Grenville Armitage
Bellcore
January 22nd, 1997

A Distributed MARS Protocol
<draft-armitage-ion-distmars-spec-00.txt>

Status of this Memo

Abstract

The Server Cache Synchronisation Protocol (SCSP) has been proposed as a general mechanism for synchronising the databases of IP/ATM Servers, such as those used by NHRP and MARS. This document specifies an application of SCSP to allow multiple MARS entities to provide fault tolerance to MARS Clusters.

1. Introduction.

[Editors note: this probably has holes you could drive a bus through. if you drive buses and have an interest in this topic, offers of co-authorship will be well received.]

SCSP [1] being developed within the Internetworking over NBMA (ION) working group as a general solution for synchronizing distributed databases such as distributed Next Hop Servers [2] and MARSs [3]. Analysis of the possible distributed MARS scenarios is available in the form of an Internet Draft submitted to the ION working group [4]. It is assumed that the reader understands the issues raised in [4] regarding the Fault Tolerant model, and understands the services provided by SCSP as described in [1]. {Whether the author does is another matter entirely!}

In the current MARS model a Cluster consists of a number of MARS Clients (IP/ATM interfaces in routers and/or hosts) utilizing the services of a single MARS. This MARS is responsible for tracking the IP group membership information across all Cluster members, and providing on-demand associations between IP multicast group identifiers (addresses) and specific sets of one or more ATM endpoint addresses (for building ATM level multipoint forwarding paths). This MARS is also responsible for allocating Cluster Member IDs (CMIs) to Cluster members (inserted into outgoing data packets, to allow reflected packet detection when Multicast Servers are placed in the data path).

In [4] two different methods of using multiple MARS entities in a single cluster are defined - the Fault Tolerant model and the Load Sharing model. This document specifies the use of SCSP to provide the Fault Tolerant model. [Actually, at this point it doesn't really do either.] This model contains the following elements:

Active MARS
: The single MARS serving the Cluster. It allocates CMIs and tracks group membership changes by itself. It is the sole entity that constructs replies to MARS_REQUESTs.

Backup MARS
: An additional MARS that tracks the information being generated by the Active MARS. Cluster members may re-register with a Backup MARS if the Active MARS fails, and they'll assume the Backup has sufficient up to date knowledge of the Cluster's state to take the role of Active MARS.

Living Group
: The set of Active MARS and current Backup MARS entities. When a MARS entity dies it falls out of the Living Group. When it restarts, it rejoins the Living Group. Election of the Active MARS takes place amongst the members of the Living Group.

MARS Group

The total set of MARS entities configured to be part of the distributed MARS. This is the combination of the Living Group and 'dead' MARS entities that may be currently dying, dead, or restarting. The list is constructed in the following order {Active MARS, Backup MARS, ... Backup MARS, dead MARS,.... dead MARS}. If there are no 'dead' MARS entities, the MARS Group and Living Group are identical.

SCSP assumes that the state held by a given server is contained in a local object known as a cache. Keeping this cached state synchronized among each server in a Server Group (SG) is what SCSP does, with the use of alignment and update messages. Client State Update (CSU) messages contain actual cache update data within individual Client State Advertisement (CSA) records. These may be transmitted by a server when a local cache state change occurs, or when solicited by a neighboring server using a Client State Update Solicitation (CSUS) message.

SCSP also defines an alignment phase where members of a Server Group discover if their caches are out of alignment be comparing Client State Advertisement Summary (CSAS) records. CSAS records are carried in Client Alignment (CA) messages between neighboring servers.

The rest of this document is structured as follows:

[TBD]

2. MARS Sub-caches

For the purpose of applying SCSP to the distributed MARS scenario, we subdivide the SCSP "cache" into a number of sub-caches, each of which are kept independently synchronized. Any given server in an SG builds up its notion of a distributed sub-cache's contents from the sum of sub-cache information flooded by other servers in the SG.

Each MARS sub-cache has a separate sequence number space, allowing sub-caches to be updated/flooded independently.

The sub-caches are derived from the following components of the overall MARS state for a given Cluster:

Cluster membership list.

Cluster Member IDs.

MCS (Multicast Server) membership list.

Absolute maximum and minimum group addresses for protocol being supported.

Member map (hostmap) for each Layer 3 group.

MCS (Multicast Server) Servermap for each Layer 3 group.

Block-join map.

Redirect_map.

The Cluster membership list is the most fundamental object for a MARS. It contains the ATM addresses of every cluster member, and explicitly maps Cluster Member ATM addresses to Cluster Member IDs. Both of these pieces of information will be combined into a single CMI map sub-cache.

