論文 95-10-10-21

The Segmented Tree Setup (STS) Protocol: A Faster Tree Setup Protocol based on the LPID Scheme

Sanghyun Ahn* Regular Members

STS (Segmented Tree Setup) 프로토콜: LPID 기법을 근간으로 한 신속한 트리 설정 프로토콜

正會員 安 相 炫*

이 논문은 1994년도 세종대학교 대양학술연구비에 의하여 연구한 것임.

ABSTRACT

In providing multipoint connections among group members, multicast tree approaches (shortest path trees and minimal Steiner trees) are known to be the most efficient ways. Segall et. al. pointed out that the size of a routing table can be reduced by advocating the Local Path IDentifier (LPID) scheme and proposed a reliable multicast tree setup protocol based on the LPID scheme. However, their protocol has a defficiency like a long tree setup delay, due to the sequential property of the protocol.

In this paper, we propose an improved tree setup protocol called the Segmented Tree Setup (STS) protocol based on the LPID scheme, whose main objective is to reduce the time for establishing a multicast tree. Our protocol adopts the concept of segmentation by which the maximum delay of a tree setup is bounded to approximately three times the longest among the shortest delays from the tree setup initiator (which is called the originator) to the rest of the nodes in the tree. Also the analysis of the STS protocol performance is provided.

要 約

그룹 멤버들 간의 다중점 연결을 제공하는 데 있어서 멀티캐스트 트리 방법 (최단 경로 트리와 최소 비용 Steiner 트리) 이 가장 효율적인 것으로 알려 졌다. Segall 등은 Local Path IDentifier (LPID) 기법을 사용함으로써 경로 배정표의 크기를 줄일 수 있음을 지적하고, LPID 기법을 근간으로 하는 신뢰성 있는 멀티캐스트 트리 설정 프로토콜을 제안하였다. 그러나 이 프로토콜은 순차적인 특성때문에 트리를 설정하는 데 걸리는 시간이 크다는 단점이 있다.

*세종대학교 전산과학과
Sejong University. Department of Computer Science
論文番號 : 95235-0708
接受日字 : 1995年 7月 8日

따라서 본 논문에서는 LPID 기법을 사용한 Segmented Tree Setup (STS) 프로토콜이라는 향상된 트리 설정 프로토콜을 제안한다. 이 프로토콜의 주 목적은 멀티캐스트 트리를 설정하는 시간을 단축하는 데 있다. STS 프로토콜은 세그멘트 개념을 채택함으로써 트리 설정에 드는 최대 지연을 트리 설정 요청자 (또는 originator) 로부터 트리내의 다른 노드로의 최단 지연중에서 가장 큰 지연의 세 배로 한정한다. 또한 STS 프로토콜의 성능에 대한 분석도 제공한다.

Ⅰ. Introduction

In recent years, Broadband Integrated Services Digital Networks (BISDNs) and its applications have been extensively studied [2, 6, 10]. Those applications which are expected to be promising over BISDNs are group-based multimedia applications like video conferencing [7, 8, 14]. *Multicast tree* approaches have been recognized as the most efficient ways of establishing multicast connections among group members [4, 9].

Some of the most common characteristics of the multimedia applications are constant high bit rate and real-time traffic, which might require multiple number of multicast trees for one multicast group [1]. Intuitively, we know that more routing table entries are demanded in multiple multicast tree schemes and it is undesirable to have a large number of multicast trees especially in a large network since maintaining a routing table is too costly. As Segall *et. al.* point out in [15, 16], by advocating Local Path IDentifier (LPID) based routing table scheme, the routing table size can be reduced. The LPID is similar to Virtual Path Identifiers/Virtual Channel Identifiers (VPIs/VCIs) of Asynchronous Transfer Mode (ATM) networks, since VPIs/VCIs do not have end-to-end meaning [3, 12]. Hence we may expect the LPID-based multicast tree setup protocol to be utilized in ATM networks.

Segall et. al. propose a multicast tree setup protocol based on the LPID scheme in [16] (from now on, we call Segall *et. al.*'s tree setup protocol the SBO protocol). The SBO protocol establishes a multicast tree in a sequential manner starting from a node (*the originator*), which initiates the tree setup process, along the tree, which results in a long delay. However, in some cases, a multicast tree is required to be established as fast as possible. For instance, remotely-located members of an organization may want to discuss a very urgent matter via a teleconferencing system. Therefore, we propose a faster tree setup protocol, the Segmented Tree Setup (STS) protocol, which segments a given tree and uses the LPID scheme. We can bound the tree setup time to three times Δ by chopping up a tree into several segments (here, a segment is a subtree of a given tree) such that the maximum delay from the originator to any one in the segment is less than or equal to Δ.

In Section Ⅱ, we describe two routing table schemes, the GPID and the LPID schemes, and the shortcomings of the SBO protocol and provide our motivations and objectives. Section Ⅲ describes how a multicast tree is divided into segments. The STS protocol is proposed in Section Ⅳ. And the performance analysis of the STS protocol is presented in Section Ⅴ. Section Ⅵ concludes this paper.

