On the Distribution of Routing Computation in Hierarchical ATM Networks

Eyal Felstaine and Reuven Cohen
Dept. of Computer Science
Technion, Haifa 32000
Israel

Abstract

ATM PNNI (Private Network-to-Network Interface) is a hierarchical and dynamic link-state routing protocol, designed to scale to the largest possible ATM networks, encompassing thousands of nodes. The paper investigates the route computation load imposed by the PNNI routing scheme, and shows that this load is unevenly distributed among the network nodes. More specifically, the routing computation load associated with the set up of a single VC grows exponentially with the hierarchy level. As a result, some of the network nodes – mainly those that function as border nodes of high levels – may be overloaded with route computation, while other nodes are rarely involved in this process. The paper also proposes a possible scheme for spreading the route computation burden more evenly. According to this scheme, heavily loaded nodes transfer route computation tasks to lightly loaded nodes.

1 Introduction

In computer networks, routing is a collection of algorithms that determine the routes that data packets will traverse until reaching their destination nodes. In order to make routing decisions, the network nodes should obtain topology information and maintain routing tables. There are two well-known approaches to perform this task distributedly. In the first approach, called *distance-vector routing* [6], each node sends to its *neighboring nodes* its *entire* routing table. The receiving nodes use the received information in order to update their own routing tables, which they then send to their own neighbors. In the second approach, called *link-state routing* [6], each node broadcasts to *all network nodes* information regarding the status of its *local links only.* The network nodes use the received information in order to create and maintain an up-to-date network map, from which they deduce their routing tables.

The main drawback of the distance-vector algorithm is that it takes a long time to reconverge to alternate paths when a failure occurs in the network [6]. During that time, the routing tables may define loops that may cause congestion in the network. The link-state protocol responds much faster to topology changes. However, in large networks it lays excessive communication, storage and processing burden on the nodes. This scalability problem can be addressed using the concept of hierarchical routing [7] which helps to avoid the excessive complexity in topology advertisements. According to this concept, the network nodes and links are organized hierarchically. At the lowest level of the hierarchy, each node represents an ATM switch and each link represents a physical link or an ATM virtual path (VP). The nodes and links of each level can be recursively aggregated into higher levels, such that a high-level node represents a collection of one or more lower level nodes, and a high-level link represents a collection of one or more lower level links. The OSPF protocol [6], used for autonomous system routing in the Internet [6], has two levels of hierarchy. The ATM PNNI (Private Network-to-Network Interface) [1] is a hierarchical, dynamic link-state routing protocol, designed to scale to the largest possible ATM networks, encompassing thousands of switches. It therefore may support a maximum of 105 hierarchy levels.

In PNNI, every lower level node (i.e. ATM switch) maintains detailed topology information about the lowest cluster to which it belongs. Such a cluster usually contains a small number of switches and links, and maintaining the detailed information does not lay excessive communication, storage and processing burden on the switch. In contrast, the switch maintains only compressed information about his parent cluster, which contains a larger number of switches and links, more

compressed information about its grandparent cluster, and so fourth.

When a virtual channel needs to be set up, the source node creates a hierarchical route consisting of a detailed path within the source node cluster, a less detailed path within the source node's parent cluster and so on until reaching the lowest level cluster which is an ancestor of both the source and destination nodes. When this cluster is reached, a new "source" route is computed (not by the source node, but by some intermediate node) to descend to the final destination. A "source" route is also computed by intermediate border nodes in each hierarchy level, that need to determine the best way for crossing their clusters.

In general, many nodes can be involved in the computation of a detailed route from the source to the destination. Some of these nodes need to choose only one route, while others need to choose two or even more routes in different clusters residing in different hierarchy levels. The exact number of such nodes and the exact load laid upon each of them depend on the hierarchical structure of the network and on the relative location of the source and destination nodes.

The present paper investigates the route computation load imposed by PNNI routing and shows that this load is unevenly distributed among the network nodes. More specifically, the routing computation load associated with a single message (i.e. a single VC) grows exponentially with the hierarchy level. As a result, some of the network nodes – mainly those that function as border nodes of high levels – may be overloaded with route computation, while other nodes are rarely involved in this process. The paper proposes a possible scheme for solving this problem. According to this scheme, heavily loaded nodes "transfer" route computation tasks to lightly loaded nodes, such that the burden of route computation is more evenly distributed among all network nodes.

The rest of the paper is organized as follows. Section 2 describes the hierarchical routing schemes of ATM PNNI in more details. Section 3 analytically analyses the number of route computations required for routing a single message, and the distribution of this load among the routing levels. The results of this section are used in Section 4, in order to analyze the distribution of the route computation load among individual nodes. A possible solution to the uneven distribution of route computation load is presented in Section 5. Section 6 concludes the paper.

2 An Overview of ATM PNNI Routing

This section presents a description of the ATM PNNI routing as defined by the ATM Forum. ATM PNNI uses hierarchical link-state routing, with maximum of 105 levels. To support this hierarchy, PNNI defines a uniform network model at each level, with a set of logical nodes connected by

logical links. Each lowest level node represents a physical ATM switch with a unique ATM address. Nodes within a given level are grouped into sets of peer groups. A peer group (PG) is a collection of logical nodes that exchange link-state messages, called PTSPs (PNNI Topology State Packets), with other members of the group, such that all members maintain an identical view of the group. Each PG is assigned a unique *peer group identifier* (PGID) and is represented in its parent PG as a single node, called *logical group node (LGN).*

Figure 1 shows an example of ATM PNNI hierarchy. In this figure, a logical node whose identity is of the form $x.y.z$ represents a physical node (ATM switch). Nodes $1.1.1 - 1.1.4$ are grouped into a PG called PG(1.1), and in the same way additional 5 PGs are created. Hence the network has 22 logical nodes in level 0 and 6 logical nodes in level 1. In level 2 there exist only two logical

