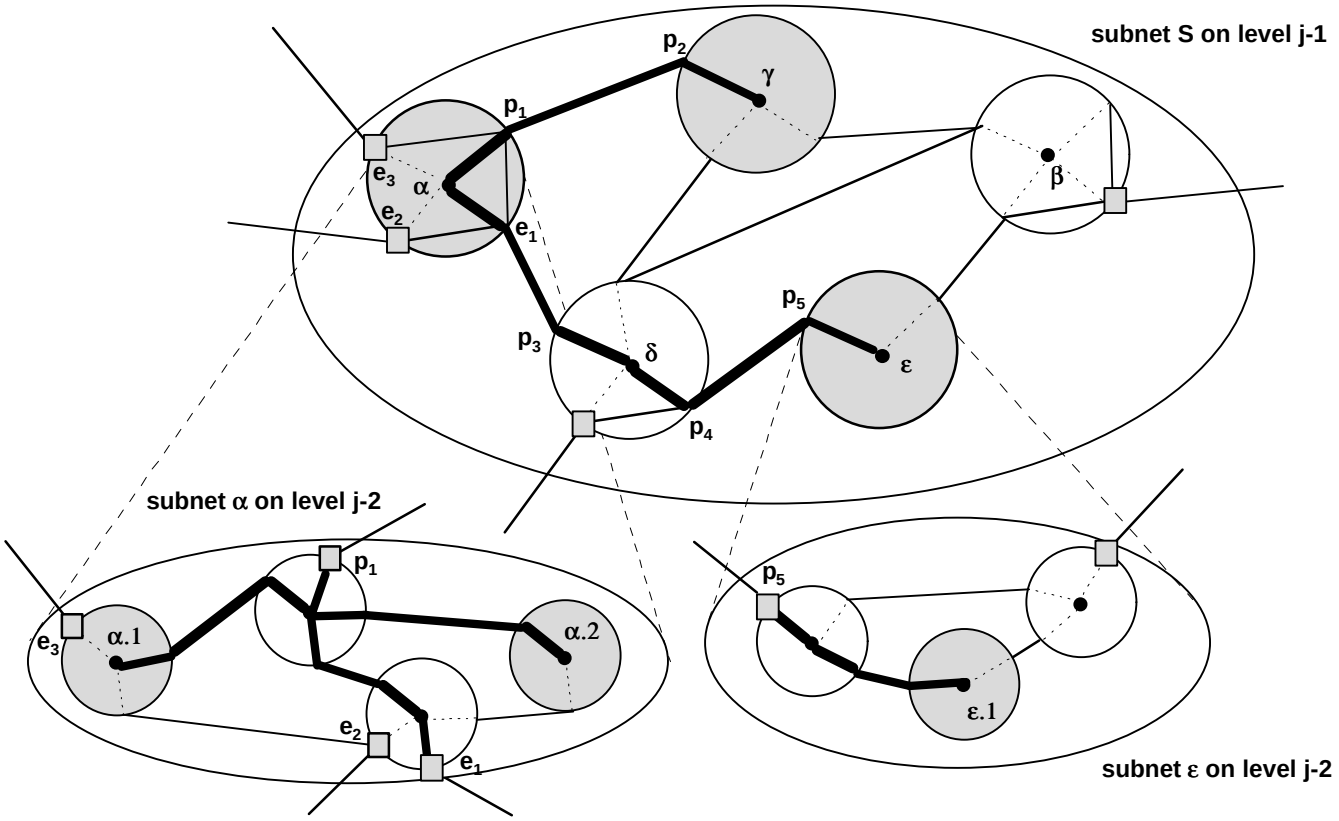

Figure 4. In the same hierarchy as in Figure 3 a static multicast tree is constructed. At level *j-1* three complex nodes *a*, *y* and *e* are member of the multicast group while at level *j-2*, subnet *a* contains two members *a.1* and *a.2* whereas subnet *e* only has one member *e.1*. At level *j-1*, the multicast spanning tree comprises the branches $a \rightarrow \gamma$ and $a \rightarrow \varepsilon$. At a lower level, the situation for subnet α is clearly different than for subnet ε. In subnet ε, it is sufficient to compute a shortest path between port p_5 and the nucleus of *e.1*. In subnet α, on the other hand, a spanning tree connecting the ports p_1 and e_1 and the nuclei of *a.1* and *a.2* needs to be calculated.

