# Technical notes

# Table of Contents

# Multi tenancy support

Multi tenancy allows same instance of Restcomm jSS7 to connect to different operators, each having its own links and point-codes.

Multi tenancy is achieved by having a model of SS7 network as a splitted to several logical networks. Each logical network has a corresponded key "networkId". It is digital one and default value (if we do not specify it) - 0.

NetworkId value is assigned for a configurable objects:

- SCCP SAP
- SCCP Rule

NetworkId is also assigned to activities:

- SccpMessage (assigned by Service Access Point (SAP) for SS7 originated messages, assigned by TCAP Dialog for TCAP originated messages)
- TCAP Dialog (assigned by SccpMessage for SCCP originated messages, assigned by upper Dialog for user messages)
- MAP / CAP Dialog (same as TCAP Dialog)
- SCCP rule is taken into account only when SCCP rule networkId == SccpMessage networkId

When an SCCP message comes to SCCP stack the a networkId is assigned to the SCCP message depending on DPC and OPC values of the message. In order to SCCP stack can properly assign a networkId value, then at least either DPC or DPC of messages for different networkId subnetworks must be different and SAPs / Destinations for SAPs were configured for unintersected intervals.

# Congestion control for outgoing messageflow

The target of outgoing congestion control is to report to upper stacks that SS7 links are congested (or down) so upper stacks / SS7 users can reduce traffic load or stop of message sending. Overload / disconnection cases are calculated per a networkID area.

For outgoing messageflow (when we are sending messages to SS7 network) low level counters are implemented at SCTP library (MTP2 level for the sigtran stack). SCTP library measures delays between message delivering from MTP3 level for delivering and a real sending to the IP network. There are 3 levels of congestion (1, 2 or 3) depending on the delay length. Then m3ua checks this data and issues MTP-STATUS indications to upper layers (ISUP or SCCP). If we use MTP3 stacks from Dialogic or Dahdi legacy cards, we use MTP-STATUS indications that are issued by MTP3 levels of these cards. SCCP level has a congestion control from SS7 specification, where SCCP by timers TIMER_A and TIMER_D processes a congestion measurement logic. At the SCCP level the blocking of outgoing SCCP messages is also implemented (it is disabled (and not used) by default but can be activated).

SCCP and M3UA levels use the outgoing congestion control results for sharing of the outgoing SS7 traffic between primary / backup links. When primary (for dominant scenario) or some link (for loadsharing scenario) is congested up to at least level 2 SCCP stack routes the traffic to a backup / another link. M3UA stack when loadsharing between ASP destinations when some link for loadshare traffic-mode is congested up to at least level 2 also tries to route the traffic to another link. (For loadsharing between AS destinations the congestion levels are not taken into account).

TCAP level (and upper stack levels) provides methods for the congestion control monitoring for SS7 stack users:

```
    /**
     * The collection of netwokIds that are marked as prohibited or congested.
     *
     * @return The collection of pairs: netwokId value - NetworkIdState (prohibited /
congested state)
     */
    FastMap<Integer, NetworkIdState> getNetworkIdStateList();

    /**
     * Returns the state of availability / congestion for a networkId subnetwork.
     * Returns null if there is no info (we need to
     * treat it as available)
     *
     * @param networkId
     * @return
     */
    NetworkIdState getNetworkIdState(int networkId);
```

By these methods the TCAP stack provides the congestion and availability status for a SS7 subnetwork for a networkId. NetworkIdState has two parameters:

```
    boolean isAvailavle();
    int getCongLevel();
```

and helps us to understand if the networkId is not available (connected) / not available (down) or congested / not. Then TCAP / CAP / MAP user can check it and reduce / temporary block the outgoing traffic.

At the SCTP and SCCP levels we have several congestion control parameters that should be tuned depending on connection type (legacy / sigtran) and capacity for the best productivity. See chapter [_sctp_property_cc_delaythreshold] for configuring of congestion control settings for SCTP level and chapters [_sccp_property_cc_timer_a], [_sccp_property_cc_timer_d], [_sccp_property_cc_algo] and [_sccp_property_cc_blockingoutgoungsccpmessages] for configuring of congestion control settings for SCCP level.

# Congestion control for incoming messageflow

The target of incoming congestion control is to measure of server load and prevent of accepting new incoming from a peer new dialogs / messages. This load info can also be used by SS7 stack users for reduce of generating of traffic load.

There are following sources of load measuring:

- ExecutorCongestionMonitor - measuring of how much time events wait at a Executor queue before processing start. This CongestionMonitors are added at the MTP stack levels and reports results to TCAP level.
- MemoryCongestionMonitor - measuring a percentage of usege of memory.
- UserPartCongestionMonitor - TCAP users can provide to TCAP level extra info of congestion to some external for SS7 stack resourse.

Then TCAP level calculate a cumulative congestion level (the maximul level is taken). There are 4 levels of congestion (0-3). 0 level means no congestion, 3 level means max congetion. When [_tcap_property_blockingincomingtcapmessages] option is turned on (disabled by default) the in case of 2 congestion level all incoming new dialogs are rejected and in case of 3 congestion level - all incoming TCAP messages are rejected.

By a following method of TCAPProvider interface upper levels can provide a congestion level of a TCAP user part:

```
    /**
     * Setting of a congestion level for a TCAP user "congObject"
     *
     * @param congObject a String with the name of an object
     * @param level a congestion level for this object
     */
    void setUserPartCongestionLevel(String congObject, int level);
```

By a following methods of TCAPProvider interface upper levels can get info of a congestion level:

```
    /**
     * Returns a congestion level of a Memory congestion monitor
     *
     * @return
     */
    int getMemoryCongestionLevel();

    /**
     * Returns a congestion level of thread Executors for processing of incoming
messages
     *
     * @return
     */
    int getExecutorCongestionLevel();

    /**
     * Returns a max congestion level for UserPartCongestion, MemoryCongestion and
ExecutorCongestionLevel
     *
     * @return
     */
    int getCumulativeCongestionLevel();
```

See configuring parameters of Executors congestion control in [_tcap_property_executordelaythreshold] chapter and configuring parameters of Memory congestion control in [_tcap_property_memorythreshold] chapter.