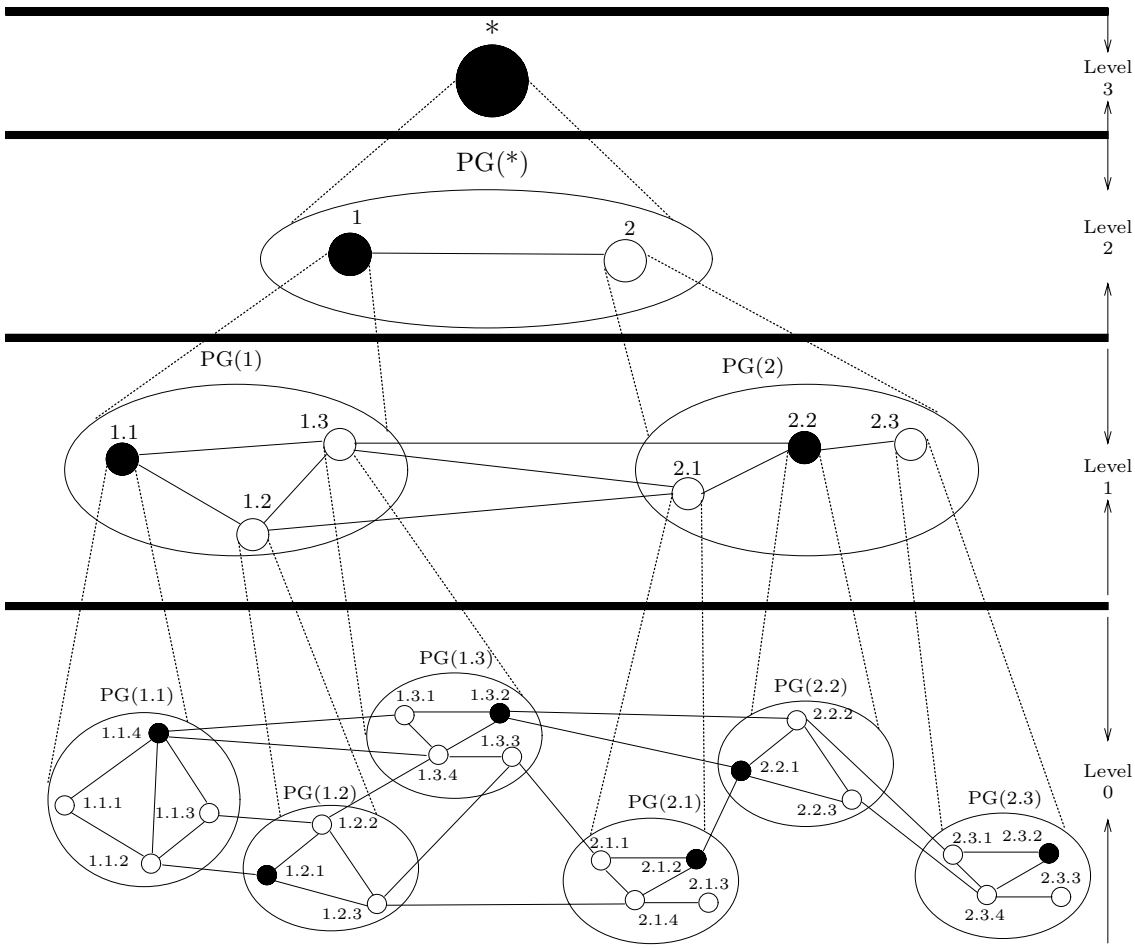

Figure 1: An example for ATM PNNI hierarchy.

nodes: PG(1), which consists of LGNs (1.1)-(1.3), and PG(2), which consists of LGNs (2.1)-(2.2).

The upper level always has only one logical node (LGN $*$ in Figure 1). The links between the physical (level 1) nodes, like (1.1.1,1.1.4) or (1.3.2,2.2.1), represent physical links or virtual path (VP) connections. By contrast, the links between logical nodes in level 2 or above, like (1.1,1.3) or (1,2), represent a *set* of physical links and VP connections.

As already indicated, each PG is represented in the next hierarchical level by a single LGN. The functions associated with an LGN are actually performed by the *peer group leader (PGL).* The PGL is determined by means of a distributed election process, executed by all the LGNs in the PG (See [1] for more details). In Figure 1, the PGLs are those LGNs represented by black circles. Note that physical node 1.1.4 functions as a PGL of PG(1.1). However, since LGN 1.1 is the PGL of PG(1) and LGN 1 is the PGL of PG($*$), the functions that LGNs 1.1 and 1 are supposed to perform as PGLs are actually implemented by physical node 1.1.4.

The physical and logical links are classified into two categories: inside links, that connect two nodes within the same peer group, and outside links, that connect two nodes within two different peer groups. Nodes connected by outside links are referred to as border nodes. For instance, in level 1 of Figure 1, physical nodes 1.1.4 and 1.3.1 are border nodes and the physical link connecting them is an outside link. In level 2, nodes 1.3 and 2.2 are border nodes and the logical link (1.3,2.2) connecting them, which represent the two physical links (1.3.2,2.2.2) and (1.3.2,2.2.1), is an outside link.

By exchanging topology information among nodes, each node obtains the information needed to create its "view of the world". Figure 2 depicts the local view obtained by the nodes in PG(2.2) of Figure 1. The ATM PNNI is supposed to provide sophisticated QoS routing while still allowing flexibility in the choice of route computation. Hence, each implementation is free to use its own algorithm. This gives rise to source routing, which does not require different switches to agree on the same computation. The source node (actually, the switch of the source host) creates a hierarchical route consisting of a detailed path within the source node PG, a less detailed path within the source node's parent PG and so on until reaching the lowest level PG which is an ancestor PG to both the source and destination nodes[1]. When the LGN that contains the destination node in the lowest level common ancestor is reached, a new "source" route is computed to descend to the final destination. A "source" route is also computed when necessary by border nodes that determine the best way to cross their PG.

The path is encoded as a set of Destination Transit Lists (DTLs), which is explicitly included

[1]An ancestor PG to a node (or to a logical node) is a PG in which that node resides

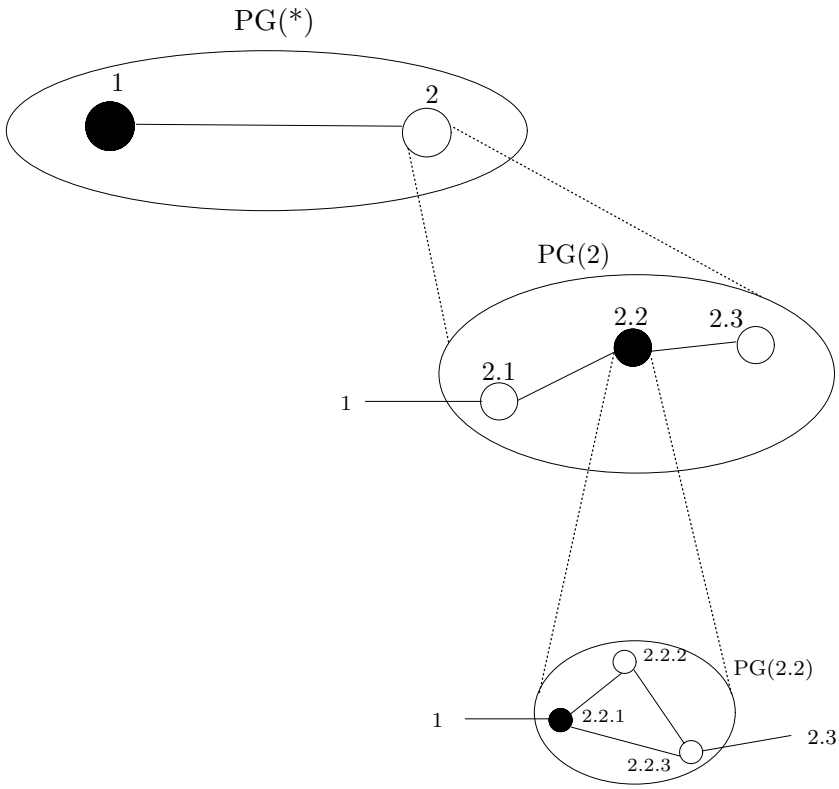

Figure 2: Local view of the network from nodes in PG(2.2).

in a stack within the PNNI signaling call setup request. Each DTL contains the description of a path for one level in the hierarchy. It explicitly specifies every LGN, and optionally every link, used to cross the PG. Each DTL is associated with a pointer that indicates the next element in the list to be processed.

As an example, assume a VC connection is to be set up between a host of 2.2.3 and a host of 1.1.2 in Figure 1. Node 2.2.3 examines its local view of the topology (Figure 2), and finds three possible paths to reach node 1, which is a LGN in the lowest level ancestor PG it shares with 1.1.2: (2.2.3,2.2.2,1), (2.2.3,2.2.1,2.1,1) and (2.2.3, 2.2.1,1). Suppose that on the basis of metrics and policy, node 2.2.3 chooses the first option: (2.2.3, 2.2.2,1). The DTL stack representing this path is:

[2.2.3, 2.2.1]

[2.2]

[2, 1]

where the underline indicates the location of the pointer in each DTL. Before forwarding the

call setup message to 2.2.1, node 2.2.3 advances the pointer of the first DTL which becomes: [2.2.3, 2.2.1]. When node 2.2.1 receives the message, it notices that the top DTL points to its own ID. Since the first and the second DTLs are exhausted, node 2.2.1 looks at the third DTL and finds that the message should be routed to node 1. Hence, it advances the pointer in the third DTL and forwards the call setup message to the border node 1.3.2 over link (2.2.1,1.3.2) with the following DTL stack:

[2, 1].

Node 1.3.2 receives the message and realizes that the current DTL destination has been reached. It therefore needs to build a new "source" route to descend to the final destination based on its local view of the network. It can select a path that goes through 1.2 to 1.1, or a path that goes directly from 1.3 to 1.1. Suppose it chooses the second option. It also needs to determine the exact routing inside 1.3, e.g. (1.3.2,1.3.4). It then sends the call setup message with a new DTL stack which looks as follows:

[1.3.2, 1.3.4]
[1.3, 1.1]
[2, 1].