Ⅱ. Problem Description

There are two types of routing table schemes, the Global Path IDentifiers (GPID)

scheme and the Local Path IDentifiers (LPID) scheme [11]. In the GPID scheme, all links of a tree are labeled with the same identifier, and this name is unique in the entire network.

The LPID scheme uses names with only local meaning. When a packet reaches a switch node, the label in its header is used for copying the packet to all outgoing links that match this label. Then, before the packet is transmitted to the adjacent node, the label is swapped with the label that is designated for this tree by the adjacent node. Figure 1 shows an example of how the LPID scheme works. In Figure 1, a circle with a capital letter means a network node and a box with a lower case letter an end node. The solid lines are those links included in the tree. Each box consisting of two small boxes represents a routing table entry. A number next to an entry implies the LPID chosen by

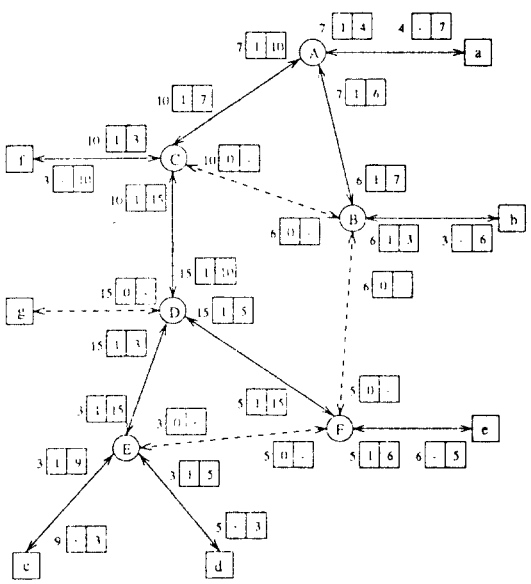

Figure 1. LPID entries for a given conversation

that node for the conversation. And the number in the left small box of the routing table entry indicates whether the corresponding link is part of a given tree. 1/0 implies the link is included/not included in the tree. The right small box of the routing table entry contains the LPID used for this conversation by the node at the other end of the link.

In [16], Segall *et. al.* propose a multicast tree setup protocol using the LPID scheme. The SBO protocol consists of three phases, START, REPLY, and CONFIRM. The originator sends out START messages to its children on that tree along with the whole tree information. Once each child of the originator receives a START message, it selects an available LPID for that tree and sends down a START message to its children along with the tree information and the chosen LPID. This process continues until every end node of the tree receives a START message. If an end node receives a START message from its parent, it selects an LPID and sends back a REPLY message with its own LPID information. Each node sends up a REPLY message to its parent with its LPID, once it receives REPLY messages from all of its children. This continues until the originator receives REPLY messages from all of its children. Then the originator sends CONFIRM messages to its children in order to inform that the tree has been set up. Each node in the tree do the same when it receives a CONFIRM message and can begin to multicast its data on the tree. Therefore, the SBO protocol suffers from a long delay to set up a multicast tree. The worst delay experienced by using the SBO protocol is three times the depth of the tree.

In this paper, we propose a tree setup protocol, the Segmented Tree Setup (STS) proto-

col, whose objective is to reduce the tree setup time. The STS protocol is based on the LPID scheme and divides a given multicast tree into several segments such that the tree setup time is bounded to three times Δ.

Ⅲ. Description of the Segmentation

Due to the fact that an LPID is selected locally at a site and used by only itself and its neighbors (parent and children) in the tree, the sufficient condition to establish a tree correctly is for each node in the tree to receive its neighbors' LPIDs successfully. Hence, the smallest unit of the tree setup is a node and its neighboring nodes in the tree.

Based upon this reasoning, we divide a multicast tree into several segments in which at least the smallest unit of the tree setup is kept and each segment works as if it is an independent tree (i.e., each segment is set up independently of other segments). However, it is inevitable for some nodes in a segment to send/receive LPID information to/from its adjacent segment(s).

In the SBO protocol, nodes are classified into two types, network nodes and end nodes. The end node which initiates the tree setup process is characterized as the originator. By adopting the concept of segmentation into a tree setup protocol, network nodes are to be categorized into several virtual node types, i.e., *root, middle, and boundary* nodes, according to their locations/roles in a segment. The following is the description of virtual network node types:

Root Node The root node of a segment is the network node which directly interacts with the originator.

Middle Node Those network nodes which are neither the root node nor boundary nodes are middle nodes of a segment.

Boundary Node Boundary nodes of a segment are network nodes at the bottom of the segment and are also included in its adjacent segments as the parents of the adjacent segments' boundary nodes.

