# Topology Aggregation Schemes for Asymmetric Link State Information

Younghwan Yoo, Sanghyun Ahn, and Chong Sang Kim

*Abstract:* **In this paper, we present two algorithms for efficiently aggregating link state information needed for quality-of-service (QoS) routing. In these algorithms, each edge node in a group is mapped onto a node of a shufflenet or a node of a de Bruijn graph. By this mapping, the number of links for which state information is maintained becomes $aN$ ($a$ is an integer, $N$ is the number of edge nodes) which is significantly smaller than $N^2$ in the full-mesh approach. Our algorithms also can support asymmetric link state parameters which are common in practice, while many previous algorithms such as the spanning tree approach can be applied only to networks with symmetric link state parameters. Experimental results show that the performance of our shufflenet algorithm is close to that of the full-mesh approach in terms of the accuracy of bandwidth and delay information, with only a much smaller amount of information. On the other hand, although it is not as good as the shufflenet approach, the de Bruijn algorithm also performs far better than the star approach which is one of the most widely accepted schemes. The de Bruijn algorithm needs smaller computational complexity than most previous algorithms for asymmetric networks, including the shufflenet algorithm.**

*Index Terms:* **Topology aggregation, link state protocol.**

## I. INTRODUCTION

With various multimedia applications appearing everyday, the quality-of-service (QoS) routing becomes more and more important than before, thus each node requires exact link state information such as bandwidth and delay. This information, however, may change frequently, so link state update messages like *Open Shortest Path First* (OSPF) packets and *PNNI Topology Statement Elements* (PTSE) messages are flooded along the Internet or the ATM PNNI hierarchy periodically, yet the amount of these flooding messages becomes a great deal of overhead to the network scalability as the size of a network increases.

For solving this scalable problem, a lot of *topology aggregation* schemes [1]–[9] have been proposed for the ATM network at first, which has a hierarchical structure as in Fig. 1. At the lowest level a total of ten physical nodes (A.1, A.2,$\cdots$, C.3) are divided into three *peer groups* A, B, and C which are connected by some physical links. These physical links are classified into two types: One is *horizontal links* that connect two nodes belonging to the same peer group like the link A.1-A.2, and the other is *outside links* that connect two nodes belonging to different peer groups such as the links A.2-B.1 and A.3-C.3. The nodes connected to external peer groups by outside links are called *border nodes* (A.2, A.3, B.1, B.3, B.4, C.1, C.3). Each peer group is represented by a peer group leader (PGL) node as a logical node at the upper level, which is called a logical group node (LGN). A PGL advertises its internal LGN information to external peer groups not in an exact form but in an abstract form to decrease the amount of link state information. Thus, an ATM node cannot recognize the exact internal topologies and link state information of peer groups other than its own group.

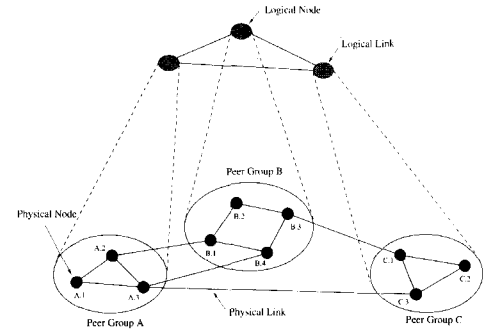



Fig. 1. A hierarchical structure of an ATM network.

In the meantime the abstract form of a peer group should be made carefully, since this abstract information is a basis for nodes to perform a source routing. The way to make an abstract form of a peer group is called topology aggregation, and the performance of the routing algorithms such as the blocking probability and the resource utilization is highly affected by the topology aggregation scheme. Ragozini *et al.* have analyzed the impact of the aggregation scheme and the link-state update frequency on the performance of PNNI networks [10]. They have shown how significantly the inefficient scheme can degrade performance. The topology aggregation, on the other hand, can have a positive impact on routing performance in some cases and can show different performance for different routing algorithms [11].

Many topology aggregation schemes have been suggested to reduce link state information effectively without losing accuracy. The full-mesh [1], the star [1], the spanning tree approach [2], and the Lee method for *asymmetric networks* whose links have asymmetric state parameters in both directions [3] have been proposed for this purpose. However, they have the following shortcomings: (i) the full-mesh approach is not scalable because it has the quadratic spatial complexity $O(N^2)$, (ii) the

Manuscript received September 6, 2002; approved for publication by Raouf Boutaba, Division III Editor, April 9, 2003.

Y. H. Yoo and C. S. Kim are with the School of Computer Science and Engineering, Seoul National University, Seoul, Korea, email: {yhyoo, cskim}@archi.snu.ac.kr.

S. H. Ahn is with the School of Computer Science, University of Seoul, Seoul, Korea, email: ahn@venus.uos.ac.kr.

This work was supported in part by the Ministry of Education under the Brain Korea 21 Project in 2003, and by the Ministry of Science and Technology under the National Research Laboratory program. The ICT at Seoul National University provided research facilities for this study.

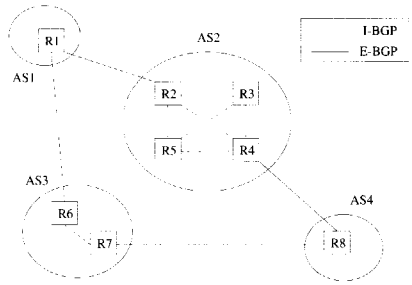



Fig. 2. Autonomous systems connected with BGP sessions.

star approach provides relatively inaccurate abstract information, (iii) the spanning tree approach can be used only in *symmetric networks* whose links have the same resource usage in opposite directions, and (iv) the Lee method requires $O(N^2)$ spatial complexity and excessive computation time.

Therefore, we propose two topology aggregation schemes called the shufflenet approach and the de Bruijn approach, which can be applied to asymmetric networks. This is a very important feature because a practical connection or flow may reserve different amount of resources in opposite directions, resulting in asymmetric link states. For instance, since the amount of output data from a multimedia server is much more than that of input data, reservations of the same QoS for each direction do not make sense. In the proposed schemes, a full-mesh network is transformed into a simple regular graph such as a shufflenet and a de Bruijn graph [12] by mapping border nodes of a peer group onto nodes of a shufflenet or a de Bruijn graph. Because $aN$ ($a$ is an integer and $N$ is the number of border nodes[1]) links are maintained in a shufflenet and a de Bruijn graph, the proposed algorithms are far more scalable than the full-mesh approach with $N^2$ links and do not need excessive computational overhead, either.

Fig. 2 shows the Internet structure using the *Border Gateway Protocol* (BGP) between autonomous systems (ASes). This is similar to the ATM structure in Fig. 1 except that it has no explicit hierarchies: ASes, BGP routers, I-BGP links, and E-BGP links correspond to peer groups, border nodes, horizontal links, and outside links, respectively. Thus, our proposed algorithms can be applied to the Internet as well as ATM networks. In this paper, however, we will focus on describing the operation of our algorithms and comparing the performance with existing methods in ATM, since topology aggregation in other networks has not yet been proposed clearly. Please refer to our previous paper [13] for further information of how the proposed schemes can be used in the Internet.

Through simulation, we show that the accuracy of bandwidth information of the shufflenet approach is close to that of the full-mesh despite the smaller amount of information. Also, its delay information is not significantly different from that of the full-mesh, compared to other methods. On the other hand, although it is not as good as the shufflenet approach, the de Bruijn algorithm also performs far better than the star approach that is most widely used. In addition, the de Bruijn algorithm has smaller computational complexity than most previous algorithms for asymmetric networks, including the shufflenet algorithm.

This paper is organized as follows. Section II describes several aggregation methods proposed in the PNNI specification and their offsprings. In Sections III and IV, we explain two novel aggregation methods using shufflenets and de Bruijn graphs respectively, and in Section V their efficiency is compared with those of the full-mesh and the star approach through simulation. Finally, Section VI concludes this paper.

## II. RELATED WORK

Three basic aggregation methods, a symmetric-node approach, a full-mesh approach, and a star approach, have been proposed in the ATM PNNI specification [1].

The symmetric-node approach aims to reduce the amount of state information. All the nodes in a peer group are merged into a single virtual node, and state information for all the pairs of border nodes is represented by a single common value.[2] This results in the smallest amount of information compared to the following methods. However, the information can be so inaccurate that network resources may be significantly underutilized.

The full-mesh approach focuses on maintaining the accuracy of link state information, so it puts a logical link for every pair of border nodes. In asymmetric networks two logical links are needed for both directions. Depending on the network management policy the state information of the logical links between any two border nodes is derived from the maximum bandwidth path or the minimum delay path among all the paths between the two nodes. The simulation in [4] shows that the full-mesh approach provides much more accurate information and gives better routing performance than all the other approaches other than a non-aggregated one under a uniformly distributed workload. However, it is not scalable to the number of border nodes because it maintains state information of $N(N-1)/2$ or $N(N-1)$ links for symmetric networks or asymmetric networks respectively.