Node 1.3.4 removes the top DTL, advances the pointer of the new top DTL, and hands the message to node 1.1.4, which is its neighbor in 1.1, with the following DTL stack:

[1.3, 1.1]
[2, 1].

Node 1.1.4 notices that it is the current target, and therefore need to find a route to 1.1.2 through PG(1.1). It chooses the route (1.1.4,1.1.1,1.1.2) and sends the call setup message to 1.1.1 with the following DTL stack:

[1.1.4, 1.1.1, 1.1.2].

Node 1.1.1 advances the pointer of the top DTL, and forwards the message to the final destination — node 1.1.2.

The process described so far is needed in order to establish a virtual channel (VC) between the source and destination nodes. After the VC is established, routing is performed in the ATM layer by means of table look-ups. In this paper we analyze the routing burden laid on the network nodes due to the VC set-ups. This burden is encountered whenever a VC setup message is received.

3 The Routing Computation Load on the Entire Network

In this section we present notations and definitions for PGs and LGNs at different levels of the network hierarchy. We then calculate the number of physical nodes that participate in the routing of a message and push DTLs to the DTL stack.

Definition 1 *An m-level PG is a PG in level m of the hierarchy, and it can therefore be recursively subdivided into PGs m times. Similarly an m-level LGN is an LGN in level m of the hierarchy.*

In Figure 3, a 3-level network (N=3) is depicted. In this figure, A.1.1 is a physical node in level 0. PG(A.1) is a 1-level PG and is also a 1-level LGN (Logical Group Node) in PG(A), where the latter is a 2-level PG. The whole network is aggregated into a single 3-level PG, referred in the figure to as PG(*), that contains 5 LGNs: A,B,C,D and E. Note that in order to simplify the figure, the internal structure of LGNs B and E is not depicted.

We continue by defining three parameters x, y and z, related to the hierarchical network design. To demonstrate these parameters, we shall consider a VC-setup message[2] routed in the network of Figure 3 from A.1.2 to D.3.3 through the following physical nodes (represented by black circles in Figure 3): A.1.2, A.1.1, A.1.3, A.3.1, A.3.3, A.3.7, A.4.1, A.4.4, A.4.3, C.1.1, C.1.2, C.1.3, C.2.1, C.2.2, C.2.3, C.3.1, C.3.2, C.3.3, D.1.1, D.1.2, D.1.3, D.2.1, D.2.2, D.2.3, D.3.1, D.3.2 and finally D.3.3.

$x =$ The mean number of LGNs a message should pass in order to cross a PG, including the entry and exit border LGNs (Figure 4(a)). For example, in order to cross PG(A.3) the considered message from A.1.2 to D.3.3 passes through 3 physical nodes: A.3.1, A.3.3 and A.3.7.

$y =$ The mean number of LGNs a message should pass in the source PG, excluding the source LGN (see Figure 4(b), assuming A is the source LGN); or equivalently, the mean number of LGNs a message should pass in the destination PG, excluding the destination LGN. For example, in order to get from the border LGN D.3.1 of PG(D.3) to the destination LGN D.3.3, the considered message has to pass two LGNs: D.3.1 and D.3.2.

$z =$ The mean number of LGNs a message should pass inside the lowest level ancestor PG the source and destination nodes have in common, excluding the source LGN and the destination LGN. In our example, the source node A.1.2 and destination node D.3.3 belong to LGN(A) and

[2] As already noted, in ATM routing decisions are made only during the setup of a VC. Therefore, in the following the term "message" will be used as an abbreviation for a "VC-Set up message".

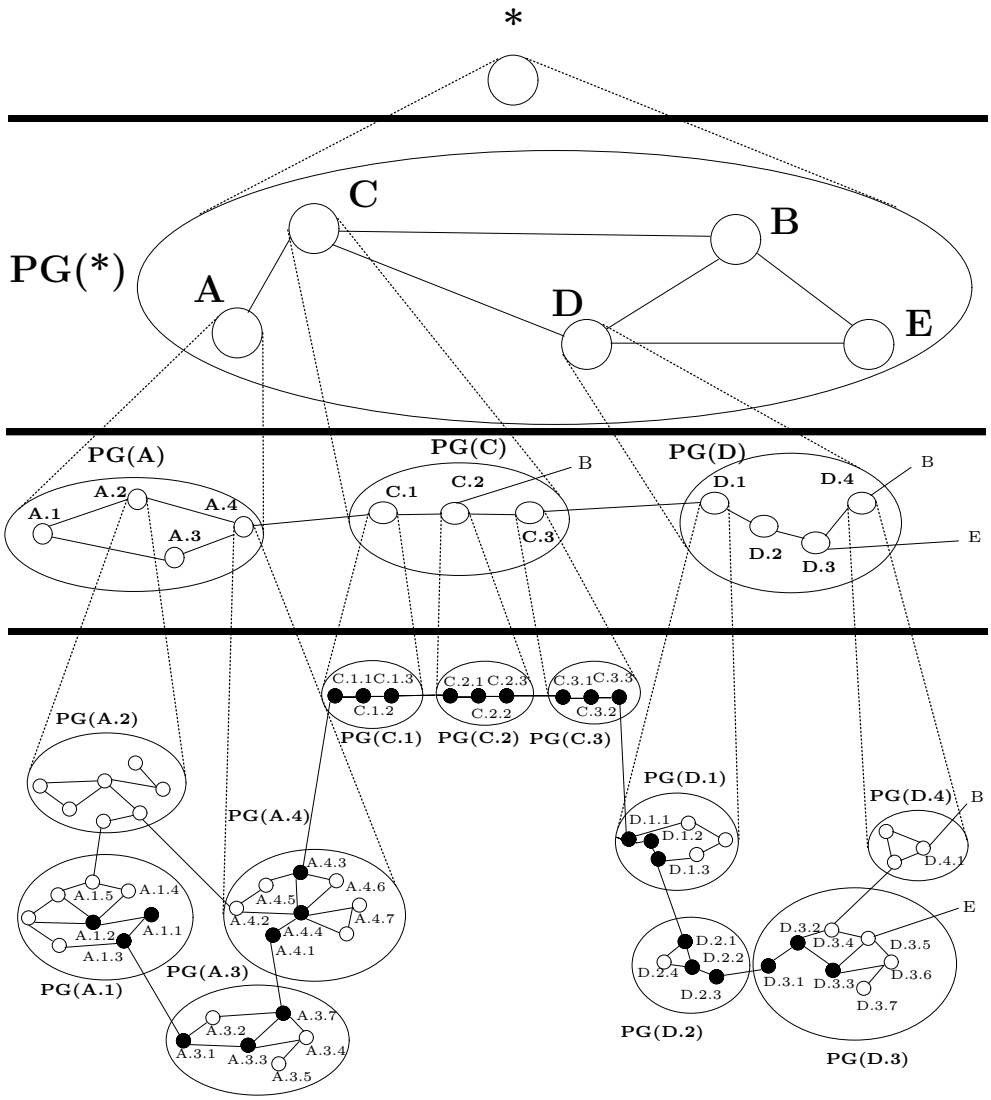

Figure 3: A hierarchical network with 3 hierarchical levels.

LGN(D) respectively, which are contained in PG(*). In order to cross PG(*) from LGN(A) to LGN(D), the message needs to pass through one LGN - LGN(C)

Note that another way to describe x, y and z is as the average distance between two border nodes (x), the average distance between two internal nodes (z), and the average distance between a border node to an internal node (y). Note also that PG(*) has no border nodes since it does not have a LGN representation inside a higher PG. This implies that the structure of this PG may affect the value of z, but not the value of x and y.