For the sake of formality, the following notations are used:

- G : a network, $G = (V, E)$, where V is the set of nodes and E the set of edges in G, $e_{ij}, e_{ji} \in E$ if nodes i, j are adjacent
- T : a multicast tree, $T = (V_T, E_T)$, where V_T is the set of nodes and E_T the set of edges in T
- Or : the originator of the tree setup protocol
- S : the set of segments of T
- s_i : the ith element of S, for $i = 1$ to $|S|$, s_i^V is the set of nodes and s_i^E the set of edges in s_i, $V_T = \cup_i s_i^V$ and $E_T = \cup_i s_i^E$
- R : the set of the roots of S
- r_i : the ith element of R, i.e., the root of s_i
- $\delta_G(n_i, n_j)$: the shortest delay from n_i to n_j in G, where $n_i, n_j \in V_T$
- $\delta_T(n_i, n_j)$: the shortest delay from n_i to n_j in T, where $n_i, n_j \in V_T$
- Δ : $\Delta = \max \delta_G(Or, n_i), \forall n_i \in V_T$
- D : $D = \max \delta_T(Or, n_i), \forall n_i \in V_T$
- $\delta_s(Or, r)$: $\delta_s(Or, r) = \delta_G(\mathrm{Or}, \mathrm{r})$, where r is the root of segment s
- $\delta_s(Or, n_i)$: $\delta_s(Or, n_i) = \delta_G(Or, r) + \delta_T(r, n_i)$, where r is the root of segment s and $n_i \in s^V$
- $\delta_s(n_i, n_j)$: $\delta_s(n_i, n_j) = \delta_T(n_i, n_j)$, where $n_i, n_j \in s^V$

The algorithm for the segmentation works as follows.

(Step 1): Let a given tree be T.

(Step 2): Find a core segment s which covers the lowest level node with the largest number of nodes in T. Let $T_{old} = T$ and $T = T_{old} - s$.

(Step 3): Adjust s such that T is not partitioned and the adjusted s is as large as possible.

(Step 4): Repeat (Step 2) to (Step 3) until T becomes null.

At (Step 2), a *core segment* rooted at node n_i consists of the nodes n_j's in T whose $\delta_G(Or, n_i)+\delta_T(n_i, n_j)$ is less than or equal to Δ. At (Step 2), if there are more than one lowest level nodes, the core segment which covers the largest number of the lowest level nodes is chosen. If there's a tie, that one which covers the largest number of nodes is selected. At (Step 3), s is modified such that the number of links in T_{old} not belonging to either T or s is 1. Once a core segment is formed, a segment can be made from it by combining the core segment and its boundary nodes which are adjacent to the core segment.

The example of performing the segmentation algorithm is shown in Figure 2. In this example, delay is assumed to be measured in terms of the number of hops. *Or* is node *e*, $\Delta = 4$, $D = 6$, and the node at the lowest level is node *b* whose level is 6 assuming that the level of *Or* is 0. At the first iteration, since only the core segment 1 rooted at node *B* covers node b, the segment 1 is chosen and has nodes *A*, *B*, *C*, a and b as non-boundary nodes, and *D* and *f* as boundary ones (in Figure 2, this core segment 1 is surrounded by a dashed line). However, the core segment 1 partitions T, so it is adjusted such that T is not partitioned. The adjusted segment 1 has non-boundary nodes *A*, *B*, *a* and *b*, and a boundary node *C* (in Figure 2, this adjusted core segment 1 is surrounded by a dotted line).

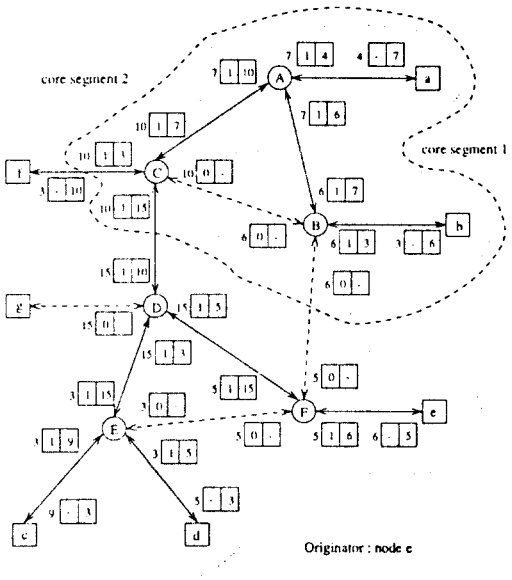

Figure 2. Example of the segmentation

At the second iteration, the core segment 2 rooted at node e covers T, hence, in this example, two segments are generated.

Ⅳ. The Segmented Tree Setup (STS) Protocol

The STS protocol consists of three phases like the SBO protocol, which sends START, REPLY, and CONFIRM messages in that order. What differentiates the STS protocol from the SBO protocol is that in the STS protocol the tree setup operation is carried out concurrently in segments almost independently of one another.