The star approach is a compromise between the first two methods. It puts a virtual node called a *nucleus* in the center of a peer group and connects each border node to the nucleus with a logical link named a *spoke*, so the spatial complexity is $O(N)$. The state information for a logical link is derived from the worst or the average case among the maximum bandwidth paths from the associated border node to the other border nodes. While the symmetric-node approach has only one link state information, each link in this approach can have different state information from the other links, which is called a *spoke exception*. In addition to the lower spatial complexity compared with the full-mesh, the star approach can also represent state information more accurately than the symmetric-node approach, thus becoming widely used compared to the other methods. To minimize the loss of information of the star approach, the ATM Forum suggests that a star topology can have up to $3N$ exception

[1] Although $N$ is actually the number of nodes in a shufflenet or a de Bruijn graph, for convenience' sake it is regarded as the number of border nodes in this paper since the complexities of all the other aggregation methods are represented by the number of border nodes. The number of nodes in the shufflenet or the de Burijn graph is almost equal to the number of border nodes, as shown in Sections III and IV.

[2] Depending on the policy, the common value may correspond to the best, the worst, or the average case of state information.

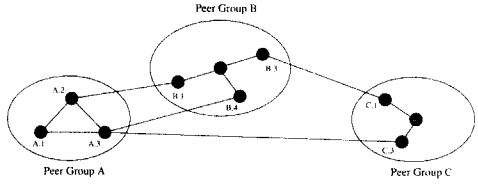



Fig. 3. The topology view from PG A.

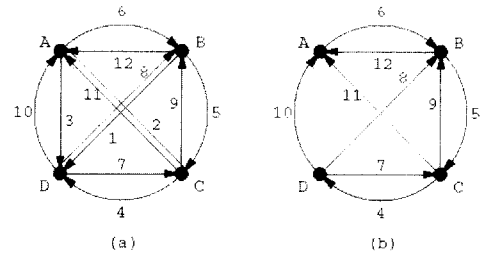



Fig. 4. Lee method for asymmetric networks [3]: (a) original topology, (b) aggregated topology.

links, including not only spoke exceptions but also *bypass exceptions* that connect two border nodes directly. When or where bypasses are established is not standardized but is implementation specific. In [8], the author suggests two algorithms to construct the set of the optimal star topologies in that they use the minimum possible number of bypasses while maintaining information as accurate as that of the full-mesh at the same time. The problem is, however, that they cannot be applied to asymmetric networks.

Fig. 3 shows the view of the Fig. 1 network from a node in peer group A when the other peer groups are aggregated by the star approach without bypass exceptions. Since an ATM node has only aggregated information of other groups, its source routing algorithm represents the path to a destination simply as a sequence of peer groups, not specifying the individual node in each group. Instead, the ingress node in each peer group takes charge of internal routing in its group. If any node along the route cannot support the QoS, the connection setup process is crankbacked to the ingress node in its own peer group first. If even the ingress node cannot find another way, the process goes backward to the upper level until another route is found.

Several enhanced schemes have been recently proposed to overcome the shortcomings of the previous methods. Lee proposed an aggregation method using a spanning tree [2]. This method constructs a full-mesh, and from this full-mesh it generates two spanning trees whose weights are available bandwidth and delay, respectively. After that, link state information of the two spanning trees is advertised to other peer groups so that an external node wishing to establish a connection traversing the aggregated peer group can reconstruct a full-mesh from the spanning trees.

The spanning tree method requires only $2(N-1)$ ($N$ is the number of border nodes) link information which is substantially smaller than the $N(N-1)/2$ link information of the full-mesh approach. Moreover, the accuracy of bandwidth information of the spanning tree method is identical to that of the full-mesh approach. However, it should be noted that the spanning tree method assumes all links are symmetric. As mentioned before, although real networks are asymmetric in general, most previous methods are designed on the assumption of symmetric networks for the sake of simplicity [1], [2].

For asymmetric networks which cannot be aggregated with the spanning tree method, Lee proposed another method [3]. Fig. 4(a) and (b) show peer groups before and after the aggregation respectively. The aggregation procedure is very simple: One link after another is checked to see if it can be replaced by another path with a higher or equal bandwidth. In that case, the link is discarded from the original full-mesh network. For example, in Fig. 4(a) the bandwidth of the link (A, D) is lower than that of the path traversing nodes A, B, C, and D in order. Similarly, the links (A, C) and (B, D) have lower bandwidth than that of the path A-B-C and the path B-C-D, respectively. Thus, those three links (A, D), (A, C), and (B, D) can be deleted from the full-mesh information.

Although this method can maintain the accuracy of state information, the worst case spatial complexity is still $O(N^2)$. In addition, since the time complexity of computation for finding links to be deleted and for reconstructing a full-mesh from the reduced information is so high, this method is not practical in real networks.

In the link state aggregation two types of computation are needed: One for state information reduction and the other for state information reconstruction. For reducing state information, Lee proposed $O(N^3)$ algorithms using the Floyd-Warshall or the Dijkstra algorithm [6]. For reconstructing a full-mesh from the reduced information, however, he did not mention the time complexity, hence we provide it here. For the full-mesh reconstruction, we have to find the maximum bandwidth paths for all the node pairs. In the first place, we should calculate the maximum bandwidth paths from a node to every other node, so we modify the Dijkstra shortest path algorithm [14]. As an example, Fig. 5 shows the modified algorithm to find the maximum bandwidth paths from node 1 to every other node. Because the time complexity is $O(N^2)$ and the number of nodes is $N$, the whole time complexity becomes $O(N^3)$.

Due to the excessive spatial and time complexity of the above methods, Iwata *et al.* suggested a feasible aggregation algorithm using the linear programming in the star topology. By decoupling non-additive QoS metrics (maximum cell rate and available cell rate) from additive metrics (cell loss rate, cell transfer delay, cell delay variation, and administrative weight) their algorithm can maintain aggregation representation more easily [5]. And Korkmaz *et al.* proposed an aggregation scheme based on the full-mesh approach like Lee's methods. However, each border node advertises information about links started from itself only, not the whole full-mesh information [7].

Meanwhile, [9] models the network as directed graphs where links can be asymmetric in the opposite directions to make the representation more flexible. During a process transforming a full-mesh into a star topology, the proposed scheme reduces the amount of state information by approximating all delay-bandwidth pairs of links started from a border node with a line segment.

Another notable scheme for asymmetric networks is the proposal by Awerbuch and Shavitt [15]. The authors focus on the

$b_{ij}$: the bandwidth of the link $(i, j)$
$b_{ij} = 0$ if the link $(i, j)$ does not exist.
$P = \{1\}, B_1 = \infty, B_j = b_{1j}$ for $j \neq 1$
Step 1: Find $i \notin P$ such that
$B_i = \max_{j \notin P} B_j$.
$P := P \cup \{i\}$.
If $P$ contains all nodes, then stop.
Step 2: For all $j \notin P$
$B_j := \max[B_j, \min[B_i, b_{ij}]]$.
Go to Step 1.

Fig. 5. An algorithm for finding the maximum bandwidth paths from node 1 to every other node.

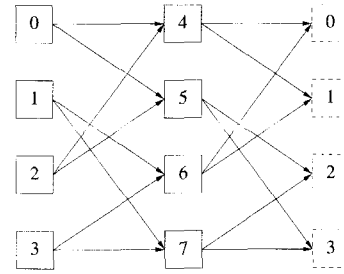



Fig. 6. A (2, 2) shufflenet.

realistic network scenario where all links are bidirectional and might have significantly different costs in the opposite directions. This method initially converts asymmetric networks into symmetric ones whose link weights are the square root of the product of two weights in both directions of asymmetric networks. Then using only border nodes, it constructs a $t$-spanner tree which is a subgraph having the property that distances for all the node pairs are at most $t$ times as long as the distances in the original graph. The $t$-spanner can bound the inaccuracy of delay information by a factor of $t$, if it regards the delay as the weight. However, when the weights in both directions of a link are significantly different, the conversion from an asymmetric network to a symmetric network might cause distortions in terms of network asymmetry.

## III. LINK AGGREGATION USING A SHUFFLENET

For the sake of simplicity, most of the previous approaches assumed that all the links are symmetric. As discussed before, this assumption does not coincide with practical networks, so we propose two aggregation methods for asymmetric networks. We explain the aggregation using a shufflenet in this section, leaving the aggregation using a de Bruijn graph in the following one. Section III-A describes characteristics of a shufflenet, and Section III-B explains the proposed shufflenet aggregation method with an example.

### A. Characteristics of a Shufflenet

A $(p, k)$ shufflenet has $N = kp^k$ nodes ($p$, $k$ = 1, 2, $\cdots$) that are arranged in $k$ columns, each of which has $p^k$ nodes. Each node is assigned a unique identification number; the $l$th node ($l$ = 1, 2, $\cdots$, $p^k$) from the top in the $n$th column ($n$ = 1, 2, $\cdots$, $k$) is assigned the number $(n-1)p^k + l - 1$. Each node has $p$ outgoing links to $p$ nodes in the next column, thus the total number of links in the $(p, k)$ shufflenet is $pN$; the node $i$ has $p$ links connected to the $(j+1)$th, the $(j+2)$th, $\cdots$, the $(j+p)$th node, where $j = (i \bmod p^{k-1})p$, in the next column. Especially, the nodes in the last column have connections to nodes in the first column in a wrap-around fashion. Fig. 6 shows an example of (2, 2) shufflenets.