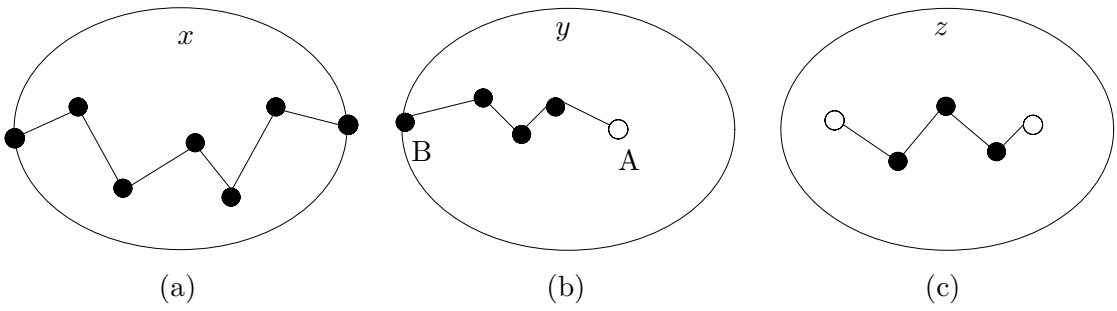

Figure 4: The definitions of x, y and z (only the grey nodes are counted).

In the definitions above, x, y and z are the average achieved for all PGs in all levels of the network. However in a uniform hierarchical network, where different PGs have identical internal topology regardless of their hierarchy level, the values of x, y and z can be deduced from the topology of a single PG.

In order to calculate the number of physical nodes that participate in the routing of a message, and therefore perform pushes to the DTL stack, we denote by Γ_L^m the average number of m-level PGs traversed by a message whose topmost routing level is L. As noted earlier, L is actually the level of the lowest common ancestor PG of the source and the destination. For example the topmost routing level for a message from A.1.2 to D.1.3 in Figure 3 is the level that contains both A and D. Hence, for such a message $L = 3$ holds. In the following we show how can Γ_L^m be expressed as a function of x, y and z.

First note that $\Gamma_L^L = 1$ holds for every message, because the number of LGNs at the highest level a message passes is by Definition 1. For example, in Figure 3 the only LGN in level 3 that a message from A.1.2 to D.3.3 passes is the topmost LGN PG(*). Note also that the number of $(L-1)$-level LGNs a message passes is $\Gamma_L^{L-1} = z + 2$. In our example these LGNs are A,C and D in level $m = 2$.

In order to express Γ_L^{L-2}, note that each of the Γ_L^{L-1} LGNs in level $L-1$ is also a PG, that consists of (L–2)-level LGNs. In the first and the last PGs only $y+1$ (L–2)-level LGNs have to be traversed by the message, whereas in each of the remaining $(\Gamma_L^{L-1} - 2)$ PGs, exactly x (L–2)-level LGNs have to be traversed. Hence $\Gamma_L^{L-2} = 2(y+1) + (\Gamma_L^{L-1} - 2)x$. Using the same rationale, Γ_L^{L-3} can be represented as a function of Γ_L^{L-2}, y and x, and in general:

$$\forall\, 0 \le m \le L-2, \quad \Gamma_L^m = \left(\Gamma_L^{m+1} - 2\right) x + 2\,(y+1)\,. \tag{1}$$

Starting with $\Gamma_L^{L-1} = z + 1$ and using Eq.1 repeatedly, we get:

$$\begin{array}{lcl}
\Gamma_L^L & = & 1 \\
\Gamma_L^{L-1} & = & z+2 \\
\Gamma_L^{L-2} & = & (z+2-2)x + 2(y+1) = zx^1 + 2(y+1) \\
\Gamma_L^{L-3} & = & (zx^1 + 2(y+1) - 2)\,x + 2(y+1) = zx^2 + 2(yx + y + 1) \\
\Gamma_L^{L-4} & = & (zx^2 + 2(yx^1 + y + 1) - 2)\,x + 2(y+1) = zx^3 + 2(yx^2 + yx + y + 1)
\end{array}$$

Hence, for every $i,\ 1 \leq i \leq L$ the following holds:

$$\Gamma_L^{L-i} = zx^{i-1} + 2y(x^{i-2} + x^{i-3} + \ldots + x^1 + x^0) + 2 =$$

$$= zx^{i-1} + 2\left(1 + y\sum_{j=0}^{i-2} x^j\right) = zx^{i-1} + 2\left(1 + y\frac{x^{i-1}-1}{x-1}\right).$$

Let $m \triangleq L - i$. Hence $i = L - m$ and for every m, $0 \leq m < L$, the following holds:

$$\Gamma_L^m = zx^{L-m-1} + 2\left(1 + y\frac{x^{L-m-1}-1}{x-1}\right).$$

Since $\Gamma_L^m = 1$ holds for $m = L$, whereas $\Gamma_L^m = 0$ holds for $m \geq L$, we conclude that

$$\Gamma_L^m = \begin{cases} zx^{L-m-1} + 2\left(1 + y\frac{x^{L-m-1}-1}{x-1}\right) & 0 \leq m < L \\ 1 & m = L \\ 0 & m > L\ . \end{cases} \qquad (2)$$

Recall that Γ_L^m has been defined as the number of m-level PGs traversed by a message whose topmost routing level is L. However, Γ_L^m may also be regarded as the number of physical nodes (i.e. 0-level PGs) in m-level PGs, that participate in the routing of the message. This is because every PG traversed by the message has a physical border node that selects the route crossing that PG. Note also that Γ_L^0 represents the number of all physical nodes traversed by the message all the way from the source to the destination.

Definition 2 *An m-level route computation is a computation of a path crossing an m-level PG.*

As shown in Sec.2 the result of an m-level route computation is a list of $(m-1)$-level LGNs.

Consider a message whose topmost routing level is L. Since the route through every PG is represented by a DTL, the sum

$$\sum_{m=1}^{L} \Gamma_L^m$$

indicates the total number of DTL pushes performed by all the nodes that participate in the routing of the considered message. For every PG traversed by the message, there exists a physical border node that determines the route through the PG for the message. A node acting as a physical border node of several levels of the hierarchy has to perform routing and to push a DTL multiple times,

one for each level. For example, node C.1.1 in Figure 3 is a border node of PG(C.1) in level 1, and a border node of PG(C) in level 2 as well. Hence, when it receives from a node in PG(A) a message destined for a node not in PG(C), it needs to perform two routing decisions and to push two DTLs: one for the route that crosses PG(C.1), and another for the route that crosses PG(C); namely,

[C.1.1, C.1.2, C.1.3]

[C.1, C.2, C.3].

Each of these routing decisions requires the border node to calculate an optimal QoS-based path in a certain PG, and to push a DTL representing the selected route into the DTL stack.

Definition 3 *An m-level route computation can only be performed by a physical node in the m-level PG for which the route is computed that is physically connected to another physical node outside that same m-level PG. Such a node will be referred to as an* m-level physical border node.

For instance in Figure 3 A.4.1 is a 1-level physical border node but not a 2-level physical border node and A.4.3 is a 2-level physical border node as well as a 1-level physical border node.

In general, a physical border node to an m-level PG is also a physical border node to a PG in every lower level: $m-1, m-2, \cdots, 1$. When such a node receives a message as an m-level physical border node, it needs to perform m route computations and to push m DTLs into the message routing stack – one for each PG.

The physical source node of a message whose topmost routing level is L, always performs L routing decisions and DTL pushes. For instance the topmost routing layer for a message from A.1.2 to D.3.3 in Figure 3 is 3. Hence the source node A.1.2 has to compute 3 routes and to push 3 DTLs: one for PG(A.1), one for PG(A) and one for PG(*); namely,

[A.1.2, A.1.1, A.1.3]

[A.1, A.3, A.4]

[A, C, D].

