Topology Aggregation Method for Multiple Link Parameters

Sung-eok Jeon

Abstract— **In a hierarchical network, each subnetwork advertises its topology information to the other subnetworks. Many studies have been done to minimize the amount of advertised topology information. The existing works assume a logical link can be represented with only few parameters. However, for the QoS and policy-based networks, more parameters will be used to represent the state of a link (e.g., cost, policies). For these networks, with the existing methods, the amount of advertised information linearly increases to the order of the number of link-state parameters. To solve this problem, this work proposes a new topology aggregation method.**

I. INTRODUCTION

FOR a network with a hierarchical structure, each subnetwork advertises its topology information to the other subnetworks. To reduce the amount of topology-advertising information and to hide the internal topology of subnetworks for security reasons, topology aggregation is done before topology advertisement. Topology aggregation can cause some information loss, however this loss is not so critical in terms of call blocking rate, call access delay, and crank-back rate [3].

A topology aggregation model should represent the original topology adequately for efficient routing and compress the original topology significantly [6]. To keep the original topology information adequately, the conventional full-mesh method is the most preferable one. However, more works are done to compress the topology more significantly: spanning-tree, shuffle-net, source-oriented aggregation method [5], [6], [9]. For example, the spanning-tree method can reduce the number of advertised links from O(n^2) (a full-mesh method) to O(n), where n is the number of border nodes of subnetwork. In case that a logical link of a full-mesh graph is modeled with a vector [BW, DELAY], the spanning-tree method can represent the full-mesh graph with two spanning trees: one for bandwidth and the other for delay. To support a quality-of-service (QoS)-based or policy-based routing, more link-state parameters are to be advertised (e.g., cost, policy, etc). Therefore, more than two link-state parameters are required to model a link in the QoS-supporting, policy-based, multi-class networks.

In the literature, the existing methods advertise a subgraph (e.g., spanning tree, shuffle-net, etc) for each topology parameter. Thus when k topology parameters are used, the existing methods advertise k subgraphs, and the advertised information will linearly increase to the order of the number of topology parameters. To avoid the complexity due to the number of topology parameters, this work proposes a new topology aggregation method, which converts a subgraph into a full-mesh graph, a full-mesh into multiple spanning trees, and each spanning tree into an isomorphic star graph. which is shown in Section III-B. And this work merges these isomorphic star graphs into one merged star graph. With the proposed method, the number of advertised links is always $(n-1)$, and is independent of the number of topology parameters.

S.E Jeon is with the School of Electrical and Computer Engineering, Georgia Institute of Technology. E-mail: sejeon@ece.gatech.edu

II. RELATED WORKS

ATM PNNI 1.0 specification defines three topology aggregation methods (symmetric-node, star, full-mesh approach) [4]. More methods are proposed later [6], [9]. The above methods are described in the following subsections, where each subnetwork has n border nodes.

• Symmetric-Node Approach: the entire subnetwork is represented with a single node, and the worst case parameter becomes the value of advertised parameter. The merit is that this approach clearly offers the greatest reduction of advertising information. The demerits are that it cannot adequately reflect any asymmetric topology information, and that it cannot capture multiple connectivity in the original subnetwork.

A star approach is an extension of the symmetric-node approach. This approach defines a pseudo center node in the subnetwork and a logical link connects the center node and a border node. The merit is that the complexity is linear to the number of border nodes. The demerit is that it cannot capture multiple connectivity in the original subnetwork.

• Full-Mesh Approach: each pair of border nodes of a subgraph are connected with a logical link. The weight of

 0-7803-8794-5/04/$20.00

a logical link is the weight of a representative path, which connects the pair of border nodes. The merit is that it is adequate for efficient routing and resource allocation. The demerit is that the number of advertised links increases as the square of the number of border nodes.

• Spanning-Tree Approach : this approach converts a full-mesh graph into multiple spanning-trees. This approach can be applied to a symmetrical network, and reduce the number of advertised links from $\frac{n\cdot(n-1)}{2}$ (that of the full-mesh approach) to $k\cdot(n-1)$. This approach constructs and advertises a spanning tree for each link-state parameter, so it advertises k spanning trees in case a link is represented with k link-state parameters.