Each node in a shufflenet has paths to any other nodes, and the hop distance between any two nodes is limited by $(2k - 1)$. In a $(p, k)$ shufflenet, the number of nodes, $n_h$, that can be reached by exactly $h$ hops from one node is given by the following equation:

$$n_h = \begin{cases} p^h & , h = 1, 2, \cdots, k-1 \\ p^k - p^{h-k} & , h = k, k+1, \cdots, 2k-1. \end{cases} \quad (1)$$

Therefore the average hop distance is $\sum_{h=1}^{2k-1} hn_h/(kp^k - 1)$. For example, the average hop distances of (2, 2), (2, 3), and (3, 2) shufflenets are approximately 2, 3, and 2, respectively.

### B. Shufflenet Aggregation

While the full-mesh approach provides the most accurate state information, it should process too much amount of information, especially, for asymmetric networks. Thus, we propose an aggregation method based on the shufflenet which can be applied to asymmetric networks and can reduce the amount of full-mesh link state information without significant degradation of its accuracy. We use a shufflenet topology for the following reasons. First, a shufflenet is suitable for asymmetric network aggregation because each node has directed paths to all other nodes. Second, a shufflenet is more scalable than a full-mesh because it requires only $pN$ logical links to represent a peer group with $N = kp^k$ $(p, k = 1, 2, \cdots)$ border nodes, whereas the full-mesh approach requires $N(N-1)$ links. Finally, after aggregation, overestimation of delay information is not severe because the paths between any two nodes in a shufflenet are bounded by $(2k - 1)$ hops.

Our shufflenet method works as follows:

[Step 1] Link state information of a peer group is aggregated into a full-mesh network,
[Step 2] the full-mesh information is reduced into a shufflenet,
[Step 3] the resulting shufflenet information is advertised to all the other peer groups, and
[Step 4] a node belonging to other peer groups reconstructs a full-mesh from the shufflenet information when it wants to establish a connection passing through the aggregated peer group.

In [Step 2], for the shufflenet construction, $pN$ links from the given full-mesh have to be chosen carefully to maintain the accuracy of bandwidth and delay information after the full-mesh reconstruction in [Step 4]. For the accuracy of bandwidth information, the shufflenet needs to consist of the highest possible bandwidth links because the bandwidth of a logical link between any two nodes in the reconstructed full-mesh is limited by the minimum bandwidth link on the path between the two nodes in

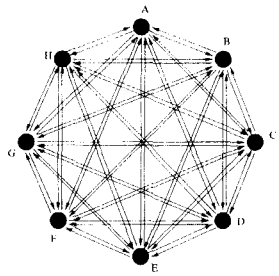



Fig. 7. Full-mesh with eight border nodes.

```
Function Connect(i, j) {
      m := ⌈(i+1)/p^k⌉ p^k + (i%p^(k-1))p ⌋ %(kp^k);
      if j ≥ m and j < m + p then return True;
      else return False;
}
Function Exist(i) {
      for j = 0 to j < kp^k {
          if In(j) = i then C_i := C_i ∪ {j};
          }
      if C_i ≠ φ then return True;
      else return False;
}
```

Fig. 8. Utility functions in our algorithms for a $(p, k)$ shufflenet.

the shufflenet. On the other hand, for the accuracy of delay information, the shufflenet needs to consist of the shortest possible delay links in order that every pair of border nodes in the shufflenet might be connected with the shortest possible paths. This is because the delay of a link between any two nodes in the reconstructed full-mesh is the sum of every link delay on the path between the two nodes in the shufflenet.

Assuming that a peer group with eight border nodes is aggregated into a full-mesh as shown in Fig. 7 ([Step 1]), we transform this full-mesh into a (2, 2) shufflenet in Fig. 6 ([Step 2]). The eight border nodes A-H should be mapped onto nodes 0-7 in the shufflenet appropriately. Since the connections between shufflenet nodes are fixed as described in Section III-A, the logical links to be included in the shufflenet are determined by the way of mapping the border nodes onto the shufflenet nodes. Because eight nodes can be arranged in as many as ${}_8P_8 = 8!$ different orders, we cannot check all the cases to find an optimal mapping case. Thus, we require a simple heuristic algorithm to map the border nodes onto the shufflenet nodes effectively. Depending on aggregation policies, under the shufflenet composition rule, we may choose one of the following three algorithms: (i) the algorithm to maintain bandwidth information as accurately as possible after the two conversion steps (full-mesh $\rightarrow$ shufflenet $\rightarrow$ full-mesh), (ii) the algorithm to maintain delay information as accurately as possible after the two conversion steps, and (iii) the algorithm to consider both bandwidth and delay information at the same time. The first algorithm makes the shufflenet have the highest possible bandwidth links because low bandwidth links can become a bottleneck of all paths. This algorithm first sorts all links of the full-mesh in the descending order of bandwidth, and checks if each link can be involved in the shufflenet one after another. On the other hand, the second algorithm makes the shufflenet have the shortest possible delay links because the long delay links can increase the whole delay of paths. Sorting all the links in the ascending order of delay, it checks if each link can be included in the shufflenet one after another. Finally, the third algorithm assigns nodes in the order of ones with high bandwidth links until each node is assigned just once to a shufflenet. Then the assignment to the residual shufflenet nodes is based on the delay information. Sorting links in the order of low delay, it checks if each link can be involved in the shufflenet one after another.

Before looking into our heuristic algorithms, we define several notations and functions used in the algorithms. $G(V, E)$ and $G_s(V_s, E_s)$ are a full-mesh and a shufflenet graph where $V$ and $E$ are the sets of nodes and edges respectively. $|V|$ and $|E|$ mean the number of elements in sets $V$ and $E$. $e^{xy}$ and $e_s^{xy}$ represent edges between nodes $x$ and $y$ in $G$ and $G_s$, respectively, and $b^{xy}$ and $d^{xy}$ are the bandwidth and the delay of $e^{xy}$. $S_p$ is the number of spare nodes (the difference between $|V_s|$ and $|V|$). $E_{used}$ and $E_{not}$ mean the set of edges that are already checked and the set of edges that are not used yet in each algorithm. $C_i$ is the set of shufflenet nodes onto which border node $i$ is mapped, and $V_{min}$ is the set of border nodes that are least frequently mapped onto the shufflenet. $In(j)$ is the macro to return the border node ID mapped onto shufflenet node $j$. The two functions in Fig. 8 are used in our algorithms. The function *Connect(i, j)* returns "True" if the $(p, k)$ shufflenet nodes $i$ and $j$ are connected, and the function *Exist(i)* returns "True" if the border node $i$ exists in any shufflenet nodes. There are three other functions used in the algorithms but not described here: (i) *Assign_src(i, j)* to assign a source node $i$ to any shufflenet node connected to one of nodes in $C_j$, (ii) *Assign_dst(i, j)* to assign a destination node $j$ to any shufflenet node connected from one of nodes in $C_i$, and (iii) *Assign_both(i, j)* to assign both nodes to the shufflenet.

Our three heuristic algorithms are shown in Figs. 9, 10, and 11. The Mapping algorithm II is almost the same as the Mapping algorithm I, except for Steps 1 and 4. Therefore, we described the two steps only. One potential problem of the shufflenet is that the number of nodes should be $kp^k$ ($k$ and $p$ are integers) since it has a regular topology. We recommend that both $p$ and $k$ should be greater than 1, even though there are no pair $(p, k)$ satisfying $N_b = kp^k$ other than the case where either $p$ or $k$ is 1 ($N_b$ is the number of border nodes.). When $p$ is 1 and $k$ is $N_b$, the spatial complexity is $O(N_b)$ which is equal to that of the star approach, since a $(p, k)$ shufflenet has a total of $pN$ links. On the other hand, when $k$ is 1 and $p$ is $N_b$, it has the same spatial complexity as that of the full-mesh approach, $O(N_b^2)$. As a result, the accuracy of the shufflenet information also equals that of the star and the full-mesh approach, respectively. Hence, to exploit the merits of the shufflenet aggregation, both $p$ and $k$ should be greater than 1, provided $N = kp^k$ is greater than $N_b$ because each border node must be used at least once. Our heuristic algorithms fill $(N - N_b)$ spare nodes using some border nodes multiple times. Note that all shufflenet nodes must be assigned associated border nodes.