In what follows we consider again a message whose topmost routing level is L, and compute the mean number of physical border nodes of PGs in level m that are *not* physical border nodes of PGs in higher levels for the considered message. This number will be denoted by Λ_L^m. Recall that an m-level physical border node that is also a physical border node of some higher level PGs must be an $(m+1)$-level physical border node as well. Hence by subtracting Γ_L^{m+1} from Γ_L^m we get the number of m-level physical border nodes that are not physical border nodes of PGs in levels higher than m. Consequently, for every $0 \leq m \leq L-1$ the following holds

$$\Lambda_L^m = \Gamma_L^m - \Gamma_L^{m+1}. \tag{3}$$

Using Eq.2 we get for every $0 \leq m < L$:

$$\Lambda_L^m = zx^{L-m-1} + 2\left(1 + y\frac{x^{L-m-1}-1}{x-1}\right) - zx^{L-m-2} - 2\left(1 + y\frac{x^{L-m-2}-1}{x-1}\right) =$$

$$= zx^{L-m-1} - zx^{L-m-2} + 2yx^{L-m-2} = (zx - z + 2y)x^{L-m-2}.$$

And in general:

$$\Lambda_L^m = \begin{cases} (zx - z + 2y)x^{L-m-2} & 0 \leq m < L-1 \\ z+1 & m = L-1 \\ 1 & m = L\ . \end{cases} \tag{4}$$

Note that by Eq.3, for every n and L, $0 \leq n \leq L$

$$\sum_{m=n}^{L} \Lambda_L^m = \Gamma_L^n - \Gamma_L^{L+1} = \Gamma_L^n.$$

This is because the sum of the border nodes in levels n or higher when a border node of multiple levels is counted once only – at the highest level – is equal to the number of border nodes at the lowest level (level n).

Note also that Λ_L^0 represents the average number of physical nodes that are traversed by a message whose topmost routing level is L, but make no routing decision for that message.

As already explained, an m-level physical border node which is not also a border node in a higher level PG makes exactly m routing decisions and performs m DTL pushes upon receiving a message. Therefore, for a message whose topmost routing level is L, $m\Lambda_L^m$ is the average total number of routing decisions performed by all the m-level physical border nodes which are not also physical border nodes to higher level PGs. Hence, the total number of routing decisions performed in the network due to a single message whose maximal routing level is L is on the average

$$\sum_{m=1}^{L} m\Lambda_L^m.$$

However, by Eq.4 and Eq.3 we get that

$$\sum_{m=1}^{L} m\Lambda_L^m = \sum_{m=1}^{L} m\Gamma_L^m - \sum_{m=1}^{L} m\Gamma_L^{m+1} = \sum_{m=1}^{L} m\Gamma_L^m - \sum_{m=2}^{L+1} (m-1)\Gamma_L^m =$$

$$\sum_{m=1}^{L} m\Gamma_L^m - \sum_{m=1}^{L+1} (m-1)\Gamma_L^m = \sum_{m=1}^{L} [m-(m-1)]\Gamma_L^m = \sum_{m=1}^{L} \Gamma_L^m.$$

Hence

$$\sum_{m=1}^{L} m\Lambda_L^m = \sum_{m=1}^{L} \Gamma_L^m.$$

This indicates that the total number of routing decisions performed for a message whose topmost routing level is L can be represented by either $\sum_{m=1}^{L} m\Lambda_L^m$ or $\sum_{m=1}^{L} \Gamma_L^m$. To demonstrate this equality, note that 13 routing decisions are needed in order to route the message from A.1.2 to D.3.2 in Figure 3. This number can be achieved in two different ways as follows:

1. Six nodes function as 1-level physical border nodes but not to as 2-level physical border nodes: A.3.1, A.4.1, C.2.1, C.3.1, D.2.1, D.3.1. Each of these nodes perform exactly one routing decision. There are 2 2-level physical border nodes that are not 3-level physical border nodes: D.1.1, C.1.1. Each of these nodes makes 2 routing decisions: one for level 1, and another for level 2. Finally, there is exactly one border node of a 3-level PG - PG(*): node A.1.2. This nodes makes 3 routing decisions. Hence the total number of routing decisions for the considered message is $6 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 = 13$.

2. Nine routing decisions are needed in order to cross 1-level PGs. These decisions are performed by A.1.2, A.3.1, A.4.1, C.1.1, C.2.1, C.3.1, D.1.1, D.2.1 and D.3.1. Three routing decisions are needed in order to cross 2-level PGs. These decisions are performed by A.1.2, C.1.1 and D.1.1. Finally, there is one routing decision for crossing a 3-level PG. This decision is performed by A.1.2. Therefore, the total number of routing decisions in this case is $1 + 3 + 9 = 13$.

4 The Routing Computation Load on a Single Node

Consider a message whose topmost routing level is L. For every $1 \leq m < L$, let Φ_L^m be the average routing burden laid upon the m-level physical border nodes (see Definition 3) due to the considered message. Assume that the total routing load laid upon the m-level physical border nodes is evenly distributed among all the physical border nodes in that level. Under this assumption, whose correction is discussed later, Φ_L^m is equal to the number of route computations performed by m-level physical border nodes due to the considered message, divided by the number of m-level physical border nodes that exist in the L-level PG where the routing is performed.

As shown in the previous section, the numerator of Φ_L^m is $m\Lambda_L^m$ since an m-level physical border node performs m route computations wherever it routes a packet in level m. Consequently, for $1 \leq m < L$

$$\Phi_L^m = \frac{m \cdot \Lambda_L^m}{\Psi_L^m} \tag{5}$$

holds, where Ψ_L^m is the number of m-level physical border nodes that reside in an L-level PG, namely the lowest PG that contains both the sender and the receiver of the message.

Recall that the analysis above is conducted under the assumption that the routing load is evenly distributed among all the m-level physical border nodes in the considered L-level PG. This assumption is quite optimistic, because an m-level physical border node located in the center of the network is likely to be involved in the routing process more often than an m-level physical border node located in the margins of the network. For example, in Figure 3 both C.2.1 and A.1.5 are 1-level physical border nodes. Nevertheless, C.2.1 is likely to perform more routing decisions then A.1.5 because it is traversed by every message going from a node in PG(A) to a node in PG (B), PG(D) or PG(E). However this asymmetry only worsens the imbalance of routing burden distribution discussed in the following.

Note also that in the topmost level of the considered message, level L, the physical "border" node is actually the source of the message. Hence the physical source node can be viewed as a border node between a PG representing "the outside world" and a PG representing the entire network. As an L-level physical entry node, the source is involved in the computation of L routes — one for each level. For example, in Figure 3 A.1.2 is not a border node in any PG. Nevertheless, it needs to compute 3 routes for a message it sends to D.3.3: a route from A.1.2 to A.1.3, a route from A.1 to A.4 and a route from A to D. This implies that there is *no* certain group of L-level physical border nodes, and that the L-level route computations are evenly distributed among all the network nodes. This is the reason for not defining Φ_L^m for $m = L$ above.

To continue with the computation of Φ_L^m, we now express Ψ_L^m as a function to the following 3 new network parameters:

$K =$ The average number of LGNs in a PG, assuming that every PG has the same number of LGNs. Hence, the total number of physical nodes that reside in an L level PG is K^L. For example in Figure 5 a network with $K = 7$ is depicted (the internal structures of PG(A.4), PG(A.5), PG(A.6) and PG(A.7) are not shown).

$\alpha =$ The probability that an m-level LGN is a border LGN in the $(m+1)$-level PG where it resides. In the network depicted in Figure 5 $\alpha = \frac{3}{7}$, since in every PG there are 7 LGNs, and 3 of which are border nodes.