The MCS membership list is essential to enable construction of a backup ServerControlVC by any one of the backup MARSs.

Each multicast group is represented by a membership map (hostmap) sub-cache. Since a given multicast group may also have MCSs registered to support it there is also a matching Servermap. Hostmaps and Servermaps are treated as separate sub-caches. To simplify and shorten the CSA Records, members of these maps are identified by their Cluster Member IDs rather than enumerating their actual ATM addresses. The key into a hostmap or servermap is the group's multicast address.

Since hostmaps for a given group may be quite large, and most MARS_JOIN/LEAVE events simply result in an incremental change to the host map, two different types of CSA record will be defined. One will represent the sub-cache in its entirety (for use when aligning servers), and the second will have semantics to match the JOIN/LEAVE event (allowing an incremental addition to, or deletion from, the sub-cache associated with the specific multicast group). The same will apply to the CSA records for Servermaps. SCSP has a mechanism for segmenting large CSA Records across multiple CSU messages.

The block-join map represents all currently valid block MARS_JOINs registered with the MARS. This allows the preceding, group-specific hostmaps to be simplified. (The CSA Records representing the hostmap for a given group only lists nodes that have issued a specific single-group MARS_JOIN for that group.) Internally, the MARS builds whatever database structure is required to ensure that replies to MARS_REQUESTs, and general hole-punching activities, take the block-

join map's contents into account.

The Redirect_map is the list of MARS entities a given server is currently sending in its MARS_REDIRECT_MAP messages. The MARS Clients consider this list to be the available Backup MARS entities. Its use is TBD.

3. Client State Advertisement Summary (CSAS) records.

Client State Advertisement Summary (CSAS) records are carried within SCSP Cache Alignment (CA) messages. They are used to inform one server of another server's {sub}cache state without sending the contents of the cache itself.

CSAS records have an 4 byte fixed header defined by SCSP, followed by protocol specific fields (the Server Group ID - SGID - is contained in the header of the CA messages that carry CSAS records).

For MARS use we add a 16 bit CSAS record (sub-cache) type field. The first 6 bytes of each CSAS record are thus:

```
csas$sequence  32 bits  CSA Sequence Number.
csas$type      16 bits  CSAS record sub-cache type.
```

The CSA Sequence number indicates how recently the specified sub-cache (csas$type) has been modified. This is used to determine whether a re-alignment is required. Each CSAS also contains an Originator ID field, which identfies which server in the SG is the "owner" (originator) of the sub-cache information to which the CSAS refers. For a MARS server group, the Orignator ID is the NBMA address of the originating MARS.

The remaining bytes of the CSAS record are determined by csas$type.

The MARS CSAS record types are:

```
CSAS_CMI_MAP          1
CSAS_MCS_LIST         2
CSAS_HOST_MAP         3
CSAS_MCS_MAP          4
CSAS_BLOCK_JOINS      5
CSAS_REDIRECT_MAP     6
```

The specific formats of each CSAS record are described in the following sub-sections.

3.1 CSAS_CMI_MAP.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 1 (CSAS_CMI_MAP)
csas$orig_len   8 bits   Length of csas$origin field.
csas$unused     8 bits   unused.
csas$origin     x octets Originator ID.
```

For this CSAS, the sequence number is incremented every time a new cluster member registers, or an old one is considered to have died or deregistered.

3.2 CSAS_MCS_LIST.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 2 (CSAS_MCS_LIST)
csas$orig_len   8 bits   Length of csas$origin field.
csas$unused     8 bits   unused.
csas$origin     x octets Originator ID.
```

For this CSAS, the sequence number is incremented every time a new MCS registers, or an old one is considered to have died or deregistered.

3.3 CSAS_HOST_MAP.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 3 (CSAS_HOST_MAP)
csas$orig_len   8 bits   Length of csas$origin field.
csas$group_len  8 bits   Length of group address.
csas$origin     x octets Originator ID.
csas$group      y octets Hostmap entry's group address.
```

For this CSAS, the sequence number is incremented whenever a cluster member joins or leaves the group specified by csas$group.

3.4 CSAS_MCS_MAP.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 4 (CSAS_MCS_MAP)
```

```
csas$orig_len   8 bits   Length of csas$origin field.
csas$group_len  8 bits   Length of group address.
csas$origin     x octets Originator ID.
csas$group      y octets Servermap entry's group address.
```

For this CSAS, the sequence number is incremented whenever an MCS joins or leaves the group specified by csas$group.