The STS protocol is presented in Figure 3. At the beginning, if an end node receives a tree setup request message from its higher layer with the multicast tree information, it becomes the *originator* and performs the seg-

mentation. The originator sends the START messages to its children, which belong to its own segment, and the root nodes of other segments. Once a node receives a START message from its parent, it chooses its own LPID and propagates START messages down to its children and this is repeated until an end or boundary node receives a START message. If an end node receives a START message, it just sends a REPLY back to its parent.

When a boundary node i receives a START message from node n, if $LPID_i = -1$, it chooses an LPID and sends back a REPLY to its parent. Since $LPID_i \neq -1$ implies node i's LPID has been selected by a START message from another segment s and a START message from i has already been propagated to node n and at n this START message has been treated as a REPLY (called as a virtual REPLY), therefore it's unnecessary to send back a REPLY to n (refer to *L1* and *L2* in Figure 3). However, in this case, node i has to check whether it needs to transmit a REPLY to node m which is node i's parent in segment s where i acts as a non-boundary node. This is necessary because node i does not know when to transmit a REPLY back to node m if node i receives a START from m earlier than some of the STARTs from its children which are the boundary nodes connected to i in segment s (refer to *L2*). And note that, at a boundary node i, the START message from node n sets $flag\_reply_n$ to 1.

If a non-boundary node i receives a START message from node n, first it checks whether its LPID has been chosen by looking at the value of $LPID_i$. If $LPID_i = -1$, it selects an LPID and sends down a START message to all of its children (refer to *L3*). Otherwise (i.e., $LPID_i \neq -1$), it sends a START message to each one of its children from whom it has not received a virtual REPLY and checks whether it has received REPLYs from all of its children, in that case it sends up a REPLY to its parent (refer to *L4*). This is due to the fact that an LPID having been chosen means at least one START message from other segments has been processed at node i prior to the START message from node n and those prior START messages are handled as REPLY (i.e., virtual REPLY) messages at i.

Once a node receives REPLYs or virtual REPLYs from all of its children, i.e., all except one (its parent) of the $flag\_reply(j)$'s are 1's, it passes up a REPLY to its parent (in the case of a root node, it passes up a REPLY to the originator). If the originator receives REPLY messages from all of its children and other segments, it propagates CONFIRM messages down to its children and the root nodes of other segments. When a node receives a CONFIRM, it checks its *flag_confirm* to see whether it has received a CONFIRM beforehand. If not, it sets its *flag_confirm* to 1 and propagates CONFIRMs to its children.

One thing we need to clarify is how to keep the information on the neighboring nodes of node i, Ii. Only in the START message for which node i acts as a non-boundary node, the information on i's neighbors in a given tree is contained. Therefore I_i is retrieved from the START message for which i acts as a non-boundary node (refer to SEG_INFO() in E-N-1). And from the START message for which node i works as a boundary node, the information on i's neighbors is not extracted. The reason is that in this case node i has only one neighbor which is its parent in the segment and i exchanges messages only with its parent in the segment.

Parameters

$LPID_i$: the LPID chosen at node i,
$LPID_i$ is a $geq0$ integer
initial value = -1
I_i : the set of neighboring nodes of i,
initial value = ϕ
$s(i)$: the subtree rooted at i in segment s
$flag\_reply_i$: a status flag of a node x,
if a $REPLY$ was from i, = 1,
if a $REPLY$ was not from i, = 0,
initial value = 0
$flag\_confirm$: a status flag of a node,
if a $CONFIRM$ was received, = 1,
if a $CONFIRM$ was not received, = 0,
initial value = 0

Types of Tree Setup Control Messages

$START(s, LPID_i)$
$REPLY(LPID_i, LPID_j)$
$CONFIRM(LPID_j)$

where, s is a segment
$LPID(i)$ is LPID of sender
$LPID(j)$ is LPID of receiver

PROCEDURE SEG_INFO (In: s, n, Out: I_n)
/* find neighboring nodes of n */

$I_n := \{x | x \in s^V$ and $e_{nx} \in s^E\}$;

PROCEDURE SUB_SEG (In: s, n, Out: $s(n)$)
/* find the subtree rooted at n within segment s */

$s(n)$:= the subtree rooted at n in s;