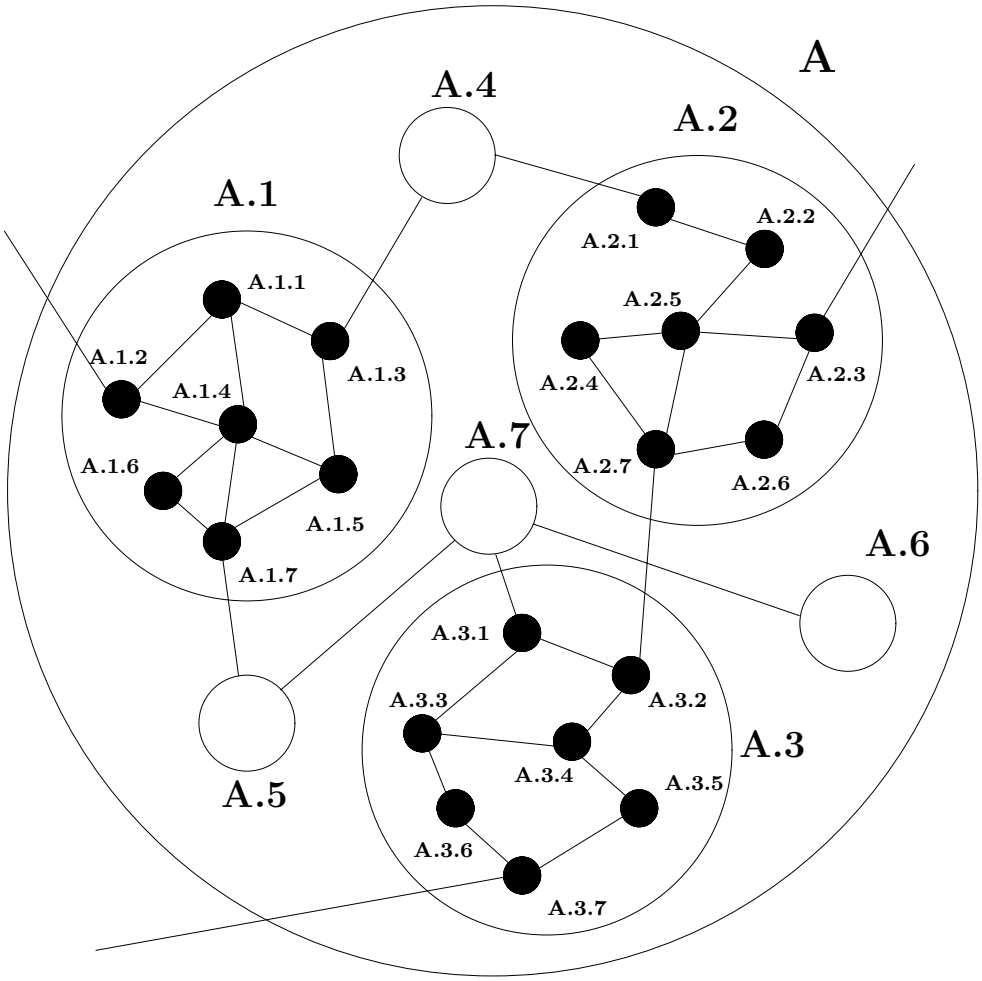

Figure 5: An example of a 2-level PG.

$\beta =$ The probability that an m-level physical border node is an $(m+1)$-level physical border node, given that the m-level PG where it resides is a border LGN inside an $(m+1)$-level PG. For example, the value of β in the network of Figure 5 is $\frac{1}{3}$ since the probability that a 1-level physical border node (like A.1.2, A.1.3, A.1.7, A.2.1, A.2.3, A.2.7, A.3.1, A.3.2 and A.3.7) *that is contained in a 1-level border PG* (like PG(A.1), PG(A.2) and PG(A.3)) is a 2-level physical border node (like A.1.2, A.2.3, and A.3.7) is $\frac{1}{3}$. Note that a 1-level physical border node that is *not* a 2-level physical border node (like A.1.3 and A.1.7) performs only 1-level route computations (see Definition 2).

It is easy to see that $\Psi_1^1 = \alpha \cdot K$, because there are K physical nodes in a 1-level PG, and each has a probability of α for having a neighbor physical node in another 1-level PG. Since an $(m+1)$-

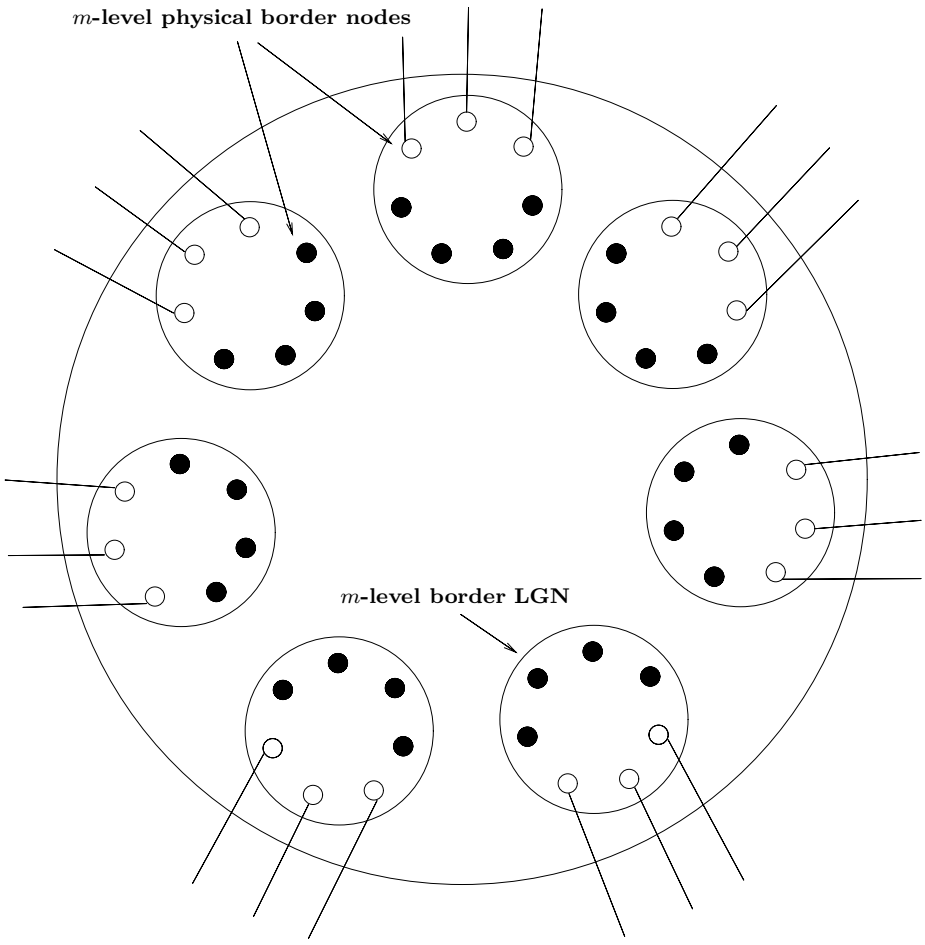

Figure 6: An $(m+1)$-level PG.

level PG contains αK m-level border LGNs (depicted as medium circles in Figure 6), and each of these LGNs has as an m-level PG Ψ_m^m m-level physical border nodes, then by the definition of β for $0 < m \leq L$:

$$\Psi_{m+1}^{m+1} = \beta \cdot \Psi_m^m \alpha K \tag{6}$$

(In term of Figure 6, Ψ_{m+1}^{m+1} is the number of white small circles; each circle represents a switch inside the big circle that is connected to a switch outside the big circle). This is because $\Psi_m^m \alpha K$ is the total number of m-level physical border nodes in an $(m+1)$-level PG, that reside in its αK m-level border PGs (i.e. the total number of small circles black and white, in Figure 6), whereas β is the probability of every such a node to be an $(m+1)$-level physical border node.