3.5 CSAS_BLOCK_JOINS.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 5 (CSAS_BLOCK_JOINS)
csas$orig_len   8 bits   Length of csas$origin field.
csas$unused     8 bits   unused.
csas$origin     x octets Originator ID.
```

For this CSAS, the sequence number is incremented whenever a block MARS_JOIN, or matching block MARS_LEAVE, occurs.

3.6 CSAS_REDIRECT_MAP.

The complete CSAS Record looks like:

```
csas$sequence  32 bits   CSA Sequence Number.
csas$type      16 bits   Set to 6 (CSAS_REDIRECT_MAP)
csas$orig_len   8 bits   Length of csas$origin field.
csas$unused     8 bits   unused.
csas$origin     x octets Originator ID.
```

For this CSAS, the sequence number is incremented whenever the local server modifies the list of MARS entities in its MARS_REDIRECT_MAP list.

4. Client State Advertisement (CSA) Records.

CSA records have an 12 byte fixed header defined by SCSP, followed by protocol specific fields. For MARS use we add a 16 bit CSA record (sub-cache) type field. The first 14 bytes of each CSAS record are thus:

```
csa$fragment  16 bits  F/Fragment Number.
csa$ttl       16 bits  TTL.
csa$sequence  32 bits  CSA Sequence Number.
csa$sgid      32 bits  Server Group ID.
csa$type      16 bits  CSA Record sub-cache type.
```

The CSA Sequence number indicates how recently the specified sub-cache (csa$type) has been modified. This is used to determine whether a re-alignment is required. The Server Group ID identifies an instance of a Server Group. Since Server Groups will exist on a per-protocol basis (IPv4, IPv6, etc) the csa$sgid field implicitly identifies the formats of any 'group' address fields within the CSA Records.

Each CSA also contains an Originator ID field, which identfies which server in the SG is the "owner" (originator) of the sub-cache information to which the CSA refers. In the case of MARS server groups, the originator is identified by its ATM address (cf. the NHRP case where the 'protocol address' is used). The format of the ATM address is irrelevant - the originator field is simply an uninterpreted octet string used for pattern matching.

The remaining bytes of the CSA record are determined by csa$type.

To match the CSAS records, the following set of CSA record types are defined:

```
CSA_CMI_MAP        1
CSA_MCS_LIST       2
CSA_HOST_MAP       3
CSA_MCS_MAP        4
CSA_BLOCK_JOINS    5
CSA_REDIRECT_MAP   6
```

In addition, to allow indication of incremental updates to some of the sub-caches, matching

```
CSA_CMI_MAP_JOIN   128
CSA_CMI_MAP_LEAVE  129
CSA_MCS_LIST_JOIN  130
CSA_MCS_LIST_LEAVE 131
CSA_HOST_MAP_JOIN  132
CSA_HOST_MAP_LEAVE 133
CSA_MCS_MAP_JOIN   134
CSA_MCS_MAP_LEAVE  135
```

(csa$type values in the range 1 to 127 correspond to entire sub-caches, whilst the range 128 to 512 are allocated to incremental sub-cache updates.)

The amount of information carried by a specific CSA_HOST_MAP or CSA_CMI_MAP may exceed the size of a link layer PDU. SCSP allows a large CSA Record to be fragmented across a number of CSU Request messages.

4.1 CSA_CMI_MAP.

This CSA Record carries the entire membership of the current cluster, along with the Cluster Member IDs (CMIs) assigned by the MARS they registered with. These CMIs are then used as a short-form representation of the actual cluster members to compress the size of subsequent CSA_HOST_MAP messages.

```
   csa$fragment   16 bits  F/Fragment Number.
   csa$ttl        16 bits   TTL
   csa$sequence   32 bits   CSA Sequence Number.
   csa$sgid       32 bits   Server Group ID.
   csa$type       16 bits   Set to 1 (CSA_CMI_MAP).
   csa$orig_len    8 bits   Length of csa$origin.
   csa$unused      8 bits   unused.
   csa$cnum       16 bits   Number of entries in this CSA (N).
   csa$thtl        8 bits   Type and length of ATM addresses.
   csa$tstl        8 bits   Type and length of ATM sub-addresses.
   csa$origin      x octets Originator's NBMA address.
   csa$atmaddr.1   q octets ATM address of member 1.
   csa$subaddr.1   r octets ATM sub-address of member 1.
   csa$cmi.1      16 bits   Cluster Member ID for entry 1.
                 [..etc..]
   csa$atmaddr.N   q octets ATM address of member N.
   csa$subaddr.N   r octets ATM sub-address of member N.
   csa$cmi.N      16 bits   Cluster Member ID for entry N.
```

4.2 CSA_MCS_LIST.

This CSA Record carries the entire list of currently registered Multicast Servers (MCSs). Each MCS is also assigned an internal ID by the MARS they registered with - this is used to compress the size of subsequent CSA_MCS_MAP messages.