• Shuffle-net Approach : this approach converts a full-mesh graph into a shuffle-net graph. Each border node in a subnetwork is mapped onto a node of a shuffle-net graph. The merit is that it can be applied to asymmetric networks. The number of links of the shuffle-net graph is $p \cdot n$, where p is an integer and n is the number of border nodes, while the number of links of a directed full-mesh graph is n^2.

III. Proposed Merged-Star Graph Method

A. Multiple Subgraphs are Fine?

This work considers the QoS, policy-based, multi-class networks in the next generation Internet (NGI), where a link is represented with multiple link-state parameters. To aggregate the topology of a subnetwork, the existing methods set up a subgraph (e.g., a spanning tree) for each link-state parameter and advertise these subgraphs via topology broadcasting [1],[6],[7],[9]. With these methods, the number of advertised links increases to the order of linear to the number of the link-state parameters. Consequently, for example, when a spanning tree is used as the subgraph, the total number of advertised links is $k\cdot(n-1)$, where k is the number of link-state parameters and n is the number of border nodes.

The vector [BW, DELAY] has been considered enough to represent the state of a link. To support QoS and policy-based services, more link-state parameters will be used to represent the other states of a link (such as delay, cost, policy, etc) [2], [5]. To represent a subnetwork, where a link is modeled with a vector of k link-state parameters, the existing methods will advertise k subgraphs. Thus above a certain threshold value $k \geq k_{th}$, the existing methods will advertise more information than the full-mesh method, whose aggregation is considered as the lower bound. For example, with the spanning tree method, k spanning trees are advertised, and the number of advertised links is $k \cdot (n-1)$. With the full-mesh approach, one full-mesh graph is advertised, and the number of advertised links is $\frac{n\cdot(n-1)}{2}$. Thus for $k > \frac{(n-1)}{2}$, the spanning-tree method advertises more links. This result shows that the existing methods may not be enough in the networks with multiple link states.

B. Converting a Spanning Tree into a Star Graph

This section shows that a weighted spanning tree (n nodes and scalar weight) can be uniquely mapped onto a pre-defined weighted star graph (n nodes and vector weight). The following notations are used in the following *lemma*s. In graph theory, a graph is represented with G={V(G),E(G)}, where V(G) is the set of vertices, and E(G) is the set of edges.

$$\begin{aligned} V(G) = &\quad \{n_1, n_2, \cdots, n_n\} \\ E(G) = &\quad \{\ (n_1, n_2, l_{n_1,n_2}), \cdots,\ (n_{n-1}, n_n, l_{n_{n-1},n_n})\}. \end{aligned} \quad (1)$$

Like Eq.(1), with the most general form, a link can be represented with a vector (n_i, n_j, l_1, l_2, $\cdots$, l_k), where n_i and n_j are two adjacent nodes, and l_is are link weights. For the convenience, each vertex n_i in V(G) is assigned with a unique sequence number, which is denoted with $sn(n_i)$. For example, if a vertex n_i is in the mth sequence (or position) in V(G), $sn(n_i) = m$.

Definition 1: $sn(n_i)$ is the sequence number of vertex n_i in V(G).

Property 1: For any two spanning trees that are subgraphs of a graph with n nodes, a node of a spanning tree is adjacent to at least one link of the other spanning tree.

Proof: A spanning tree connects every node and does not conform any circuit, thus the degree of a node is at least one. As a result, for a node n_i on a spanning tree, it always has an adjacent link (n_i,n_j) on the other spanning tree. ■

Property 2: For a star graph and a spanning tree that are subgraphs of a graph, for a node n_i of the star graph with $i \geq 2$ (n_1 is the root node), at least one adjacent link (n_i, n_j) for $j \neq i$ exists on the other spanning tree.

