Pilot ARM Session Recovery Design.

Dmitri Kondratiev

dkondr@bigfoot.com

Table of Contents

This document describes pilot design of ARM Session Recovery in Active Network (AN) running Active Reliable Multicast family of protocols (ARM). The design is 'pilot' as it provides limited, partial solution to ARM session recovery. Design is limited because of several important requirements and interdependencies are remaining unclear or not fully specified at the time of writing this document.

Problem statement

Initial setting is a group of active nodes (GSNs) participating in data transfer session with one designated active node as primary data source. This primary data source called GSN1 in this paper receives data from the data source external to GSN network. The use cases include bulk data transfer and media streaming. Regardless of data type being transfered both cases deal with multicasting data from one GSN1 source to many GSN2 receivers, which in turn send session data to their data clients - Retail Stores (RSs) or media clients. Any type and instance of data (bulk file, media stream) is transfered in GSN network in a distinct ARMTP session. Every active (also called 'master') GSN has a 'stand-by backup' GSN participating in the same data session. All GSN2 receivers, as well as original GSN1 sender, maintain local session data cache and equally participate in data recovery from their local caches. The only difference between 'active' and stand-by GSN is that stand-by does not have any data clients while it is working in backup mode.

The main goal of ARM Session Recovery is to detect any GSN failure and redistribute its load to a stand-by back-up GSN.

The following conditions are defined as GSN failure:

- 1) System crash
- 2) Any physical links failure that results in disconnection of the given GSN from GSN network.
- 3) Detection of network congestion local to the given GSN when receiving session data slows down considerably on given GSN

The pilot design does not provide any solution of GSN failure for cases 1), 2) and provides partial solution for case 3). The reasons for this are described later in this document.

Assumptions and interdependencies

- Multicast topology of all GSN nodes including stand-by backup nodes is provided by routing subsystem.
- Reconfiguration and liveliness of multicast topology in case of physical links failures constituting this topology is provided by routing subsystem.

Pilot design limitations.

The design is 'pilot' as it provides limited, partial solution to ARM session recovery. Design is limited because of several important requirements and interdependencies are remaining unclear or not fully specified at the time of writing this document.

Limitations and their reasons are:

- Recovery in case of GSN failure due to system crash is not supported.

  Reason: At the moment of this writing there is no mechanism of detecting node system crash.

- Recovery in case of GSN failure due to any physical links failure that results in disconnection of the given GSN from GSN network is not supported.

  Reason: At the moment of this writing there is no support in routing subsystem to detect link failure as well as clear solution for this.

- Partial recovery of GSN2s are only supported. Recovery for GSN1 is not supported. From this follows that when GSN1 one fails *all* GSN2s participating in *all* GSN1 sessions will fail too.

  Reason: For GSN1 to recover a stand-by GSN needs to work in parallel with its master receiving the same data at the same time from the same data source (DC or video server). This is the only way for GSN1 to be able to recover. Not a single GSN2 can start working at any given moment as primary GSN1 simply because it will never have complete session data in its cache at the moment of original GSN1 failure.

- GSN2 recovery is supported partially.

  Reason: There is no congestion detection for active GSN. Instead failure condition is detected according to indirect measurements performed by AA participating in ARMTP session.

- When session of some type on GSN2 fails all in-progress sessions with its clients are closed, this includes TFTP and RTP also.

  Reason: Currently we work in the model of "GSN failure" as opposed to "session failure". As a result in the current model we can not define partial failure and partial recovery. Where partial failure is a failure of some set of GSN sessions and not the total failure of GSN.

- Stand-by reopens all in-progress sessions from his master and starts them from the very first data packet.

  Reason: To resume in-progress sessions on stand-by node Ip spoofing support is required in routing subsystem. We don’t have this.

From these limitations also follows: Only GSN2s have stand-by recovery GSNs. GSN1 does not have stand-by recovery GSN.

Design principles

Pilot design is done according to the following principles:

- Recovery decision is made separately for every session type. That means that the decision on TFTP session recovery is independent from decision of RTP session recovery.

  Note: From this follows that in future when we have definition of session recovery as opposed to GSN recovery, then at one and the same time both master and stand-by GSN2 may work in parallel, servicing different types of sessions to their data clients. A

special case is when all session types from master are redirected to stand-by GSN2 (total GSN failure, that we work with today).