```
   csa$fragment   16 bits   F/Fragment Number.
   csa$ttl        16 bits   TTL
   csa$sequence   32 bits   CSA Sequence Number.
   csa$sgid       32 bits   Server Group ID.
   csa$type       16 bits   Set to 2 (CSA_MCS_LIST).
   csa$orig_len    8 bits   Length of csa$origin.
   csa$unused      8 bits   unused.
   csa$cnum       16 bits   Number of entries in this CSA (N).
   csa$thtl        8 bits   Type and length of ATM addresses.
   csa$tstl        8 bits   Type and length of ATM sub-addresses.
   csa$origin      x octets Originator's NBMA address.
   csa$atmaddr.1   q octets ATM address of MCS 1.
   csa$subaddr.1   r octets ATM sub-address of MCS 1.
```

```
csa$cmi.1      16 bits   Internal MCS ID for entry 1.
               [..etc..]
csa$atmaddr.N   q octets ATM address of member N.
csa$subaddr.N   r octets ATM sub-address of member N.
csa$cmi.N      16 bits   Internal MCS ID for entry N.
```

4.3 CSA_HOST_MAP

This CSA Record carries the list of cluster members who have joined a specified group using a single-group MARS_JOIN operation. The Cluster Member IDs are used to represent each group member with each CSA Record fragment. A recipient MARS uses this CSA in conjunction with the current Cluster membership list to derive the actual ATM addresses of group members.

```
csa$fragment   16 bits   F/Fragment Number.
csa$ttl        16 bits   TTL
csa$sequence   32 bits   CSA Sequence Number.
csa$sgid       32 bits   Server Group ID.
csa$type       16 bits   Set to 3 (CSA_HOST_MAP).
csa$orig_len    8 bits   Length of csa$origin.
csa$group_len   8 bits   Length of csa$group.
csa$cnum       16 bits   Number of entries in this fragment (N).
csa$origin      x octets Originator's NBMA address.
csa$group       y octets Multicast group's protocol address.
csa$cmi.1      16 bits   Cluster Member ID for entry 1.
csa$cmi.2      16 bits   Cluster Member ID for entry 2.
               [..etc..]
csa$cmi.N      16 bits   Cluster Member ID for entry N.
```

4.4 CSA_MCS_MAP

This CSA Record carries the list of MCSs who have joined to support a specified group. The internal MCS IDs from prior CSA_MCS_LIST CSA Records are used to represent each MCS. A recipient MARS uses this CSA in conjunction with the current MCS membership list to derive the actual ATM addresses of group members.

```
csa$fragment  16 bits   F/Fragment Number.
csa$ttl       16 bits   TTL
csa$sequence  32 bits   CSA Sequence Number.
csa$sgid      32 bits   Server Group ID.
csa$type      16 bits   Set to 4 (CSA_MCS_MAP).
csa$orig_len   8 bits   Length of csa$origin.
csa$group_len  8 bits   Length of csa$group.
```

```
csa$cnum      16 bits   Number of entries in this fragment (N).
csa$origin     x octets Originator's NBMA address.
csa$group      y octets Multicast group's protocol address.
csa$cmi.1     16 bits   Internal MCS ID for entry 1.
csa$cmi.2     16 bits   Internal MCS ID for entry 2.
              [..etc..]
csa$cmi.N     16 bits   Internal MCS ID for entry N.
```

4.5 CSA_BLOCK_JOINS

This CSA Record carries the list of Cluster Members who have joined blocks of the layer 3 group address space. The Cluster Member IDs from prior CSA_CMI_MAP CSA Records are used to represent each cluster member and associate it with a specific <min,max> pair.

```
csa$fragment  16 bits   F/Fragment Number.
csa$ttl       16 bits   TTL
csa$sequence  32 bits   CSA Sequence Number.
csa$sgid      32 bits   Server Group ID.
csa$type      16 bits   Set to 5 (CSA_BLOCK_JOINS).
csa$orig_len   8 bits   Length of csa$origin.
csa$group_len  8 bits   Lengths of csa$min and csa$max fields.
csa$cnum      16 bits   Number of entries in this fragment (N).
csa$origin     x octets Originator's NBMA address.
csa$min.1      y octets <min> group address of block 1.
csa$max.1      y octets <max> group address of block 1.
csa$cmi.1     16 bits   Cluster Member ID for block 1.
              [..etc..]
csa$min.N      y octets <min> group address of block N.
csa$max.N      y octets <max> group address of block N.
csa$cmi.N     16 bits   Cluster Member ID for block N.
```

4.6 CSA_REDIRECT_MAP

This CSA Record carries the list the source server is using to generate MARS_REDIRECT_MAP messages.