Using Eq.6 repeatedly, starting with $\Psi_1^1 = \alpha \cdot K$ we get for $0 < m \leq L$:

$$\Psi_m^m = \frac{(\alpha\beta K)^m}{\beta}.$$

Since each PG has K LGNs, then an L-level PG has K^{L-m} m-level PGs. Hence, the following holds for every $m, 0 < m \leq L$:

$$\Psi_L^m = \frac{(\alpha\beta K)^m}{\beta} K^{L-m} = \frac{(\alpha\beta)^m}{\beta} K^L. \tag{7}$$

Substituting Eq.4 and Eq.7 into Φ_L^m (Eq.5) yields:

$$\Phi_L^m = \begin{cases} \frac{m(zx-z+2y)\beta x^{L-m-2}}{(\alpha\beta)^m K^L} & 0 < m < L-1 \\ \\ \frac{(L-1)(z+1)\beta}{(\alpha\beta)^{L-1} K^L} & m = L-1 \ . \end{cases}$$

For $0 < m < L-1$, Φ_L^m can be algebraically manipulated as follows:

$$\Phi_L^m = \frac{m(zx-z+2y)\beta x^{L-m-2}}{(\alpha\beta)^m K^L} = \frac{m(zx-z+2y)\beta x^{L-2}}{K^L} \frac{1}{(\alpha\beta x)^m}.$$

And therefore for $0 < m < L-1$:

$$\Phi_L^m = C_L m \left(\frac{1}{\alpha\beta x}\right)^m \tag{8}$$

holds, where $C_L = \frac{(zx-z+2y)\beta x^{L-2}}{K^L}$ is a constant whose value depends on the parameters of the network structure only. From Eq.8 follows that for $0 < m < L-1$

$$\frac{\Phi_L^m}{\Phi_L^{m-1}} = \frac{m}{m-1}\left(\frac{1}{\alpha\beta x}\right) \geq \frac{1}{\alpha\beta x}, \tag{9}$$

whereas for $m = L-1$ we have

$$\frac{\Phi_L^{L-1}}{\Phi_L^{L-2}} = \frac{\frac{(L-1)(z+1)\beta}{(\alpha\beta)^{L-1}K^L}}{\frac{(zx-z+2y)(L-2)x^0\beta}{(\alpha\beta)^{L-2}K^L}} = \frac{L-1}{L-2} \cdot \frac{1}{\alpha\beta} \cdot \frac{z+1}{zx-z+2y} \geq \frac{1}{\alpha\beta} \cdot \frac{z+1}{zx-z+2y.}$$

Assuming a symmetric network, where the border nodes are evenly distributed around the non–border nodes, $y \approx z$ holds. Since $x \geq y$, we get

$$\frac{\Phi_L^{L-1}}{\Phi_L^{L-2}} \geq \frac{1}{\alpha\beta} \frac{z+1}{zx-y+2y} \geq \frac{1}{\alpha\beta} \frac{z+1}{zx+x} \geq \frac{1}{\alpha\beta x}. \tag{10}$$

The conclusion from Eq.9 and Eq.10 is that when $\alpha\beta x < 1$, the route computation burden imposed by a message whose topmost routing level is L on the physical border nodes, grows exponentially with the level of the physical border nodes traversed by that message.

Note that apart of that, the ratio between Λ_L^0 (the number of physical nodes which are traversed by the message but are not involved in the routing decisions) and Λ_L^m (the number of m-level physical border nodes traversed by the message), grows exponentially with m by a factor of x. This is true because by Eq.4, for $0 < m < L-1$:

$$\frac{\Lambda_L^0}{\Lambda_L^m} = \frac{x^{L-2}}{x^{L-m-2}} = x^m.$$

In order to estimate the actual routing computation load associated with every switch, we need to determine the topmost routing level of every message. To this end, we study in the following the case where call destination are uniformly distributed among the network levels. That is, for every $1 \leq l \leq N$ the probability that the topmost routing level of a message will be l is $p(L) = \frac{1}{N}$ where N is the highest level in the network.

Since a message whose topmost routing level is l has $K^l - K^{l-1}$ potential physical node destinations, whereas a message whose topmost routing level is $l+1$ has $K^{l+1} - K^l$ potential physical node destinations, namely K times more, the considered uniform distribution implies that nodes located closer to the source have higher probability for being the destination. More specifically, the probability for a physical node of being the destination of a message, assuming the node and the source have lowest common ancestor in level l, decreases exponentially with l. Hence, the considered call distribution fulfills the "locality of calls" concept, where users tend to call local users more often.

Let Δ_N^m be the average routing load laid upon an m-level physical border node due to a single messages sent in the network under the considered call distribution. It is equal to the expected number of route computations performed by all the m-level physical border nodes (under the considered call distribution) divided by the number of m-level physical border nodes in the entire N-level network:

$$\Delta_N^m = \frac{\sum_{l=m+1}^{N} p(l) \cdot m \cdot \Lambda_l^m}{\Psi_N^m} \tag{11}$$

Substituting Eq.4 into Eq.11 we get for $0 < m < N-1$:

$$\Delta_N^m = \frac{p(m+1) \cdot m \cdot (z+1) + \sum_{l=m+2}^{N} \left[p(l) \cdot m \cdot (zx - z + 2y) \cdot x^{l-m-2}\right]}{\frac{(\alpha\beta)^m}{\beta} K^N} =$$

$$= \frac{\frac{m}{N}(1 + z + (zx - z + 2y) \sum_{l=m+2}^{N} x^{l-m-2})}{\frac{(\alpha\beta)^m}{\beta} K^N} = \frac{\frac{m}{N}(z + 1 + (zx - z + 2y)\frac{x^{N-m-1}-1}{x-1})}{\frac{(\alpha\beta)^m}{\beta} K^N} =$$

$$= \frac{\beta m}{N(\alpha\beta)^m(x-1)K^N} \cdot \left[(x-2y-1)+(zx-z+2y)x^{N-m-1}\right]$$

For $0 < m < N-2$,

$$\frac{\Delta_N^{m+1}}{\Delta_N^m} = \frac{m+1}{m} \cdot \frac{1}{\alpha\beta} \cdot \frac{(x-2y-1)+(zx-z+2y)x^{N-(m+1)-1}}{(x-2y-1)+(zx-z+2y)x^{N-m-1}}$$

holds. Recalling Figure 4, we note that $x > 1$ and that $x \geq y + \frac{1}{2}$ (i.e. $x-2y-1 \geq -x$). Hence, for $0 < m < N-2$ we get:

$$\frac{\Delta_N^{m+1}}{\Delta_N^m} \geq \frac{m+1}{m}\frac{1}{\alpha\beta}\frac{-x+(zx-z+2y)x^{N-m-2}}{(x-2y-1)+(zx-z+2y)x^{N-m-1}} \geq \frac{1}{\alpha\beta}\frac{-x+(zx-z+2y)x^{N-m-2}}{(x-2y-1)+(zx-z+2y)x^{N-m-1}}$$

Since by the same rationale $x \leq 2y+1$ (i.e. $0 \geq x-2y-1$) then for $0 < m < N-2$

$$\frac{\Delta_N^{m+1}}{\Delta_N^m} \geq \frac{1}{\alpha\beta} \cdot \frac{-x+(zx-z+2y)x^{N-m-2}}{(zx-z+2y)x^{N-m-1}}$$

holds. Since for $0 < m \leq N-3$, $x^{N-m-2} \geq x$ holds, then for $0 < m < N-2$ we get:

$$\frac{\Delta_N^{m+1}}{\Delta_N^m} \geq \frac{1}{\alpha\beta} \cdot \frac{-x^{N-m-2}+(zx-z+2y)x^{N-m-2}}{(zx-z+2y)x^{N-m-1}} = \frac{1}{\alpha\beta x}\frac{zx-z+2y-1}{zx-z+2y} = \frac{1}{\alpha\beta x}(1-\frac{1}{zx-z+2y}).$$

It turns out that for most "conventional" (i.e. not extremely asymmetric) network structure, $C_N = \frac{1}{\alpha\beta x}(1-\frac{1}{zx-z+2y}) > 1$. For these networks, Δ_N^m grows exponentially by m under the considered call distribution. This implies that *high*-level physical border nodes, are likely to be overloaded by route computation tasks. Moreover as becomes evident from Eq.8 for any other call distribution that gives a lower priority to close destinations Δ_N^m will grow even faster.

5 Load Sharing of Route Computation

In the previous section it was shown that route computation burden is unevenly distributed among the network nodes. There are several possible ways to solve this problem, one of which is presented in this section. The main idea behind the proposed scheme is to transfer route computation tasks from overloaded physical nodes, referred to as *supported nodes,* to underloaded physical nodes, referred to as *supporting nodes.* The supporting nodes compute routes for different QoS parameters and capacities according to specific tasks they are assigned. The results of these computations are sent periodically to the supported nodes. The supported nodes store the received results in a routing database for later use. When such a node is engaged in the setup of a VC, it searches its routing database for an appropriate pre-computed route.