We should be careful in choosing one pair of $p$ and $k$ among several pairs satisfying $N = kp^k$, considering the trade-off between the amount of information and its accuracy. Note that $p$ and $k$ have an indirect inverse relationship. First, a large $p$ and

```
Determine p and k such that kp^k ≥ N_b; // N_b: the number of border nodes
S_p := |V_s| − |V|; E_used := φ; E_not := E;
Step 1: if E_not = φ then go to Step 4;
        Find i and j such that b^ij = max_{e^xy ∈ E_not} b^xy;
        E_not := E_not − {e^ij}; E_used := E_used ∪ {e^ij};
Step 2: src := Exist(i); // Compute C_i
        dst := Exist(j); // Compute C_j
        if src=True and dst=True {
            if S_p < 2 or Connect(n,m)_{n∈C_i, m∈C_j}=True then go to Step 1;
        } else if src=True and dst=False {
            if Assign_dst(i,j)=True then go to Step 3;
            else if S_p < 1 then go to Step 1;
        } else if src=False and dst=True {
            if Assign_src(i,j)=True then go to Step 3;
            else if S_p < 1 then go to Step 1;
        }
        if Assign_both(i,j)=True {
            if src=True and dst=True then S_p := S_p − 2;
            else if src=True or dst=True then S_p := S_p − 1;
        }
Step 3: if ∀j ∈ V_s, In(j) ≠ NULL then stop; // The algorithm ends.
        else go to Step 1;
Step 4: for n = 0 to n < kp^k
            if In(n) = NULL {
                V_min := φ;
                if ∀i ∈ V, |C_i| = min_{j∈V} |C_j| then V_min := V_min ∪ {i};
                Find m such that Σ_{j∈V} b^mj = max_{i∈V_min} Σ_{j∈V} b^ij then In(n) := m;
            }
```

Fig. 9. Mapping algorithm I (for the accuracy of bandwidth information).

```
Step 1: if E_not = φ then go to Step 4;
        Find i and j such that d^ij = min_{e^xy ∈ E_not} d^xy;
        E_not := E_not − {e^ij}; E_used := E_used ∪ {e^ij};
Step 4: for n = 0 to n < kp^k
            if In(n) = NULL {
                V_min := φ;
                if ∀i ∈ V, |C_i| = min_{j∈V} |C_j| then V_min := V_min ∪ {i};
                Find m such that Σ_{j∈V} d^mj = min_{i∈V_min} Σ_{j∈V} d^ij then In(n) := m;
            }
```

Fig. 10. Mapping algorithm II (for the accuracy of delay information).

a small $k$ improve the accuracy of information at the expense of increasing the number of links, $pN$. On the other hand, a small $p$ and a large $k$ decrease the amount of information at the expense of degrading accuracy of information. Since a large $k$ has negative effects on the maximum hop count, $(2k-1)$, and the mean hop count, it causes the accuracy of bandwidth and delay information to be degraded. Therefore, when $N$ is not so large, a small $k$ is preferred. On the contrary, when $N$ is so large, a small $p$ is desirable to decrease the amount of information, even though the accuracy degrades. The choice of appropriate $p$ and $k$ is dependent on the administration policies. For simplicity, the mapping algorithms in the former section determine $p$ and $k$ so that $N$ may be as close to the number of border nodes, $N_b$, as possible while $N$ is greater than $N_b$.

Mapping algorithms I, II, and III work in all cases, or they always produce a shufflenet topology from which a full-mesh topology information can be reconstructed. According to the characteristics of shufflenets, each node in the shufflenet has directed paths to all the other nodes. Thus, only if each border node exists on at least one of the shufflenet nodes, the path information between all pairs of border nodes can be calculated. In addition, every shufflenet node should be assigned a border node, since an empty shufflenet node gives no information of links on a path.

Meanwhile, Mapping algorithms I, II, and III guarantee that all border nodes are mapped onto the shufflenet nodes at least once and that each shufflenet node is assigned a border node. First, they determine two values of $p$ and $k$ so that $N = kp^k$ is greater than or equal to the number of border nodes, $N_b$. Then the heuristic algorithms set a variable $S_p = |V_s| - |V|$, which means the number of spare shufflenet nodes. This is used to guarantee that the remaining shufflenet nodes are not less than border nodes which have not yet been assigned.

**1. In algorithms I and II**

When $S_p = 0$, the heuristic algorithms only assign nodes which have never been assigned to the shufflenet before. As a result, all border nodes are guaranteed to be assigned at least once.

When $S_p > 0$, they allow the border node which already exists in the shufflenet to be assigned again and decrement $S_p$.

**2. In algorithm III**

If $S_p > 0$ even after every border node is assigned just once to the shufflenet according to bandwidth information, remaining shufflenet nodes are filled according to delay information.

These node assignment steps continue until all shufflenet nodes are filled or there are no more full-mesh links to be checked. In the case where every shufflenet node is assigned

```
Determine p and k such that kp^k ≥ N_b; // N_b: the number of border nodes
S_p := |V_s| − |V|; E_used := φ; E_not := E;
Step 1: if E_not = φ then go to Step 6;
        Find i and j such that b^ij = max_{e^xy ∈ E_not} b^xy;
        E_not := E_not − {e^ij}; E_used := E_used ∪ {e^ij};
Step 2: src := Exist(i); // Compute C_i
        dst := Exist(j); // Compute C_j
        if src=True and dst=True then go to Step 1;
        else if src=True and dst=False {
             if Assign_dst(i,j)=False then go to Step 1;
        } else if src=False and dst=True {
             if Assign_src(i,j)=False then go to Step 1;
        } else if src=False and dst=False {
             if Assign_both(i,j)=False then go to Step 1;
        }
Step 3: if ∃i ∈ V, Exist(i)=False then go to Step 1;
        else if S_p = 0 then stop; // The algorithm ends.
        E_used := φ, E_not := E;
Step 4: if E_not = φ then go to Step 6;
        Find i and j such that d^ij = min_{e^xy ∈ E_not} d^xy;
        E_not := E_not − {e^ij}; E_used := E_used ∪ {e^ij};
Step 5: if Assign_dst(i,j)=True then S_p := S_p − 1;
        else if Assign_src(i,j)=True then S_p := S_p − 1;
        else if S_p > 1 and Assign_both(i,j)=True then S_p := S_p − 2;
        else go to Step 4;

        if S_p = 0 then stop; // The algorithm ends.
        else go to Step 4;
Step 6: for n = 0 to n < kp^k
             if In(n) = NULL {
                  V_min := φ;
                  if ∀i ∈ V, |C_i| = min_{j∈V} |C_j| then V_min := V_min ∪ {i};
                  Find m such that Σ_{j∈V} d^mj = min_{i∈V_min} Σ_{j∈V} d^ij then In(n) := m;
             }
```

Fig. 11. Mapping algorithm III (using the trade-off between the accuracy of bandwidth and the accuracy of delay information).

a border node, the algorithms stop. On the other hand, in the case where there are no more full-mesh links to be checked, the algorithms proceed to the final step. The algorithms check if any shufflenet node is empty, and an empty node is assigned a border node (which has the maximum outgoing bandwidth for the algorithm I; and has the minimum outgoing delay for the algorithms II and III) among the least used ones. Therefore, a border node that has never been assigned becomes a candidate with the first priority. Owing to this final step, all shufflenet nodes are assigned corresponding border nodes, and each border node is guaranteed to be used at least once.

As mentioned above, if each border node is mapped onto the shufflenet at least once and every shufflenet node is assigned a border node, the full-mesh topology information can be reconstructed. Therefore, our algorithms always perform correctly.

Using the Mapping algorithm I, the border nodes of the full-mesh in Fig. 7 are assigned to the shufflenet nodes as shown in Fig. 12, resulting in the 56 logical links in the full-mesh being reduced into 16 links. When we try to emphasize the accuracy of delay information, we can make a shufflenet using the Mapping algorithm II.

This reduced shufflenet information is advertised to other peer groups, and nodes in other groups derive a full-mesh from the shufflenet when they want to establish a connection going through the peer group aggregated by the shufflenet approach. State information for every full-mesh link except the ones that are not deleted in the prior reduction step is recalculated from the widest-shortest path (i.e., the maximum bandwidth path among the shortest paths) between the two associated nodes in the shufflenet. For instance, the bandwidth of link (E, B) is the minimum between the bandwidth of link (E, C) and the bandwidth of link (C, B), and its delay is the sum of the delay of the two links. We could also consider either of E-H-G-C-B and E-H-F-D-B as a path from E to B, but in this case the delay increases too much. Thus, we take account of paths with a hop count less than $(2k-1)$ only. As mentioned earlier, since the average hop count of a (2, 2) shufflenet is 2, the average delay of a (2, 2) shufflenet is about twice as much as that of the original full-mesh if we assume that the delay is proportional to the hop count.

The shufflenet aggregation method requires less computation than the Lee method for asymmetric networks. First, consider the computation involved in state information reduction. The most complex part of the shufflenet method computation is sorting $N^2$ links in descending order of bandwidth, which requires $O(N^2 \log N)$ operations when using the quicksort, while the Lee method requires $O(N^3)$ operations to locate links to be omitted [6].

Second, when a full-mesh is reconstructed from the reduced information, the shufflenet method requires $O(\frac{N^3}{pk^2})$ operations, while the Lee method requires $O(N^3)$ as shown in Section II. The time complexity of the shufflenet method can be derived by multiplying the complexity of finding paths from one node to every other node by the total number of nodes, $N$. In a $(p, k)$ shufflenet, the number of nodes visited to find paths from a node to all the other nodes is calculated by (2), regardless of where the

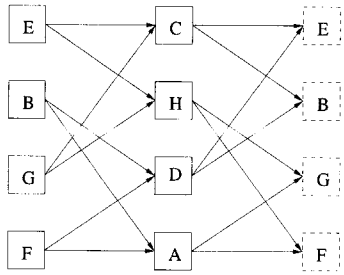



Fig. 12. The result of mapping border nodes onto shufflenet nodes.