The STS Protocol

at NETWORK NODE i

E-N-1: RECEIPT of a $START(s, LPID_n)$ from n

```
    if (i = boundary node) { /* i.e., |s^V| = 1 */
L1 :  if (LPID_i = -1) { /* if an LPID wasn't chosen */
        LPID_i := an available LPID;
        flag_reply_n := 1;
        send a REPLY(LPID_i, LPID_n) to n;
      }
L2 :  else { /* if an LPID was chosen */
        flag_reply_n := 1;
        if (flag_reply_j = 1, ∀j ∈ I_i except only one k,
            where k ∈ I_i and k ≠ j) /* k is parent of i */
          send a REPLY(LPID_i, LPID_k) to k;
      }
    }
    else { /* if i is not a boundary node */
      if (source of START(s, LPID_n) is Or)
        I_i := {Or}; /* when i is the root of s */
      else
        I_i := {n};
      SEG_INFO(s, i, I_i);
L3 :  if (LPID_i = -1) {
        LPID_i := an available LPID;
        for (j ∈ I_i and j ≠ n) {
          SUB_SEG(s, j, s(j));
          send a START(s(j), LPID_i) to j;
        }
      }
L4 :  else {
        for (j ∈ I_i and j ≠ n)
          if (flag_reply_j = 0) {
            SUB_SEG(s, j, s(j));
            send a START(s(j), LPID_i) to j;
          }
        if (flag_reply_j = 1, ∀j ∈ I_i except only one k,
            where k ∈ I_i and k ≠ j) /* k is parent of i */
          send a REPLY(LPID_i, LPID_k) to k;
      }
    }
```

E-N-2: RECEIPT of a $REPLY(LPID_n, LPID_i)$ from n

```
if (flag_reply_n = 0) {
  flag_reply_n := 1;
  if (flag_reply_j = 1, ∀j ∈ I_i except only one k,
      where k ∈ I_i and k ≠ j) /* k is parent of i */
    send a REPLY(LPID_i, LPID_k) to k;
}
```

E-N-3: RECEIPT of a $CONFIRM(LPID_i)$ from n

```
if (flag_confirm = 0) {
  flag_confirm := 1;
  for (j ∈ I_i and j ≠ n)
    send a CONFIRM(LPID_j) to j;
}
```

at END NODE i

E-E-1: RECEIPT of an $ACT(T)$ from the higher layer;
at the originator

```
Or := i;
LPID_i := an available LPID;
SEGMENT(i, T, S, R, seg_sz);
SEG_INFO(s_1, i, I_i); /* s_1 is the segment rooted at Or */
for (j ∈ I_i) {
  SUB_SEG(s_1, j, s_1(j));
  send a START(s_1(j), LPID_i) to j;
}
for (s_j ∈ S, for j = 2 to seg_sz)
  send a START(s_j, LPID_i) to r_j;
```

E-E-2: RECEIPT of a $START(s, LPID_n)$ from n

```
LPID_i := an available LPID;
send a REPLY(LPID_i, LPID_n) to n;
```

E-E-3: RECEIPT of a $REPLY(LPID_n, LPID_i)$ from n;
at the originator

```
if (flag_reply_n = 0) {
  flag_reply_n := 1;
  if (flag_reply_j = 1, ∀j ∈ I_i and j ∈ R)
    for (j ∈ I_i or j ∈ R)
      send a CONFIRM(LPID_j) to j;
}
```

E-E-4: RECEIPT of a $CONFIRM(LPID_i)$ from n

```
flag_confirm := 1;
```

Figure 3. The STS protocol

All the control messages between the originator and the root node of a segment are sent via a shortest path between them and we assume that each site has the knowledge of the shortest path to any node as in ARPANET [13]. START messages include the

originator and the unique tree identifier (which is unique only at the originator) in order to inform nodes of messages belonging to the same tree (in case of the SBO protocol, this tree identification information is not used since the LPID information is enough to identify a tree uniquely). However, a REPLY/CONFIRM message does not require this kind of information since it has already obtained its parent's/children's LPID(s). The tree identification information is kept at each site until a CONFIRM message is received. This is because a node may receive START messages twice and send back a REPLY for the first START (refer to Figure 4 (a)). In this case, if we erase the tree identification information after receiving the REPLY, the second START message may lose the connection to the first START. This tree identification information could be one of the overheads of using the STS protocol.

Figure 4 shows two possible scenarios of passing control messages between two adjacent segments. Here, "(4) $S(LPID(x))$" implies a START message from node x with x's LPID which is the fourth message generated. Node i is a boundary node in segment B and a non-boundary node in segment A. And node j is a boundary node in segment A and a non-boundary node in segment B. Only the possible cases from the aspect of node i are illustrated as the operations at nodes i and j are symmetrical. Figure 4 (a) is the case when i receives a START message from B prior to a START from A, in which i is treated as a boundary node of B. In this case, i sends back a REPLY to node j after selecting the i's LPID and making as if node i received a REPLY from j by setting $flag\_reply_j$ to 1. A START message from node x does not cause the LPID selection since the tree setup operation at node i has been carried out by the prior START message from node j (in this case node x acts as a non-boundary node of segment A). Instead, node x sends a START to each one of its children which hasn't replied to x, and passes up a REPLY to its parent if all of its children has replied (refer to *L4*). Figure 4 (b) is the case when i receives a START message from A earlier than a START from B, in which i is considered as a non-boundary (middle) node of A (refer to *L3*). After node i transmits a START to j with its LPID, i receives a START from j, which is ignored and considered as a REPLY from j since the tree setup operation at i has been already performed by the earlier START message from x (refer to *L2*). When node j receives a START from i, j also assumes it as a REPLY from i because j has already sent out a START to i.