- AA managing particular session type on GSN2 itself decides when it fails.
- AA managing particular session type on GSN2 itself signals its stand-by GSN about the failure.
- TFTP AA and RTP AA on master (active) GSN2 sends OFL (offload request) to his stand-by GSN2 when they detect the failure independently from each other. This request is sent in context of failing ARM session as ARM_CMD(APPLICATION(OFL)) packet. Stand-by GSN acknowledges this request with OFL_ACK messages sent to master GSN in ARM_ACK(OFL) packet. For details of ARMTP commands, messages and packet format see [2].
- OFL request is separately confirmed by OFL_ACK from stand-by GSN2 for every session type.
- Session redirection OFL messages are implemented independently from session type with Session Message Sender (SMS) and Session Message Receiver (SMR) channels. SMS and SMR channels are implemented as a shared library of reusable channel / elements that can be deployed by any AA managing any session type.

Session Message Sender (SMS) channel

SMS channel is used by AA controlling ARMTP session (TFTP, RTP, etc.) to send OFL (offload) requests to stand-by GSN. OFL request is sent in context of failing ARM session as ARM_CMD(APPLICATION(OFL)) packet. Data carried in OFL request depends on session type.

SMS channel is also used by stand-by GSN to reply to his master OFL request with ARM_ACK(OFL) message.

OFL request contains the following info:

- IP address of stand-by GSN that must offload failed GSN.
- Failed GSN Node Session Status (NSS).

Node Session Status (NSS) depends on session type.

TFTP NSS contains the following information:

- ARMTP Session ID
- GSN IP
- GSN Cache Info. This includes the latest block written into cache on this GSN.
- Data Receivers Status (DRS). DRS is an array of records where each record describes current state of one Data Receiver (DR) (client of this GSN). DR record includes: DR IP, number of the last data block successfully received by this DR.

RTP NSS contains the following information:

TBD

Sending OFL request with ARM_CMD(APPLICATION(OFL))

New ARMTP command : ARM_CMD(APPLICATION(OFL)) must be implemented in ARMTP protocol to send OFL requests. This ARMTP command allows the ARM application to robustly transmit OFL request. The message is repeated ROBUST_FACTOR times at a rate of once per 2*GRTT. This rate of repeat allows the application to collect a response before it is repeated. For details see [2].

OFL commands is sent in parallel with ongoing data. OFL command is sent repeatedly with some time-out between sends. Both number of repetitions and send time-out are configurable parameters of this channel.

Session Message Receiver (SMR) channel

SMR channel is used by stand-by GSN to receive ARM_CMD(APPLICATION(OFL)) from his master.

SMR channel is used by master GSN to receive ARM_ACK(OFL) messages from its stand-by GSN. This message indicates that stand-by agreed to offload the session indicated in ARM_CMD(APPLICATION(OFL)) from his master.

Replying to OFL request with ARM_ACK(OFL) messages.

Stand-by GSN replies to every OFL request from his master. Nevertheless stand-by starts servicing offloaded to him session immediately after the first OFL request it sees.

Failure detection.

In pilot design the only event signalling GSN failure results from indirect detection of ARMTP congestion for the given session. Indirect detection of ARMTP congestion is achieved with measuring two parameters of ARMTP in run-time:

- MAX_NACK_NUMB: Maximum number of NACKs sent for the same block during some interval T.
- AVE_DATA_RATE: Average rate of incoming data packets during some interval T.

Configuration

In pilot design Session Recovery is configured on every master and stand-by GSN with configuration files. Master GSN configuration goes into master_session_recovery.cfg file that contains the following parameters:

- Stand-by GSN IP. This IP denotes stand-by GSN that will back-up this GSN.
- OFL_TIMEOUT : time-out between OFL requests.
- OFL_NUMB : number of OFL requests to send.
- Array of TFTP Client Entries (TFTP_CE). Each entry contains client IP.
- MAX_NACK_NUMB, NACK_TIME_INTERVAL : Maximum number of NACKs sent for the same block during NACK_TIME_INTERVAL interval. This parameter is used to detect congestion.
- AVE_DATA_RATE, DATA_RATE_TIME_INTERVAL : Average rate of incoming data packets during DATA_RATE_TIME_INTERVAL time interval. This parameter is used to detect congestion.

•

Stand-by GSN configuration goes into standby_session_recovery.cfg file that contains the following parameters:

- Master GSN IP. This IP denotes master GSN that this GSN will back-up.
-

References

[1] Kondratiev, Dmitri *ARM Session Recovery and Load Balancing*

[2] Kondratiev, Dmitri *ARM Transport - Active Reliable Multicast Transport Protocol*

Pilot ARM Session Recovery Design.