CONCLUSIONS

We have demonstrated that the hierarchy poses no additional problems for unicast and multicast routing. Moreover, we argue that every routing problem, solvable in single topology (one subnet), also has a solution in a hierarchy, that is feasible and tractable. Intuitively, routing in a hierarchy seems complicated, but, if one bears in mind that the hierarchical structure is nothing else than a pre-processing of the original network (to exploit scaling advantages), it is easier to accept that all routing problems have a solution in that hierarchy.

Due to the information condensation, the path computed in the hierarchy is likely to be different than the exact path computed in the original network. The accuracy of the 'hierarchical path' strongly depends on the methodology used to calculate the hierarchical structure. Therefore, the largest difficulty with the hierarchy lies in the problem of how to construct the hierarchy rather than in routing.

At last, the principle described here is closely related to the DTL as defined in PNNI. Contrary to the 'projection' view (where a bottom-up routing is used), the hierarchical principle computes paths via a top-down approach.

ACKNOWLEGDEMENTS

We thank Y. T'joens for his valuable discussion where he mentioned the alternative picture for routing which I have called 'projected' topology routing and H. De Neve for his instructive inputs.

APPENDIX: ALTERNATIVE IDENTIFIER SCHEME

Fortunately, from the PNNI identifier scheme an alternative with optimal efficiency, in the sense that only one integer is needed per node, can be deduced. Let x be an integer such that $x > \max(a_m)$ and consider the polynomial[1],

$$A^* = a_N x^N + a_{N-1} x^{N-1} + ... + a_2 x^2 + a_1 x + a_0$$

With this definition, the identifier of node A equals $A^* > 0$. Moreover, the identifier decreases with increasing level and the possible range varies from x^N (the identifier of the highest level subnet that comprises the whole original network) to the integer smaller than x^{N+1}. Thus, the PNNI scheme requires to store (N+1) identifier coefficients of the order of x while the polynomial transform scheme just needs one integer of the order of x^{N+1}.

Notice, however, that the binary representation of numbers, needs $(N+1)\ log_2 x$ bits for both schemes.

The identifier coefficients immediately follow by decomposition of A^* in base x. Explicitly, the identifier coefficients are found from the integer A^* via the short algorithm

1. initialise: $k = N$ and $Z^* = A^*$
2. **do until** k = 0:
3. $a_k = [Z^*/ x^k]$
4. $Z^* \leftarrow Z^* - a_k x^k$ (the right hand side is equivalent to $Z^* \bmod x^k$)
5. $k \leftarrow k-1$

The verification of the algorithm is instructive because it will emphasise the role of the integer x. Since $a \bmod b = a - [a/b]\ b$, where the integral part operator $[y]$ denotes the largest integer smaller (or equal to) y, step 3 means that

$$Z^*/ x^N = a_N + a_{N-1} x^{-1} + ... + a_2 x^{2-N} + a_1 x^{1-N} + a_0 x^{-N}$$

Since we have required that $x > \max(a_m)$ for all m or equivalent $a_m \leq x-1$, it holds that

$$a_{N-1} x^{-1} + a_{N-2} x^{-2} + ... + a_2 x^{2-N} + a_1 x^{1-N} + a_0 x^{-N}$$

$$\leq (x-1)(x^{-1} + x^{-2} + ... + x^{-N})$$

$$< (x-1)(x^{-1} + x^{-2} + ...) = (x-1)(1/(1-1/x)-1) = 1$$

and taking the integral part of the left hand side leads to a_N. The remaining steps are obvious. Now, in order to reduce the size of the address A^*, it is important to find the smallest possible value of x .