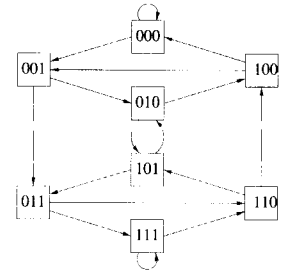



Fig. 13. A (2, 3) de Bruijn graph.

node is located in the shufflenet:

$$f(p,k) = \sum_{h=1}^{k-1} p^h + (p^k - 1) + \sum_{h=k+1}^{2k-1} (p^k - p^{h\%k})p^{h\%k}. \quad (2)$$

First, when the hop count $h$ is less than the number of columns $k$, the number of nodes that can be reached with exactly $h$ hops is $p^h$. Second, when $h$ equals $k$, we return to the column where the originating node is located, and in this case the number of reachable nodes is $(p^k - 1)$ if the originating node is excluded. Finally, when $h$ is greater than $k$, all the nodes at the column ($h$ mod $k$) can be reached. If we exclude those visited in the first step, a total of $(p^k - p^{h \pmod k})$ nodes can be reached in this step. Note that there are $p^{h \pmod k}$ possible paths to each node when the hop counts are greater than $k$. Among them we choose the one with the maximum bandwidth.

The complexity of (2) for finding paths from a node to every other node is $O(\frac{N^2}{pk^2})$, which is significantly lower than the $O(N^2)$ of the Lee method for the same task. For example, when $N$ is 8 ($p$=2, $k$=2), the number of nodes to be visited is 9 in the shufflenet method, while it is 64 in the Lee method. When $N$ is 18 ($p$=3, $k$=2), the numbers are 29 and 324, respectively, and when $N$ is 24 ($p$=2, $k$=3), the numbers are 41 and 576. As $N$ increases, the gap between the two numbers increases rapidly.

The time complexity for the information reconstruction is more important than that for the information reduction, since the reconstruction takes place in all peer groups that need to communicate through the aggregated peer group, whereas the reduction takes place in only one peer group.

## IV. LINK AGGREGATION USING A DE BRUIJN GRAGH

The other proposed method uses a de Bruijn graph [12] instead of a shufflenet. First, in Section IV-A we explain the characteristics of the de Bruijn graph. Second, we describe the motivation of using de Bruijn graphs and the aggregation using the graph in Section IV-B.

### A. Characteristics of a de Bruijn Graph

Like the shufflenet, a de Bruijn graph is represented by a pair of integers, ($\Delta$, $D$). The number of nodes in a graph is $N = \Delta^D$ and each node is assigned an identification number, $a_1a_2\cdots a_D$, where $a_i \in \{0, 1, 2, \cdots, \Delta - 1\}$. And node $a_1a_2\cdots a_D$ has a directed edge that is connected to node $b_1b_2\cdots b_D$, where $a_{i+1}$ is equal to $b_i$, $1 \le i \le D-1$. Therefore, each node has $\Delta$ directed outgoing edges and the total number of links in a ($\Delta$, $D$) de Bruijn graph is $L = \Delta N = \Delta^{D+1}$. Fig. 13 shows a (2, 3) de Bruijn graph.

The important merit of de Bruijn graphs is the routing simplicity. For determining the shortest path from node $A = (a_1a_2\cdots a_D)$ to node $B = (b_1b_2\cdots b_D)$, we should first obtain the match between the last several digits of $A$ and the first several digits of $B$. For example, if $(a_{D-k+1}a_{D-k+2}\cdots a_D)$ is equal to $(b_1b_2\cdots b_k)$, the path is given by $(a_1a_2\cdots a_Db_{k+1}\cdots b_D)$. The path that starts at node $A = (a_1a_2\cdots a_D)$ traverses node $(a_2a_3\cdots b_{k+1})$, node $(a_3a_4\cdots b_{k+2})$, $\cdots$, and reaches node $B = (a_{D-k+1}a_{D-k+2}\cdots b_D) = (b_1b_2\cdots b_D)$ at last. The maximum hop count between any two nodes is $D$.

### B. De Bruijn Aggregation

A de Bruijn graph has some advantages over a shufflenet. First, while a shufflenet has $kp^k$ nodes and $kp^{k+1}$ edges, a de Bruijn graph has $\Delta^D$ nodes and $\Delta^{D+1}$ edges. Thus, compared to the shufflenet aggregation, the de Bruijn aggregation is more flexible to the number of border nodes, in turn being more scalable. For example, if the number of border nodes increases from eight to nine, the shufflenet aggregation comes to use a (3, 2) shufflenet instead of a (2, 2) shufflenet, thus the number of links increases from 16 to 54. On the other hand, the number in the de Bruijn aggregation increases to only 27 links by adopting a (3, 2) de Bruijn graph instead of a (2, 3) de Bruijn graph.

Second, since the maximum hop count between two nodes is limited by $D$, the delay information distortion is not significant and we can easily control the distortion by changing the value of $D$. The mean hop distance, $\bar{h}_{deBr}$, between two nodes is as follows:

$$\bar{h}_{deBr} \le D\frac{N}{N-1} - \frac{1}{\Delta - 1} \quad [12]. \quad (3)$$

The mean hop count of a (2, 3) de Bruijn graph is less than or equal to 2.43.

Similar to the shufflenet aggregation, the de Bruijn method has the trade-off between the amount of information and its accuracy. We can reduce the amount of information by decreasing the value of $\Delta$ since the de Bruijn information has $\Delta N$ links. This, however, increases the value of $D$ because $N$ is equal to $\Delta^D$, in turn the maximum hop count increases and the accuracy of information degrades. On the other hand, a large $\Delta$ and a

Table 1. Complexities to aggregate a PG with $N$ border nodes.

| Agg. method | Spatial | Computational time | |
|---|---|---|---|
| | | Reduction | Reconstruction |
| Lee [3] | $O(N^2)$ | $O(N^3)$ | $O(N^3)$ |
| Shufflenet | $O(pN)$ | $O(N^2 \log N)$ | $O(\frac{N^3}{pk^2})$ |
| de Bruijn | $O(\Delta N)$ | $O(N^2 \log N)$ | $O(N^2)$ |

small $D$ improves the accuracy of information by increasing the amount of information and decreasing the maximum hop count. The choice of $\Delta$ and $D$ depends on the administration policies. For convenience' sake, de Bruijn mapping algorithms in this paper determine $\Delta$ and $D$ so that $N = \Delta^D$ may be as close to the number of border nodes, $N_b$, as possible while $N$ is greater than $N_b$.

Finally, the computational complexity to decode a full-mesh from the de Bruijn information is lower than that of the shufflenet aggregation. For finding paths from a node to all the other nodes, only $(N-1)$ nodes are visited by a breadth first search algorithm because only one shortest path exists between any two nodes. Therefore, when the number of nodes is $N$, the total computational complexity is only $O(N^2)$, whereas the complexity of the shufflenet aggregation is $O(\frac{N^3}{pk^2})$. When $N$ is 8, the number of nodes to be actually visited to reconstruct a full-mesh is 56 in the de Bruijn method, while it is 72 in the shufflenet method. When $N$ is 18, the numbers are 306 and 522, respectively, and when $N$ is 24, the numbers are 552 and 984.

Table 1 compares the spatial and time complexities of the Lee, the shufflenet, and the de Bruijn method in the aggregation of a peer group with $N$ border nodes. For the spatial complexity the shufflenet and the de Bruijn method are superior to the Lee method, and for the time complexity the de Bruijn method is the best. Note that because both $p$ and $k$ in the reconstruction complexity of the shufflenet method are generally greater than or equal to 2, its computation is much simpler than that of the Lee method.

The mapping algorithms from a full-mesh to a de Bruijn graph are similar to those of the shufflenet method in Section III-B. We should just be careful in choosing an available pair of de Bruijn nodes in the function *Assign_both*$(i, j)$. Since nodes whose identification numbers consist of only one digit (i.e., 000, 111, $\cdots$) have a cyclic link, the two end nodes of a chosen link may be eventually a single node.

Like the shufflenet, the de Bruijn graph also should have fixed numbers of nodes, which is determined by $\Delta$ and $D$. However, the number of nodes in a de Bruijn graph, $\Delta^D(\Delta \geq 2, D \geq 2)$, cannot always be equal to the number of border nodes in a full-mesh graph, so we allow a border node to be assigned to multiple de Bruijn nodes.

## V. SIMULATION RESULT

We simulated the proposed shufflenet aggregation and de Bruijn aggregation to evaluate their performance. After aggregating random peer groups with the star, the Lee method for asymmetric networks, the shufflenet, and the de Bruijn approach, we compared the accuracy of their information with that of the full-mesh approach, since the full-mesh approach is known as the one with the most accurate information. Then we analyzed their effectiveness in regard to the spatial and the time complexity. Meanwhile, the spanning tree method was excluded from the simulation because it cannot be applied to asymmetric networks, even though it performs well in symmetric networks.