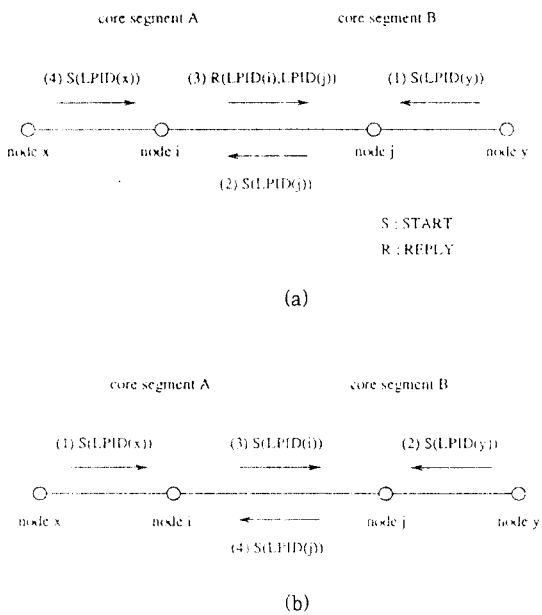

Figure 4. Scenarios of passing control messages between two adjacent segments

During the course of tree setup or the group conversation, if a node/link fails, a multicast tree can be partitioned. The SBO protocol assures reliability by allowing members in a tree partition to continue their conversation and the tree setup process to be completed within the partition. By means of CNCL messages, this functionality is achieved. The reliability issue is not covered in this paper and left as a future work. In this paper, we assume no link/node failures and no errors in control messages.

For the verification of the correctness of the proposed STS protocol, we need to show that each node in a given tree selects its own LPID only once and knows all of its neighbors' LPIDs in a finite time.

Lemma 1 *Each node in a tree selects its own LPID only once.*

Proof: Since Or receives only one ACT message and no START messages, the selection of the LPID for Or is performed only once at E-E-1. Since each end node receives only one START message and no ACT message, it selects its LPID only once at E-E-2. In case of a network node, it receives one or more START messages according to its node type. If a network node is a boundary node, it may receive one or two START messages (refer to Figure 4). But before it selects its LPID, it first checks its LPID and, only when LPID = -1, it is assigned with an available LPID at E-N-1. For a non-boundary network node, since it receives only one START message, only one available LPID is assigned to it at E-N-1. Therefore, each node in a tree selects its own LPID only once. □

Lemma 2 *Each node in a tree receives the LPIDs from all of its neighbors in a finite time.*

Proof: In a segment, each node at the bottom level can be either an end or a boundary node. In case of an end node which is not *Or*, its only neighbor is its parent and it receives a START from its parent at E-E-2. In case of an end node which is *Or*, its neighbors are its children and it receives REPLYs from them at E-E-3. In case of a boundary node, its neighbor is only its parent whose LPID is obtained from a START (in Figure 4 (b), node *j* in segment *A* receives a START from node *i*; at E-N-1) or a REPLY (in Figure 4 (a), node *j* in segment *A* receives a REPLY from node *i*; at E-N-2).

In each segment, those nodes not at the bottom are non-boundary nodes. Each one of these can belong (i) to only one segment or (ii) to more than one segment. In case (i), the node is not a boundary node in any segments and its neighbors are its parent and children who belong to the same segment. Its parent's LPID is obtained via a START and its children's LPIDs via REPLYs. In case (ii), this node is treated as a boundary node in another segment. Its neighbors are its parent and children which can be either end or boundary nodes. Its parent's LPID is obtained from a START, and its children's LPIDs from STARTs at E-N-1 (virtual REPLYs; in Figure 4 (a) node *i* in segment *A* receives a START from node *j*) or REPLYs at E-N-2 (in Figure 4 (a) node *j* in segment *B* receives a REPLY from node *i*).

Therefore each node in a tree receives the LPIDs from all of its neighbors. Since it is assumed that there are no failures during the tree setup, this operation can be accomplished in a finite time. □

Theorem 1 *A multicast message on a tree is transmitted only once to every end node of the tree.*

Proof: By Lemmas 1 and 2, each node

which receives a message recognizes the tree to which the message should be forwarded by looking at the LPID information within the message. Once it identifies the tree, it forwards the message to its neighboring nodes with the corresponding LPIDs, except that one which has transmitted the message. Due to the properties of the tree structure, i.e., there is a unique path from the root to a node in a tree and a tree is a connected graph, each node in the tree receives a message only once and a message is multicast to all the end nodes in the tree if there are no failures involved. □

V. Performance Analysis

In order to analyze the performance of our STS protocol, we come up with the following performance factors:

τ : the time to establish a tree completely

μ : the number of tree setup control messages generated

σ : the total amount of node information carried in START messages

Before we get into the performance analysis, the following considerations are made on the relationship between the number of segments and the number of boundary nodes. In order to consider boundary nodes, we introduce the term "connection" which means a pair of nodes x and y that belongs to two segments at the same time and $e_{xy} \in E_T$. Each node involved in a connection belongs to one segment as a boundary node and to the other segment as a non-boundary node. And if there exists at least one edge $e \in E_T$ between two segments, we say those two segments are adjacent. When $\cup_i s_i^V$ and $\cup_i s_i^E$ yields a connected graph, we say segments s_i's are *merged*.