This approach assumes off-line pre-computation of routes with different QoS parameters and capacities between source/destination pairs. This idea, of using idle processor periods for route

computations, is discussed in [2, 5]. It is motivated by the fact that an on-line QoS-based route computation, performed upon the receipt of a setup message, is computationally expensive and can therefore extensively delay the setup process. Here we go one step further and *suggest to transfer route computations tasks in order to utilize idle processor periods of underloaded supporting nodes.*

In the following a distributed algorithm referred to as RCLB (Route Computation Load Balancing) is presented. This algorithm aims at assigning route computation tasks to supporting nodes such that the total route computation load will be distributed more evenly among the network physical nodes. A *route computation task* consists of the following parts: (a) the address of the source and destination LGNs of the requested path; (b) QoS parameters; (c) capacity ; and (d) the ATM address of the supported physical node (or nodes) to which the computed paths should be reported. One supporting node may send a computed path to several supported nodes, if they are all physical border nodes that happen to be connected to the same end point of an aggregated link. Note that this definition is flexible enough to allow the distribution of one path calculation among several supporting nodes according to different QoS parameters or capacity levels.

In addition to route computation load balancing, the proposed approach may also reduce the total route computation load imposed on the whole network. This is because in many cases different physical nodes need to perform the same route computation since they belong to the same border PG. For instance, in Figure 1, switches 1.3.2 and 1.3.3 are two different physical border nodes in PG(1), but both may need to compute a route from PG(1.3) to PG(1.1). The number of different physical nodes that need to perform the same route computation increases for routes crossing high level PGs. Moreover, it is easy to show that running an "all pairs shortest path" algorithm - such as Floyd-Warshell [4] - once, can be more efficient then running a "single source shortest path" algorithm - such as Dijkstra [4] *several* times at different logical border nodes.

A major constraint that should be taken into consideration by the RCLB algorithm is that route computation tasks have to be assigned only to supporting nodes whose local view of the network (see Figure 2) contains sufficient information for the assigned path computation. Since the local view of every physical node contains only the internal graph of its ancestor PGs, route computation tasks should be assigned only to a supporting node located inside the PG of the to-be-computed route. For example, in Figure 3 physical D.3.2 can be the supporting node of D.1.1 for a route crossing PG(D) or PG(*). However, it cannot help D.1.1 concerning the computation of a route in PG(D.1), since the later is not an ancestor PG of D.3.2.

The tasks assignment process of the RCLB algorithm is performed in an increasing order, such

that tasks requiring the computation of routes crossing lower level PGs are assigned to supporting nodes before tasks requiring the computation of routes crossing higher level PGs.

Definition 4 *A task requiring the computation of routes crossing m-level PG will be referred to as m-level routing tasks.*

The RCLB algorithm is solely performed by the PGL (Peer Group Leader) nodes. Hence, an m-level PGL starts assigning m-level routing tasks only after all $(m-1)$-level routing tasks of its $(m-1)$-level PGs are already assigned. For example, in Figure 3 only after the 1-level routing tasks in PGs D.1, D.2, D.3 and D.4 are assigned to supporting nodes, the assignment of 2-level tasks (such as finding a route from D.1.1 to D.3.5) starts. Note that the supported nodes for the 2-level tasks are in this case D.1.1, D.3.5 and D.4.1 because these are the 2-level physical border nodes in PG(D). This process continues until N-level tasks are assigned, and the RCLB is completed.

During each iteration of the RCLB protocol, the assignment of an m-level task list to supporting nodes starts at an m-level PGL. The m-level PGL distributes tasks to its *son PGLs* (i.e. PGLs of its $(m-1)$-level PGs). These PGLs pass the tasks recursively to their son PGLs until the tasks are received by 0-level PGs, which are the actual supporting nodes. For example, in Figure 3 the assignment of 2-level routing tasks in PG(D) is carried out by dividing the task list into four sub-lists and delivering every sub-list from the PGL of PG(D) to the PGLs of PG(D.1), PG(D.2), PG(D.3) and PG(D.4). The latter, in their turn, distribute the received task lists to their internal 0-level PGs.

In order to guarantee equal distribution of the routing load, the PGL tries to divide its task list among its son PGs such that the ratio between the length of every task list and the number of physical nodes in the PG will be equal. A PGL enters the RCLB algorithm after all its son PGLs finish to distribute their own task lists among the physical nodes in their PG. Hence, the first PGLs to enter the algorithm are those in level 1. Upon entering the algorithm, the PGL determines the list of route computation tasks needed for crossing its PG. Then, it invokes a procedure called task-distribution(). This procedure distributes the task sub-lists among the son PGLs. When the procedure is completed, namely after each task has been assigned to a physical node, the PG sends an END indication to its parent PG. Note that an execution of the task-distribution() procedure by an m-level PGL triggers the execution of the same procedure by every descendent PGLs in levels $(m-1),(m-2),\cdots,1$. This implies that during the execution of RCLB, an m-level PGL invokes the task-distribution() procedure $N-m$ times.

The proposed scheme can be implemented in the framework of *active networks* [8]. Under this

model, tasks can be packed into "active" messages that contain, in addition to the task parameters, the actual RCLB code to be executed at the nodes to which the messages are sent.

Another scheme that may solve the problem of uneven routing load distribution, is to use the ATM virtual path (VP) concept [3] in order to increase the number of high level physical border nodes. The idea is to establish virtual paths between underloaded nodes, namely nodes that are *not* physical border nodes of high level PGs, of neighboring high level PGs. For instance a virtual Path can be established between A.1.4 and C.2.2 in Figure 3. This will allow the routing process to be performed not only by physical border nodes but also by VP end points. There are many considerations in the design of a proper VP layout that balances the routing load. The penalty of wrong design because of inefficient routing, may exceed the advantage of computation load balancing. Hence, more study is needed in order to explore this method.

6 Conclusions

The paper has investigated the route computation load laid upon the network nodes by the ATM PNNI routing scheme, assuming a uniform hierarchical network model that has been defined by several topology parameters.

The total number of route computations imposed by every message was expressed as a function of the network topology and the message topmost routing level. In the same way, the number of route computations that need to be performed for crossing a single m-level PG along the message path, for arbitrary m, was computed. The number of nodes where these m-level route computations are performed (refered to as m-level physical border nodes) was also obtained.

Using these results, the paper has shown that in many cases the average number of route computations imposed by a single message on the m-level physical border nodes may grow exponentially with m. It has also shown that the ratio between the number of nodes residing on the message path but not involved in the routing process and the number of m-level physical border nodes (that are involved in m route computations) grows exponentially with m. Finally, the paper has shown that under many practical call distributions, the average number of route computations performed by an m-level physical node may grow exponentially with m. This implies that the route computation load can be unevenly distributed among the network nodes.

The paper has indicated that *high*-level physical border nodes, that connect the end points of links between pairs of high level PGs, are likely to be overloaded with route computations. A scheme that balances the computational burden among the network nodes has been proposed. The

main idea behind the proposed scheme is to utilize idle processor periods in the underloaded nodes by transfering route computation tasks from overloaded nodes to underloaded nodes.

References

[1] ATM Forum PNNI SWG 94-0471R13. *ATM Forum PNNI Draft Specifications*, March 1996.

[2] I. Cidon, T. Hsiao, P. Jujjavarapu, A. Khamisy, A. Parekh, R. Rom, and M. Sidi. Openet architecture specification. 1995.

[3] R. Cohen and A. Segall. Connection management and rerouting in ATM networks. In *INFOCOM*, 1994.

[4] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[5] O. Crochat, J. Le Boudec, and T. Przygienda. A path selection method in ATM using precomputation. In *Broadband Communications, Proceeding of IZS'96*, February 1996.

[6] C. Huitema. *Routing in the Internet*. Prentice Hall, 1995.

[7] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[8] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.