Using random functions of SMPL [16], we generated 50 peer groups with 30 nodes and asymmetric horizontal links. Each node has four input ports and four output ports, and the bandwidth of all the links in each peer group was determined by exponential distribution with mean 622Mbps.

### A. The Accuracy of Bandwidth Metrics

After selecting one among our 50 sample peer groups, we randomly picked eight nodes among all nodes in the peer group as border nodes. Then using the full-mesh approach we calculated available bandwidth information of logical links between all pairs of border nodes as shown in Table 2. This full-mesh was converted into a shufflenet and a de Bruijn graph by the Mapping algorithm I, and the shufflenet and the de Bruijn graph information was advertised to other peer groups. This shufflenet and de Bruijn graph information was used to reconstruct a full-mesh topology in other groups.

Table 3 displays the bandwidth of each link in the full-mesh which was reconstructed from the shufflenet based on Table 2. The 16 links with underlined values were the links included in the shufflenet after the original full-mesh in Table 2 information was converted to the shufflenet information. A value in parentheses is the ratio of the recomputed bandwidth to its associated bandwidth in the original full-mesh. For instance, '298.44 (94.93)' in the A-B block means that the bandwidth of A-B link in the reconstructed full-mesh is 298.44Mbps, which is 94.93% of that in the original full-mesh shown in Table 2. In cases where the recomputed bandwidth was equal to the original value, we omitted the percentage. Among a total of 56 link values, 14 link values have been changed, and the total bandwidth of the reconstructed full-mesh accounted for 93.35% of the original total bandwidth. In addition, the number of links in the shufflenet is no more than $2N$, where $N$ is the number of border nodes. Consequently, the shufflenet aggregation method could give about the same information as that of the full-mesh approach only with far less amount of information. On the other hand, when the original full-mesh was converted to a star topology with $N$ links, the total bandwidth of the star was no more than 38.72% of the full-mesh. Note that we did not take account of bypass links in the star approach since when and where a bypass is established has not been standardized but implementation specific. Likewise, we considered only basic links of the shufflenet in our approach for a fair comparison with the star, though bypass links can be adopted to reduce error ratios of links such as H-A and E-F, etc.

Table 4 shows the bandwidth information of the full-mesh reconstructed from the de Bruijn graph which was based on Table 2. The de Bruijn graph included only 14 links among the 56 original full-mesh links, and 24 link information was changed more or less from the original values after the full-mesh reconstruction. The total bandwidth of the reconstructed full-mesh accounts for 76.96% of the original total bandwidth. This is still

Table 2. Link bandwidth of a sample original full-mesh (# of border nodes: 8).

| BN | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | 314.39 | 314.39 | 178.61 | 314.39 | 314.39 | 298.44 | 298.44 |
| B | 362.75 | - | 314.39 | 178.61 | 314.39 | 314.39 | 298.44 | 298.44 |
| C | 501.64 | 1065.56 | - | 178.61 | 1743.54 | 606.15 | 298.44 | 298.44 |
| D | 362.75 | 891.03 | 314.39 | - | 314.39 | 314.39 | 298.44 | 298.44 |
| E | 501.64 | 1065.56 | 2385.53 | 178.61 | - | 606.15 | 298.44 | 298.44 |
| F | 121.90 | 121.90 | 121.90 | 121.90 | 121.90 | - | 121.90 | 121.90 |
| G | 501.64 | 983.41 | 983.41 | 178.61 | 983.41 | 606.15 | - | 520.74 |
| H | 501.64 | 785.50 | 785.50 | 178.61 | 785.50 | 606.15 | 785.50 | - |

Table 3. Link bandwidth of the full-mesh reconstructed from the shufflenet based on Table 2 (# of border nodes: 8).

| BN | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | 298.44 (94.93) | 298.44 (94.93) | 121.90 (68.25) | 298.44 (94.93) | 314.39 | 298.44 | 298.44 |
| B | 362.75 | - | 298.44 (94.93) | 178.61 | 178.61 (56.81) | 314.39 | 298.44 | 298.44 |
| C | 362.75 (72.31) | 1065.56 | - | 178.61 | 1743.54 | 314.39 (51.87) | 298.44 | 298.44 |
| D | 362.75 | 891.03 | 314.39 | - | 314.39 | 314.39 | 298.44 | 298.44 |
| E | 362.75 (72.31) | 1065.56 | 2385.53 | 178.61 | - | 298.44 (49.24) | 298.44 | 298.44 |
| F | 121.90 | 121.90 | 121.90 | 121.90 | 121.90 | - | 121.90 | 121.90 |
| G | 362.75 (72.31) | 983.41 | 983.41 | 178.61 | 983.41 | 520.74 (85.91) | - | 520.74 |
| H | 121.90 (24.30) | 785.50 | 785.50 | 121.90 (68.25) | 785.50 | 606.15 | 785.50 | - |

Table 4. Link bandwidth of the full-mesh decoded from the de Bruijn graph based on Table 2 (# of border nodes: 8).

| BN | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | - | 178.61 (56.81) | 178.61 (56.81) | 178.61 | 178.61 (56.81) | 178.61 (56.81) | 178.61 (59.85) | 178.61 (59.85) |
| B | 121.90 (33.60) | - | 314.39 | 121.90 (56.81) | 314.39 | 314.39 | 298.44 | 298.44 |
| C | 121.90 (24.30) | 1065.56 | - | 121.90 (68.25) | 314.39 (18.03) | 298.44 (49.24) | 298.44 | 298.44 |
| D | 121.90 (33.60) | 891.03 | 298.44 (94.93) | - | 314.39 | 298.44 (94.93) | 298.44 | 298.44 |
| E | 121.90 (24.30) | 1065.56 | 2385.53 | 121.90 (68.25) | - | 606.15 | 298.44 | 298.44 |
| F | 121.90 | 121.90 | 121.90 | 121.90 | 121.90 | - | 121.90 | 121.90 |
| G | 121.90 (24.30) | 983.41 | 983.41 | 121.90 (68.25) | 314.39 (31.97) | 606.15 | - | 298.44 (57.31) |
| H | 121.90 (24.30) | 785.50 | 785.50 | 121.90 (68.25) | 785.50 | 606.15 | 298.44 (37.99) | - |

more accurate than 38.72% of the star approach, but it is lower than 93.35% of the shufflenet method. This is because there is only one path between any two nodes in the de Bruijn graph, while there are some candidate paths in the shufflenet. As discussed before, instead, the de Bruijn method has an advantage over the shufflenet method in the aspect of the computational complexity, which is $O(N^2)$ versus $O(\frac{N^3}{pk^2})$.

Table 5 shows the results from the cases where the star, the Lee for asymmetric networks, the shufflenet, and the de Bruijn method were applied to the 50 sample peer groups. The available bandwidth that the star approach represents ranges from 13.61% to 75.88% of the full-mesh bandwidth, and the average is 45.77%. The values in the right half of Table 5 are the numbers of links whose bandwidth information was changed after a full-mesh was reconstructed from the reduced information. Note that the Lee method requires far more computations and might not reduce the amount of information in some cases[3] (its spatial complexity is $O(N^2)$), even though it does not change the bandwidth information of the full-mesh at all. In addition, the simulation results show that the Lee method may distort the delay information significantly compared to other aggregation methods — this will be described later.

The accuracy of bandwidth information in the shufflenet approach ranges from 76.15% to 94.59% of the full-mesh approach, and the average is no less than 85.01%. In addition, the

[3]In our simulations where there are eight border nodes, the number of links in the Lee method ranged from 14 to 32, while always 16 in the shufflenet method regardless of the shape of an original network. In the case of twelve border nodes, the number of links in the Lee method ranged from 44 to 104, whereas 54 or 48 in the shufflenet method depending on the choice between a (3, 2) shufflenet and a (2, 3) shufflenet.

Table 5. Bandwidth information accuracy (# of border nodes: 8).

(Compared to the full-mesh)

| | Bandwidth ratio (%) | | | Links w/ changed BW (#) | | |
|---|---|---|---|---|---|---|
| | Max | Min | Avg | Max | Min | Avg |
| Star | 75.88 | 13.61 | 45.77 | 48 | 16 | 39.13 |
| Lee | 100 | 100 | 100 | 0 | 0 | 0 |
| Shufflenet | 94.59 | 76.15 | 85.01 | 25 | 9 | 17.08 |
| de Bruijn | 89.83 | 53.84 | 78.26 | 33 | 11 | 20.92 |

average number of links whose bandwidth information changed is merely 17.08, while 39.13 in the star approach. For the de Bruijn method, the range of bandwidth accuracy is from 53.84% to 89.83% and the average is 78.26%. Although the two proposed methods have $aN$ ($a$ is an integer, $N$ is the number of border nodes) links, whereas the star approach has only $N$ links, the bandwidth information of them is far more accurate than that of the star for every peer group, and on average, two times. Even compared to the full-mesh approach having $N^2$ links, the shufflenet approach and the de Bruijn approach have error ratios of no more than 15% and 22% respectively, which are negligible if we consider that they are more scalable than the full-mesh approach.