Proof: A star graph is also a kind of spanning tree. Thus from *property 1*, for a node of a star graph, there is an adjacent link on the other spanning tree. ■

Lemma 1: For a spanning tree that is a subgraph of a graph (G) with n nodes, this spanning tree has (n-1) links. Let denote the set of these (n-1) links with a link set E_1. Let randomly choose (n-1) nodes from V(G) and denote the set of these (n-1) nodes with node set V_1. The set V_1 can be uniquely mapped onto the set E_1.

Proof: Let exclude a node from V(G) and denote it with node n_1: $n_1 = V(G) - V_1$. With the topology information

of the spanning tree, the link set E_1 and the node set V_1 are one-to-one mapped each other, with the following mapping function f(E_1,V_1).

• Mapping function f(E_1,V_1):
Case 1: For an element of E_1, link (n_i, n_j), if this link is an edge link and if the edge node n_i is not n_1, this edge link can be uniquely mapped onto the edge node n_i of V_1. Let denote a node on which a link is mapped with an assigned node. Let denote a spanning tree whose mapped links are deleted from the original spanning tree with the updated spanning tree.
Case 2: For an edge link (n_i, n_j) of an updated spanning tree, if the edge node n_i is n_1 or an assigned node, the edge link can be uniquely mapped onto the node n_j.
Case 3: For an edge link (n_i, n_j) of an updated spanning tree, if the edge node n_i is neither n_1 nor an assigned node, the edge link can be uniquely mapped onto the node n_i.

• Mapping function f(E_1, V_1) provides a one-to-one mapping: From the topology of a spanning tree, the case that the edge node n_i of an edge link (n_i, n_j) is n_1 or an assigned node only happens on the links (n_a, n_b), where n_a is n_1 or a parent node of n_1. Here, the parent nodes of node n_1 are the nodes that are located between the node n_1 and the root node of the spanning tree. Therefore, after mapping according to the Cases 1 and 3, the remaining, unmapped links are only the links (n_a, n_b), where n_a is n_1 or the parent nodes of n_1. If the number of parent nodes of n_1 is P, after updating the spanning tree according to Cases 1 and 3, the the number of unmapped links of E_1 is P and the number of unmapped nodes of V_1 is P too. The unmapped nodes in the spanning tree are the node n_1 and its parent nodes. If each remaining link (n_i, n_j) of E_1 is mapped onto the node n_j of V_1, each remaining link is also uniquely mapped to a remaining node. As a result, the mapping is one-to-one.

■

Lemma 2: For a spanning tree and a star graph that are subgraphs of a graph with n nodes, a link of the spanning tree can be uniquely mapped onto a link of the star graph.
Proof: Assume the root node of the star graph is node n_1. Let denote the set of the nodes of the star graph except n_1 with V_s, and the set of the links of the star graph with E_s. The set V_s can be uniquely mapped onto E_s according to *lemma 1*. Each node element n_i of V_s can be uniquely replaced with a link (n_1, n_i). Let denote this new set of links with E_{s_2}, which is equal to the edge set of the star graph. There is a unique mapping from V_s to E_{s_2}. Because there is a unique mapping from E_s to V_s and from V_s to E_{s_2}, there is a unique mapping from E_s to E_{s_2}. Consequently, there is a unique mapping from a set of links of a spanning tree to a set of links of a star graph.

■

Lemma 3: For a spanning tree and a star graph that are subgraphs of a graph (G) with n nodes, the spanning tree can be mapped onto the star graph, and the star graph also can be mapped uniquely onto the spanning tree by using an additional index with a one-to-one mapping.
Proof: Converting the spanning-tree graph into the star graph (proof from left to right): With the information of V(G), a link (n_i, n_j) of a weighted spanning tree can be represented with a vector (n_i, n_j, l_{n_i,n_j}). According to *lemma 2*, a link of a spanning tree can be uniquely mapped onto a link of a star graph. Thus from *lemma 2*, the link (n_i, n_j) of the spanning tree can be uniquely mapped onto the link (n_1, n_k) of a star graph with the vector (n_1, n_k, l_{n_i,n_j}, ld). Here, l_{n_i,n_j} is the link-state parameter of link (n_i, n_j) of the spanning tree, n_i is the edge node of link (n_i,n_j), n_k is either n_i or n_j, and the value ld is