Suppose the original network contains M physical nodes. Then, the maximal number of nodes in a hierarchy is $2M$. Indeed, the minimal condensation equivalent to a maximal number of logical nodes, occurs in case every complex node combines precisely 2 lower level nodes.

Then, on level $k = 0$, there are M nodes, on level $k = 1$, there are about $M/2$ nodes, on level $k = 2$, there are about $M/4$ nodes, and so on. In total over the N levels, we have $M(1+1/2+1/4+1/8+...+2^{-N}) < 2M$. Thus, we have at most M logical nodes and, in addition, the maximum number of levels is not higher than $N < log_2 M$. On the other hand, we clearly have that M = (number of nodes in subnet 1 on level 0) + (number of nodes in subnet 2 on level 0) + ... + (number of nodes in subnet p on level 0). The requirement on x implies $x > \max_i$(number of nodes in subnet i on level 0) with $1 \leq i \leq p$. In addition, the number of nodes on level 1 precisely equals p (as follows from the construction of a hierarchical structure). Again, $x > p$. But, the number of nodes on level 2 can never exceeds p implying that higher level considerations are irrelevant for the determination of x. Hence, we conclude,

$x = \max[p, \max_{1 \leq i \leq p}$(number of nodes in subnet i on level 0)]

The presented analysis also demonstrates that the desired level j is related to the difference

[1] As an intermezzo at this stage, we would like to point out that the same principle - using an integer polynomial transformation - can be applied to represent paths in a single network. Indeed, each path between two nodes, say A and B, defined by a node list $P = \{n_1, n_2, n_3,.., n_p\}$ where $A = n_1$ and $B = n_p$, maximally consists of $p \leq M-1$ integers n_i lying between $1 \leq n_i \leq M$, where M is the number of nodes in that network. Hence, we choose $x = M+1$. Further, denoting the path-transform analogously as above by a star P^*, we observe that $(M+1)^{M-1} \leq P^* \leq (M+1)^M$. Although this range of P^* rapidly increases with the number of nodes, this transform may be useful to quickly compare two different paths. Performing operations (e.g. subtractions) between two path-transforms (similar to (1), but not entirely the same) will give at once the number of common (and vice-versa different) nodes visited by the paths. This knowledge can be of interest for re-routing problems.

$$V^* = A^* - B^* = (a_{j-1} - b_{j-1})x^{j-1} + (a_{j-2} - b_{j-2})x^{j-2} + \ldots + (a_2 - b_2)x^2 + (a_1 - b_1)x + a_0 - b_0$$

Moreover, since $V^*/x^m = x^{j-1-m}(a_{j-1} - b_{j-1} + e)$ with $|e| < 1$ implying that $(a_{j-1} - b_{j-1} + e) < x$, we have that $\log_x(V^*/x^m) = j-1-m + \log_x(a_{j-1} - b_{j-1} + e) = j-m+e^*$ with $|e^*| < 1$ and that the desired level j is the solution $m=j-1$ of the equation $[\log_x(V^*/x^m)] = 0$. But, this equation is readily rewritten as $[\log_x(V^*) - m] = 0$ which brings us to our final result that

$$j = 1+[\log_x(V^*)] = 1+[\ln(V^*)/\ln(x)] \qquad (1)$$

REFERENCES

ATMF, 1996, *Private Network Network Interface*, specification version 1.

Cormen, T. H., C. E. Leiserson and R. L. Rivest, 1995, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts.

Van Mieghem, P., 1997, "Estimation of an Optimal PNNI Topology", Proceedings of the IEEE ATM'97 Workshop, May 26-28, Lisboa, Portugal, pp. 570-577.

Van Mieghem, P., 1998a, "Node and Link Aggregation in a Hierarchy", submitted to IEEE/ACM Transactions on Networking.

Van Mieghem, P., 1998b, "Dividing a Network into Peer Groups to Build a Hierarchical Structure", submitted to the First International Workshop on the Design of Reliable Communication Networks, DRCN 98, Brugge, May 18-20.