```
csa$fragment  16 bits   F/Fragment Number.
csa$ttl       16 bits   TTL
csa$sequence  32 bits   CSA Sequence Number.
csa$sgid      32 bits   Server Group ID.
csa$type      16 bits   Set to 6 (CSA_REDIRECT_MAP).
csa$orig_len   8 bits   Length of csa$origin.
csa$unused     8 bits   unused.
csa$cnum      16 bits   Number of entries in this fragment (N).
```

```
csa$thtl       8 bits   Type and length of ATM addresses.
csa$tstl       8 bits   Type and length of ATM sub-addresses.
csa$origin     x octets Originator's NBMA address.
csa$atmaddr.1  q octets ATM address of member 1.
csa$subaddr.1  r octets ATM sub-address of member 1.
               [..etc..]
csa$atmaddr.N  q octets ATM address of member N.
csa$subaddr.N  r octets ATM sub-address of member N.
```

4.7 Incremental update CSA Records.

The incremental update CSA Records types use the same format as alignment CSA Records, except that only a single entry (of whatever information) is passed.

Two examples:

CSA_CMI_MAP_LEAVE is coded as a CSA_CMI_MAP but with csa$type = 129, csa$cnum = 1, and only a single ATM address and CMI pair provided.

CSA_HOST_MAP_JOIN is coded as a CSA_HOST_MAP but with csa$type = 132, csa$cnum = 1, and only a single CMI is provided (indicating the specific host that has now become a group member).

The CSA Sequence number space for incremental update CSAs is the same space used by alignment CSAs for the identfied sub-cache.

Alignment updates (containing a full version of an identified sub-cache) are always accepted by a server.

Incremental update CSAs are are accepted as updates to the local server's copy of the specified sub-cache, from the specified Originator ID, only if they arrive with a larger (newer) CSA Sequence number than the existing local entry.

Incremental update CSAs are discarded if they arrive with a smaller (older) CSA Sequence number than the local server already has for the specified sub-cache, from the specified Originator ID.

[The rest of this explanation is TBD.]

5. Use of CSA Records.

The most important sub-caches for a MARS to exchange are the CSA_CMI_MAP and CSA_MCS_LIST. Without alignment of these sub-caches, members of the Server Group cannot interpret the other CSA Record types, which identify nodes using ID values supplied in the CSA_CMI_MAP and CSA_MCS_LIST records.

MARS_JOIN/LEAVE events are propagated by issuing CSA_HOST_MAP_JOIN and CSA_HOST_MAP_LEAVE CSA Records in CSU messages.

There are no CSAS Record types equivalent to the incremental sub-cache update CSA Record types. The semantics of the CSAS Record in the CA message is to indicate the state of an entire sub-cache. It would make no sense to try and discover (or convey) an 'incremental state' of a sub-cache.

As a consequence, incremental sub-cache update CSA Record types SHALL only be sent in un-solicited CSU Request messages. Client State Update Solicit (CSUS) messages SHALL only trigger the delivery of CSA Records containing entire sub-caches as atomic units.

Security Consideration

Security consideration are not addressed in this document.

Acknowledgments

To Liptons, for making the tea that keeps me going.

Author's Address

Grenville Armitage
Bellcore, 445 South Street
Morristown, NJ, 07960
USA

Email: gja@bellcore.com
Ph. +1 201 829 2635

References

[1] J. Luciani, G. Armitage, J. Halpern, "Server Cache Synchronization Protocol (SCSP) - NBMA", INTERNET DRAFT, draft-ietf-

ion-scsp-00.txt, November 1996.

[2] J. Luciani, et al, "NBMA Next Hop Resolution Protocol (NHRP)", INTERNET DRAFT, draft-ietf-rolc-nhrp-10.txt, October 1996.

[3] G. Armitage, "Support for Multicast over UNI 3.0/3.1 based ATM Networks.", Bellcore, RFC 2022, November 1996.

[4] G. Armitage, "Redundant MARS architectures and SCSP.", INTERNET DRAFT, draft-armitage-ion-mars-scsp-02.txt, November 1996.