$$ld = sn(n_j) - sn(n_i). \quad (2)$$

For given spanning-tree and star graph (the root node is n_1), the algorithm that converts the spanning-tree graph into the star graph is:

```
list E0 = edge links of a spanning tree
while (E0 ≠ empty)
    list E = edge links of an updated spanning tree
    while (E ≠ empty)
        choose an edge link e (ni, nj) at the front of list E
        map the link onto the link (n1, ni) of star graph
        update the spanning tree by deleting link e
    end
end
```

Converting the star graph into the spanning-tree graph (proof from right to left): From *property 2*, for a link (n_1, n_i) of the star graph, the node n_i is adjacent to at least one link in the spanning tree. And, from *lemmas* 1 and 2, there is a one-to-one mapping between the link set of a star graph and that of a spanning-tree graph. For a link (n_1, n_i) of the star graph, the corresponding link of the spanning-tree graph is (n_i,n_j). From the link-state vector (n_1, n_i, l_{n_i,n_j}, ld) of the link (n_1,n_i) of the star graph, we can decode the other adjacent node n_j of the corresponding link of the spanning tree: $sn(n_j) = sn(n_i) + ld$. Therefore, for a link of the star graph, the corresponding link of the spanning tree can be decoded from the link-state vector of the link of the star graph.

■

In summary, with *propositions* and *lemmas*, a spanning-tree graph can be mapped onto a predefined star graph with a new vector link cost.

Definition 2: When the link cost of a link of a spanning-tree graph is lc, the link cost of the corresponding link of the predefined star graph is $lc' = (lc, ld)$, where ld is defined in Eq.(2).

An example of one-to-one mapping between a spanning-tree graph and a star graph is shown in Fig.s 1, $\cdots$, 8. A spanning and star graphs are shown in Fig.s 1 and 2 respectively. The star graph is predefined, and the topology information of the spanning-tree graph is en-

coded into the star graph. The encoding procedure starts from the edge nodes of the spanning-tree graph. At the beginning, the given spanning-tree graph has three edge nodes. Each edge node corresponds to the Case 1 of mapping function f(E_1, V_1), which is defined in *lemma 1*. So, as Fig.3 shows, these edge links can be mapped onto the corresponding links with vector link costs, which is defined as $lc^{'}$ in *Definition 2*. After mapping with f(E_1,V_1), the spanning-tree graph (=deleting the mapped nodes and links) can be updated, and the resultant graph is shown in Fig.4. The updated spanning-tree graph has two edge nodes, and each edge node corresponds to the Case 3 of f(E_1, V_1). Thus as Fig.5 shows, these two edge links also can be mapped onto the corresponding links with vector link costs. After updating the previous updated spanning-tree graph, a new updated spanning-tree graph can be gotten in Fig.7. The updated spanning-tree graph in Fig.6 has one edge link, and it belongs to the Case 2 of f(E_1, V_1) (note that the edge node n_1 is the root node of star graph). Thus as Fig.3 shows, these edge links can be mapped onto the corresponding links with vector link costs. The resultant star graph is shown in Fig.7. From *lemma 3*, the procedure of the reverse one-to-one mapping is clear: from the mapped star graph onto the original spanning-tree graph. For an edge node, n_i, in the mapped star graph, its adjacent node can be known from $sn(n_i) + ld$. For example, for the edge node 2 of the mapped star graph in Fig.7, its adjacent node is $2 + (-1) = 1$, the link cost of link (2,1) of the spanning-tree graph is L_2, and so on. The corresponding spanning-tree graph that is decoded from the star graph in Fig.5 is shown in Fig.8.