Lemma 3 *There exists only one connection between two adjacent segments.*

Proof: If there exists more than one connection between two adjacent segments, a loop is formed. Because segments are made from a tree, merging two adjacent segments must produce a tree, i.e., there should not be any loops. Therefore, there exists only one connection between two adjacent segments. □

Lemma 4 *A segment s_i does not have a connection to another segment s_j if there exists a path from si to s_j via other segments.*

Proof: If there already exists a path from s_i to s_j via other segments, having a connection between s_i and s_j introduces a loop. Therefore s_i does not have a connection to segment s_j. □

Theorem 2 *There exist (n -1) connections if n segments are merged.*

Proof: By Lemmas 3 and 4, the only possible structure of merging n segments is a tree where a segment corresponds to a node and a connection to an edge. Therefore, if there are n segments, $(n - 1)$ connections are needed to merge them. □

The tree setup time

$\tau_{SBO} = 3 \cdot D + \nu_D \cdot \alpha$

$\tau_{STS} = 3 \cdot \Delta + \nu_\Delta \cdot \alpha + \epsilon$

where ν_D: the number of nodes on the longest path from *Or* in a tree

ν_Δ : the number of nodes on the longest path from the roots of segments in all segments

α : the tree setup processing delay incurred at a node

ϵ : 1, if there exist two adjacent segments whose depths are Δ 0, otherwise

Since $D \geq \Delta$ and $\nu_D \geq \nu_\Delta$, $\tau_{STS} \leq \tau_{SBO}$. When a given tree is a shortest path tree rooted at *Or*, $\tau_{STS} = \tau_{SBO}$. "3" comes from the fact that both protocols consist of three phases. τ_{STS} introduces overhead delay one if there are

two adjacent segments having depth Δ, because the corresponding boundary nodes transmit STARTs and wait for REPLYs (in fact, virtual REPLYs) (see Figure 4 (b)).

The number of tree setup control messages

$\mu_{SBO} = 3\ |E_T|$

$\mu_{STS} = 3\ (|E_T|+|S|-1)+\gamma \cdot (|S|-1)$

where γ: the fraction of connections in which the depths of the corresponding boundary nodes are the same

As shown in Figure 4, two control messages (excluding CONFIRM messages) are passed on each link in the tree. "$|S|-1$" in calculating μ_{STS} comes from the fact that those tree setup control messages between Or and its only child (i.e., the network node to which Or is attached) are already counted in the factor "E_T". Since $|S|$ 1, μ_{STS} μ_{SBO}. When $|S| = 1$ (i.e., a given tree is a shortest path tree rooted at Or), $\mu_{STS} = \mu_{SBO}$. As |S| (i.e., the number of segments) increases, the STS protocol works worse in terms of μ. Therefore the number of segments needs to be minimized. By Theorem 2, "$|S|-1$" implies the number of connections. Therefore "$\gamma \cdot (|S|-1)$" gives the number of extra CONFIRMs generated on links involved in connections because, only when two nodes in a connection have the same delay from Or, they will transmit CONFIRMs to each other, one of which is extra.

The amount of node information carried

$\sigma_{SBO} = \eta \cdot (\sum_{n_i \in V_T} \delta_T(Or, n_i))$, $\forall n_i \in V_T$

$\sigma_{STS} = \eta \cdot (\sum_{s_i \in S} \sum_{n_j \in s_i} \delta_{G(Or, r_i)} + \delta_T(r_i, n_j) - (1-\gamma)(|S|-1))$, $\forall s_i \in S$, $n_j \in s_i^V$

where η : the amount of the node information

Both in the STS and the SBO protocols, the information on a node n is carried along the path from Or to n. Since the path from Or to n in the STS protocol is shorter than that in the SBO protocol, the amount of tree information carried by START messages in the STS protocol is less than that in the SBO protocol if we do not take account into boundary nodes.

By Theorem 2, the total number of nodes whose information is carried twice by START messages is "$2 \cdot (|S|-1)$" (see Figure 4 (b)). Even for the boundary node which receives only one START message, its node information needs to be delivered by two START messages even though one of the START messages is delivered only upto its adjacent node, (i.e., the node information on node j in Figure 4 (a) is carried by $S(LPID(x))$ and $S(LPID(y))$, even though $S(LPID(x))$ is delivered only upto i).