Fig. 14 compares the amount of link state information required for each aggregation method. The vertical axis is the number of links needed to represent a peer group consisting of each number of border nodes in the horizontal axis. The number of links is calculated as $L = kp^{k+1}$ and $L = \Delta^{D+1}$ in a ($p$, $k$) shufflenet and a ($\Delta$, $D$) de Bruijn graph, respectively. The pairs of ($x$, $y$) on the lines for the shufflenet approach and the de Bruijn approach mean that a ($x$, $y$) shufflenet and a ($x$, $y$) de Bruijn graph are used in each method. These $x$ and $y$ are determined so that the total numbers of nodes $N = yx^y$ and $N = x^y$ in a ($x$, $y$) shufflenet and a ($x$, $y$) de Bruijn graph are greater than the number of border nodes as slightly as possible. For instance, when the number of border nodes changes from 25 to 26, a (5, 2) de Bruijn graph is replaced with a (3, 3) de Bruijn graph because the former graph cannot accommodate 26 nodes and the latter one has the closest number of nodes to 26 among all possible de Bruijn graphs. Thus, the number of links in the aggregated form decreases from $L = 5^3$ to $L = 3^4$.

If the number of border nodes is just more than 10 or so, the amount of full-mesh topology information is two times as much as that of the proposed solutions. Considering a wide area network with scores of border nodes, the proposed methods will practically bring a great advantage over the full-mesh approach. The Lee method was not compared in the graph because depending on the shape of peer groups the amount of required information is so fluctuate that the average cannot have a significant meaning. Also, the required computation for the de Bruijn and the shufflenet approach is simpler than that of the Lee method as shown in Table 1.

Fig. 15 shows the variation of bandwidth information accuracy compared to the number of border nodes per peer group. We compare the star, the three shufflenet, and the three de Bruijn approaches using the Mapping algorithms I, II, and III, each of which is labeled by BW, DLY, and BW-DLY, respectively. The x-axis is the number of border nodes in a peer group, and the y-axis is the ratio of the amount of bandwidth in each method

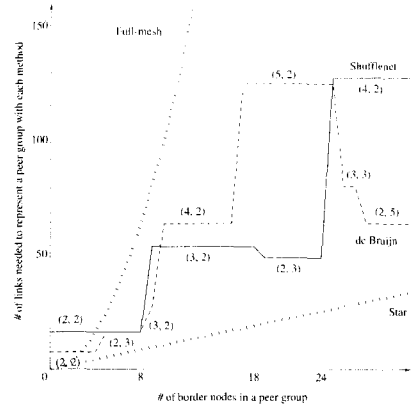



Fig. 14. The amount of link state information.

to the full-mesh which has the second most accurate information next to non-aggregated information. For bandwidth information, the shufflenet method outperforms the other methods in most cases. The figure shows that the performance of the shufflenet and the de Bruijn approaches repeats the graceful degradation and the substantial improvement. For example, the performance of the shufflenet approach worsens continuously until the number of border nodes reaches 8 or 24, and it improves abruptly when the number of border nodes becomes 9 or 25. This corresponds to the change of the amount of link state information in the shufflenet approach in Fig. 14. While the amount of link state information in the full-mesh approach increases quadratically to the number of nodes, that in the shufflenet approach does not change until the number of border nodes is greater than the number of shufflenet nodes. As a result, the ratio of the number of shufflenet links to the number of full-mesh links continuously decreases, which is $2/7$ and $2/23$ when peer groups have 8 and 24 border nodes, respectively, thus the accuracy of link state information continues to be aggravated. Then when the ratio becomes $3/4$ or $16/75$ for 9 or 25 border nodes, the state information becomes almost the same as that of the full-mesh. The variation of the performance of the de Bruijn method can be explained with the same reason. In the comparison of the Mapping algorithms I (BW), II (DLY), and III (BW-DLY), BW performs best for bandwidth information in most cases.

### B. The Accuracy of Delay Metrics

Delay as well as bandwidth is one of the important elements in link state information. We assume that all the physical links have the same delay and that every path delay is proportional to the physical hop count. Table 6 shows the average hop counts of paths in the full-mesh, the star, the Lee, the shufflenet, and the de Bruijn method for five random peer groups PG1–PG5. The values of the shufflenet and the de Bruijn method are the results when the Mapping algorithm I, which focuses on the bandwidth accuracy only, is used. In general, the full-mesh and the Lee method have the smallest and the largest hop count, respectively. Specifically, in contrast to the others, the Lee method has a drawback that it might often distort the delay information too much.

In Sections III-A and IV-B, we calculated the average hop

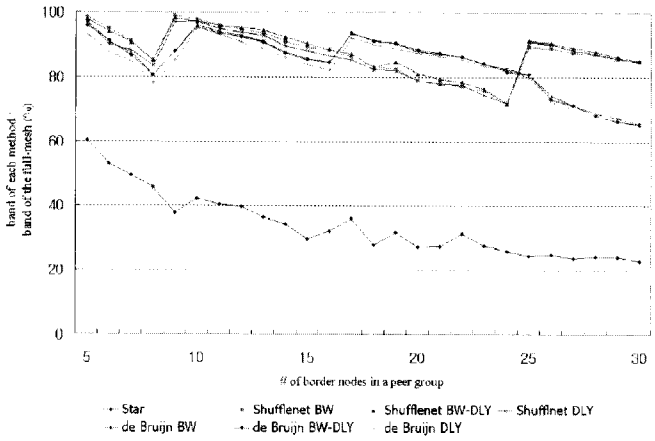



Fig. 15. The accuracy of bandwidth information.

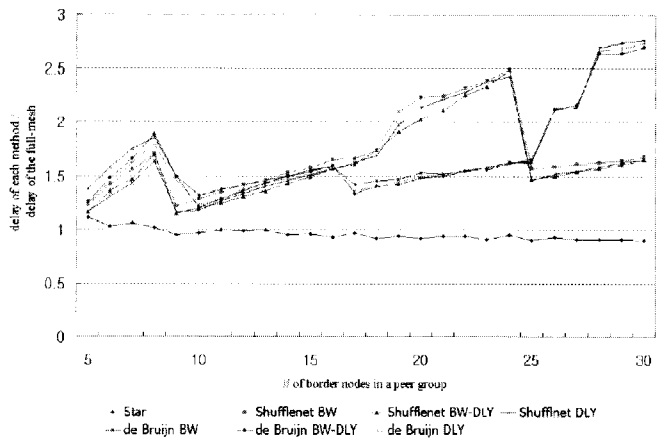



Fig. 16. The accuracy of delay information.

Table 6. Comparison of average hop counts (# of border nodes: 8).

(Unit: Hop count)

| | Full-mesh | Star | Lee | Shufflenet | de Bruijn |
|---|---|---|---|---|---|
| PG1 | 4.95 | 6.62 | 25.79 | 9.79 | 8.55 |
| PG2 | 4.55 | 6.12 | 12.32 | 7.29 | 8.68 |
| PG3 | 4.61 | 4.88 | 14.71 | 7.29 | 7.80 |
| PG4 | 4.27 | 3.62 | 7.75 | 7.79 | 7.86 |
| PG5 | 5.20 | 4.62 | 13.16 | 8.14 | 10.48 |

counts of a (2, 2) shufflenet and a (2, 3) de Bruijn graph, respectively, which are 2 and 2.43. This means that delay of a logical link in the reconstructed full-mesh is the sum of delay of, on average, two or 2.43 concatenated links in the shufflenet or the de Bruijn graph. The simulation results also show that the shufflenet and the de Bruijn hop count are not more than twice the full-mesh hop count. Assuming that every link has the same delay, we can say that the average delay in the shufflenet and the de Bruijn approach is bounded by the product of the average hop count of each method and the delay in the original full-mesh, whereas in the Lee method the degradation of the delay information cannot be bounded at all. Note that the star approach is very ineffective in representing the bandwidth information, although its delay is generally more accurate than those of the shufflenet and the de Bruijn method. This is because the delay information of a spoke in a star depends on only one among the paths, each of which is the maximum bandwidth path from a border node to each other border node. For instance, assume that a peer group has four border nodes A, B, C, and D, and that the pairs of bandwidth and delay information $(b_{ij}, d_{ij})$ from node A to the other nodes are (3, 4), (3, 8), and (12, 6). Because most star approach algorithms use the shortest-minimum path (i.e., the shortest one among the minimum bandwidth paths) as a spoke, the pair (3, 4) is selected for the spoke between node A to the nucleus, and the pair of metrics, in itself, is regarded as the average of all paths starting from node A. On the other hand, the shufflenet and full-mesh approaches maintain all path information, and the averages of bandwidth and delay are (6, 6) because $(3 + 3 + 12)/3 = 6$ and $(4 + 8 + 6)/3 = 6$.

Meanwhile, if we use the Mapping algorithm II that focuses on the accuracy of delay information, we can decrease the hop counts of the shufflenet and the de Bruijn method by 15–20% of the current values.