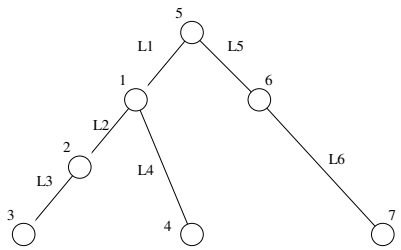

Fig. 1. Spanning Tree Graph

C. Merging Multiple Spanning Trees into a Star Graph

For k link-state parameters, the existing methods use k subgraphs [1], [6], [7], [9]. If k subgraphs are used for k parameters, the number of advertised links increases to the order of linear to the number of parameters. To avoid using k subgraphs, the proposed method constructs k spanning trees (one spanning tree for each parameter), and merges these k

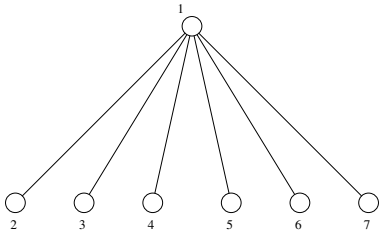

Fig. 2. Predefined Star Graph

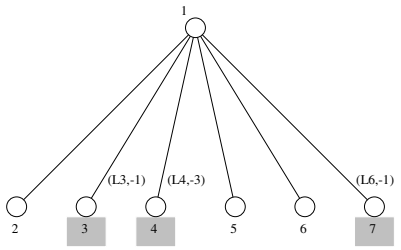

Fig. 3. Encoding (Step 1): Encode the Edge Links of the Spanning Tree Graph

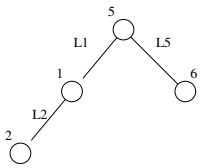

Fig. 4. Encoding (Step 2): Update the Spanning Tree Graph

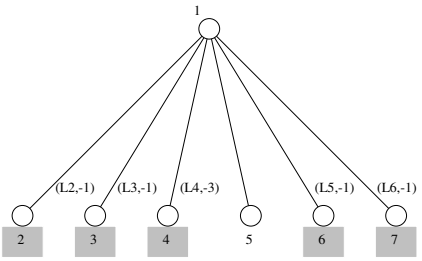

Fig. 5. Encoding (Step 3): Encode the Edge Links of the Updated Spanning Tree Graph

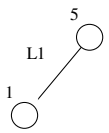

Fig. 6. Encoding (Step 4): Update the Spanning Tree Graph

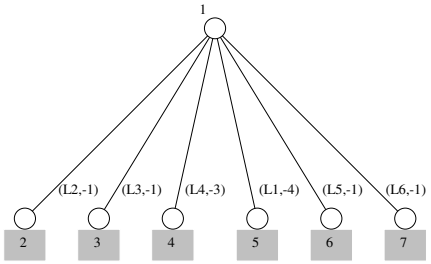

Fig. 7. Encoding (Step 5): Encode the Edge Links of the Updated Spanning Tree Graph

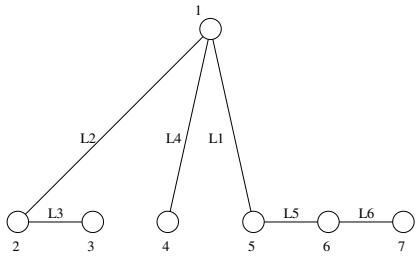

Fig. 8. Decoding

spanning trees into one star graph. The Section III-B shows that a spanning tree can be converted into a predefined star graph. Thus k spanning trees can be converted into k isomorphic star graphs.

Lemma 4: The k weighted isomorphic star graphs can be merged into one weighted star graph.
Proof: If the k isomorphic star graphs are overlapped, the resultant graph is also an isomorphic star graph. The link-state of a link of the overlapped star graph can be represented with a vector, whose ith element corresponds to the link-state of the ith overlapped star graph. Thus the link-state of a link of the merged-star graph is

$$l_{1,j} = \quad ((l^1_{1,j}, ld^1_{1,j}), \cdots, (l^k_{1,j}, ld^k_{1,j})), for\ 2 \le j \le n.$$

■

By converting and merging k spanning trees into a star graph, the number of advertised links is always $(n-1)$ regardless of the number of parameters, where n is the number of border nodes. When converting a spanning tree into a star graph, a weighting l_{n_i,n_j} of a link (n_i, n_j) is converted into a weighting vector $(l_{n_i,n_j}, ld_{n_1,n_k})$ and stored in the link (n_1, n_k) of a star graph, where n_k is either n_i or n_j. Thus a link parameter l of a spanning tree is represented with the link parameter l and an additional index value ld in a star graph.