For example, in Figure 2, the performance factors of the STS and the SBO protocol are obtained as follows:

$\tau_{SBO} = 3 \cdot 6+7 \cdot \alpha = 18$, $\tau_{STS} = 3 \cdot 4+5 \cdot \alpha = 12$ (assuming $\alpha = 0$)

$\mu_{SBO} = 3\ 11 = 33$, $\mu_{STS} = 3 \cdot 12+1 = 37$

$\sigma_{SBO} = 41$, $\sigma_{STS} = 41$ (assuming $\eta = 1$)

Ⅵ. Conclusions

In this paper, we proposed a faster tree setup protocol using segmentation of a multicast tree based on the LPID scheme. The tree setup time is bounded to three times Δ by chopping up a tree into several disjoint core segments whose maximum delay from the originator to anyone in the segment is less than or equal to Δ (a segment is a core segment with boundary nodes). When a minimal Steiner tree is established, we can get most benefit from the STS protocol. By introducing the concept of segmentation, problems like more control messages and the overhead for segmentation are also introduced. To allievi-

ate these problems, we need to minimize the number of segments and leave this problem to be solved. Also the support of reliability in the STS protocol needs to be studied further.

As pointed out earlier in Section I, the VPI/VCI concept of the ATM is similar to the LPID. Therefore the proposed STS protocol can be used in ATM networks. In ATM networks, a multipoint connection can be either a multipoint VP (Virtual Path) or a multipoint VC (Virtual Channel) (a multipoint VC can be established based on either preexisting point-to-point VPs or a preexisting multipoint VP) [1]. In case of a multipoint VP setup, since each node selects its own VPI independently of other nodes, the STS protocol can be applied without any modifications. On the other hand, setting up a multipoint VC based on preexisting point-to-point VPs requires the STS protocol to be modified such that preestablished point-to-point VPs are treated as links.

References

1. Ahn, S., "Multicast Tree Approaches for Group-based Multimedia Applications over ATM Networks", *Ph.D. dissertation, Computer Science Department*, University of Minnesota, Minneapolis, MN, Dec. 1993.
2. Amin-Salehi, B., Flinchbaugh, G.D., and Pate, L.R., "Implications of New Network Services on BISDN Capabilities", *IEEE Globecom*, pp.1038-1045, 1990.
3. Boudec, J., "The Asynchronous Transfer Mode: a tutorial", *Computer Networks and ISDN Systems*, Vol. 24, pp.279-309, 1992.
4. Chow, C.H., "On Multicast Path Finding Algorithms", *IEEE Infocom*, Vol. 3, pp.11B.1.1-10, 1991.
5. Cidon, I., Gopal, I., and Gu rin, R., "Bandwidth Management and CongestionControl in plaNET", *IEEE Communications Magazine*, pp.54-64, Oct. 1991.
6. Day, A., "International Standardization of BISDN", *IEEE LTS*, pp.13-20, Aug. 1991.
7. Ferguson, M.J., and Mason, L.G., "Network Design for a Large Class of Teleconferencing Systems", *IEEE Transactions on Communications*, Vol. COM-32, No. 7, pp.789-796, July. 1984.
8. Flinchbaugh, G.D., Martinez, P.L., and Rouse, D.S., "Network Capabilities in Support of Multimedia Applications", *IEEE Globecom*, pp. 308.7.1-5, 1990.
9. Frank, A.J., Wittie, L.D., and Bernstein, A.J., "Multicast Communication on Network Computers", *IEEE Software*, Vol. 2, No. 3, pp.46-61, May. 1985.
10. Kawarasaki, M., and Jabbari, B., "B-ISDN Architecture and Protocol", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, pp.1405-1415, Dec. 1991.
11. Markowsky, G., and Moss, F.H., "An Evaluation of Local Path ID Swapping in Computer Networks", *IEEE Trans. on Communications*, Vol. COM-29, No. 3, pp.329-336, March. 1981.
12. McDysan, D.E. and Spohn, D.L., "ATM: Theory and Application", *McGraw-Hill*, Inc., 1995.
13. McQuillan, J.M., Richer, I. and Rosen E.C., "The New Routing Algorithm for the ARPANET", *IEEE Transactions on Communications*, Vol. COM-28, No. 5, pp.711-719, May. 1980.
14. Sabri, S., and Prasada, B., "Video Conferending Systems", *Proc. of the IEEE*, Vol. 73, No. 4, pp.671-688, April. 1985.
15. Segall, A., and Jaffe, J.M., "Route Setup with Local Identifiers", *IEEE Trans. on Communications*, Vol. COM-34, No. 1, pp.45-53, Jan. 1986.

16. Segall, A., Barzilai, T.P., and Ofek, Y., "Reliable Multiuser Tree Setup with Local Identifiers", *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, pp.1427-1439, Dec. 1991.

安 相 炫(Sanghyun Ahn) 정회원

1986년 : 서울대학교 컴퓨터공학과 학사
1988년 : 서울대학교 컴퓨터공학과 석사
1993년 : University of Minnesota 전산학 박사
1988년~1989년 : 데이콤 연구원
1994년~현재 : 세종대학교 전산과학과 전임강사