Fig. 16 shows the variance of delay information accuracy against the number of border nodes. The horizontal x-axis is the number of border nodes in a peer group, and the y-axis is the ratio of the delay of each method to that of the full-mesh information. The graceful degradation and the substantial improvement in the shufflenet and the de Bruijn approaches is caused by the variance of the amount of state information in Fig. 14, similar to the result of bandwidth information in the former section. For instance, when the number of border nodes increases from 24 to 25, a (4, 2) shufflenet is used instead of a former (2, 3) shufflenet. Since a (4, 2) shufflenet has 128 links, which is about three times as much as 48 links of a (2, 3) shufflenet, the greater number of links among 600 links of the full-mesh can remain in the aggregated information than before, resulting in the improvement of accuracy. It is an interesting result that BW-DLY generally performs best for delay information among the three Mapping algorithms, when the number of border nodes is greater than or equal to 10, whereas DLY is the best for delay information only when the number of border nodes is less than 10. Consequently, considering both bandwidth and delay, BW-DLY is recommended since each of bandwidth and delay information is close to the best without distorting the accuracy of the other information.

On the other hand, delay in the star approach becomes slightly shorter as the number of border nodes increases. This is because the delay is derived from the shortest-minimum one among the paths, each of which is the maximum bandwidth path from a border node to each other border node. The more border nodes there are, the more paths with the same minimum bandwidth and different delay exist. Among these minimum bandwidth paths we can select one with the shortest delay. As a result, as the number of border nodes increases, the probability of selecting the more shorter path increases.

### C. The Control of Information Accuracy

As shown in Figs. 15 and 16, while the star approach degrades accuracy of bandwidth information continuously as the number of border nodes increases, the shufflenet and the de Bruijn method bound the worst case performance by control-

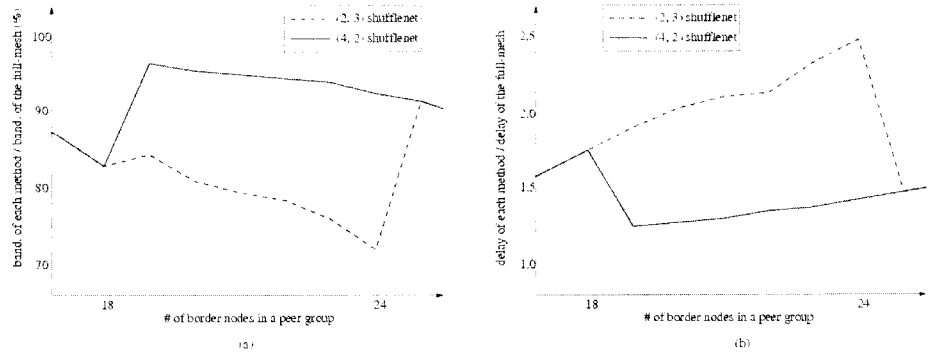



Fig. 17. The comparison between a (2, 3) and a (4, 2) shufflenet: (a) comparison of bandwidth, (b) comparison of delay.

ling the amount of information. Nevertheless, the shufflenet and the de Bruijn approaches give low performance in some ranges in the two graphs. For example, the performance of the shufflenet method is low when the number of border nodes ranges from 19 to 24 (i.e., a (2, 3) shufflenet). This low performance is caused by the fact that $k$ is 3, and in turn, the maximum hop count, $(2k-1)$, is 5, while the maximum hop count of the other shufflenets (2, 2), (3, 2), and (4, 2) is 3. Since the large hop count in a shufflenet may make the state information inaccurate in the full-mesh reconstruction step, we fixed $k$ at 2 even though it increases the amount of information to be delivered. As a result, a (4, 2) shufflenet is used instead of a (2, 3) shufflenet when the number of border nodes is greater than 18.

Fig. 17 compares the performance of a (4, 2) shufflenet with that of a (2, 3) shufflenet, based on the Mapping algorithm III (BW-DLY). Using a (4, 2) shufflenet, the information accuracy is substantially improved when the number of border nodes ranges from 19 to 24.[4] Although a (4, 2) shufflenet requires twice as much information as a (2, 3) shufflenet does, it still requires far less than the full-mesh does. Likewise, if the de Bruijn method gives unsatisfied performance when the number of border nodes is greater than or equal to 28 (i.e., a (2, 5) de Bruijn graph), the problem could be resolved by limiting the value of $D$ which is used as the maximum hop count in the de Bruijn graph.

## VI. CONCLUSION

We proposed the shufflenet aggregation and the de Bruijn aggregation to overcome the problems of previous topology aggregation methods. Because these two methods have $aN$ links, where $a$ is an integer and $N$ is the number of border nodes, they are more scalable than the full-mesh approach which has a total of $N^2$ links. Despite a much smaller amount of information compared to the full-mesh approach, the simulation results show that the accuracy of bandwidth and delay information of the shufflenet method is close to that of the full-mesh. The performance of the de Bruijn method is lower than the shufflenet method, but it is still far better than that of the star. Also, unlike the star approach, the performance of the shufflenet and the de Bruijn approach does not degrade proportionally as the number of border nodes increases, since they can bound the worst case performance by controlling the amount of information.

[4] A (3, 2) shufflenet is used in common until the number of border nodes is 18.

One of the main contributions of the proposed methods is that they can be applied to networks with asymmetric link state information. Contrarily, most previous methods such as the spanning tree method can support only symmetric link state parameters, though practical networks generally have asymmetric link information. Also, compared to the Lee method for asymmetric networks, the proposed methods are far more advantageous due to the fact that they require less computation and do not degrade delay information as much as the Lee method does.

## ACKNOWLEDGMENTS

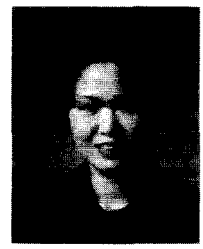
We acknowledge the comments and suggestions by the reviewers. Special thanks to Dr. Eun-kyong Song for correcting grammatical errors in this paper.

## REFERENCES

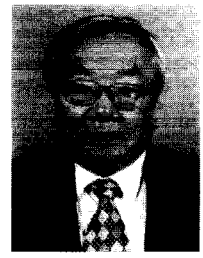
[1] The ATM Forum, *Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)*, 1996.
[2] W. C. Lee, "Spanning tree for link state aggregation in large communication networks," in *Proc. IEEE INFOCOM'95*, 1995, pp. 297–302.
[3] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," *ACM SIGCOMM Compuer Communication Review*, vol. 25, no. 2, pp. 82–92, Apr. 1995.
[4] L. Guo and I. Matta, "On state aggregation for scalable QoS routing," in *Proc. IEEE ATM'98*, 1998, pp. 306–314.
[5] A. Iwata *et al.*, "QOS aggregation algorithms in hierarchical ATM networks," in *Proc. IEEE ICC'98*, 1998, pp. 243–248.
[6] W. C. Lee, "Minimum equivalent subspanner algorithms for topology aggregation in ATM networks," in *Proc. 2nd Int. Conf. ATM'99*, 1999, pp. 351–359.
[7] T. Korkmaz and M. Krunz, "Source-oriented topology aggregation with multiple QoS parameters in hierarchical ATM networks," in *Proc. Int. Workshop QoS*, 1999, pp. 137–146.
[8] I. Iliadis, "Optimal PNNI complex node representations for restrictive costs and minimal path computation time," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 493–506, Aug. 2000.
[9] K. S. Lui and K. Nahrstedt, "Topology aggregation and routing in bandwidth-delay sensitive networks," in *Proc. IEEE GLOBECOM 2000*, 2000, pp. 410–414.
[10] A. R. Ragozini *et al.*, "Analysis of the performance of a hierarchical PNNI networks," in *Proc. 2nd Int. Conf. ATM'99*, 1999, pp. 365–374.
[11] F. Hao and E. W. Zegura, "On scalable QoS routing: Performance evaluation of topology aggregation," in *Proc. IEEE INFOCOM 2000*, 2000, pp. 147–156.
[12] B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, 1997.
[13] Y. Yoo *et al.*, "Border gateway protocol extensions for quality of service routing," in *Proc. Int. Conf. Internet Computing*, 2001, pp. 464–470.
[14] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.
[15] B. Awerbuch and Y. Shavitt, "Topology aggregation for directed Graphs," *IEEE/ACM Trans. Networking*, vol. 9, no. 1, pp. 82–90, Feb. 2001.

[16] M. H. MacDougall, *Simulating Computer Systems Techniques and Tools*, MIT Press, 1987.

**Younghwan Yoo** received the BS and MS degree in Computer Engineering from Seoul National University in 1996 and 1998, respectively. He is now a Ph.D. candidate in the School of Computer Science and Engineering at Seoul National University. His research interests include the routing protocols and the survivability issues in WDM, ATM, Internet, and Ad-hoc networks.



**Sanghyun Ahn** received the BS and MS degrees in Computer Engineering from Seoul National University, Seoul, Korea, in 1986 and 1988, respectively, and received the Ph.D. degree in Computer Science from University of Minnesota in 1993. She is currently an associate professor in the School of Computer Science, University of Seoul, Seoul, Korea. Her research interests include wired and wireless networks, optical networks, Internet protocols, and routing protocols.



**Chong Sang Kim** received the Ph.D. degree in Electronic Engineering from Seoul National University. He is currently a professor in the School of Computer Science and Engineering at Seoul National University, where he has been since 1977. His research interests are in computer architecture and computer networks.