IV. Complexity of Each Approach in Terms of Total Advertised Links and Information

The spanning-tree method is the most representative one among the methods using multiple subgraphs. Thus this work compares the complexity of the proposed method and the spanning tree method.

• Complexity in terms of Total Advertised Links: for a symmetric subnetwork with n border nodes and k link parameters, with the full-mesh approach, $\frac{n\cdot(n-1)}{2}$ links are needed; with the spanning tree approach, $k\cdot(n-1)$ links are needed; and with the proposed merged-star graph, only $(n-1)$ links are needed.
• Complexity in terms of Total Advertised Information: without losing generality, it can be assumed that a node ID is encoded with m bytes and a link-state parameter is encoded with s bytes. Then, to represent a graph G=(V,E) in Eq.(1), $n\cdot m+e\cdot(2m+s)$ bytes are needed.

Each aggregation method can be compared in terms of the amount of advertisement information. Assume there are k link states at each link and the value ld can be encoded with $\frac{log_2^n}{8}$, which is denoted with ϵ. Then with the multiple spanning-tree method, the total amount of advertisement information is $k\cdot(n\cdot m+e\cdot(2m+s))$ bytes. And with the proposed merged-star method, total $n\cdot(m+\epsilon)+e\cdot(2m+k\cdot s+\epsilon)$ bytes are needed. The difference is $3knm-3nm-2m(k-1)+(1-2n)\epsilon \simeq 3knm = O(knm)$, where $e=n-1$ from the spanning-tree structure.

V. Conclusions

For the topology aggregation problem with k parameters, with the existing methods, k topologies are to be advertised. To solve this communication complexity problem, this work proposes to merge multiple k spanning trees into one merged-star graph, which can handle any number of link-state parameters. As a result, the number of advertised links is always $(n-1)$, and the total advertisement information is decreased to the order of $O(knm)$ compared to the multiple spanning-tree method. Here n is the number of border nodes, k is the number of link parameters, and m is the bytes to encode a node ID.

References

[1] B. Awerbuch, Y. Shavitt, *"Topology Aggregation for Directed Graphs," IEEE/ACM Trans. on Networking*, vol. 9, no. 1, pp. 82-90, Feb. 2001.
[2] S. Chen, K. Nahrstedt, *"An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," IEEE Network*, pp. 64-79, Nov./Dec. 1998.
[3] K. Chung, B. Kim, K. Kim, H. Shin, B. Chon, *"Performance Evaluation of Topology Aggregation in PNNI Network,"* In *Proc. IEEE TENCON*, vol. 1, pp. 261-264, 2001.
[4] ATM Forum Technical Committee, *"Private Network Network Interface Specification Version 1.0 (PNNI1.0), af-pnni-0055.000," ATM Forum Technical Committee*, Mar. 1996.
[5] T. Korkmaz, M. Krunz, *"Source-oriented Topology Aggregation with Multiple QoS Parameters in Hierarchical ATM Networks Quality of Service," IWQoS '99*, pp. 137-146, 1999.
[6] W. Lee, *"Spanning Tree Method for Link State Aggregation in Large Communication Network," ACM SIGCOMM*, pp. 297-302, Apr. 1995.
[7] W. Lee, *"Topology Aggregation for Hierarchical Routing in ATM Networks," ACM SIGCOMM*, Apr. 1995.
[8] R. Tarjan, *"Data Structures and Network Algorithms," SIAM*, 1983.
[9] Y. Yoo, S. Ahn, C. Kim, *"Link State Aggregation Using a Shufflenet in ATM PNNI Networks,"* In *Proc. of IEEE ICC*, vol. 1, pp. 481-486, Jun. 2001.