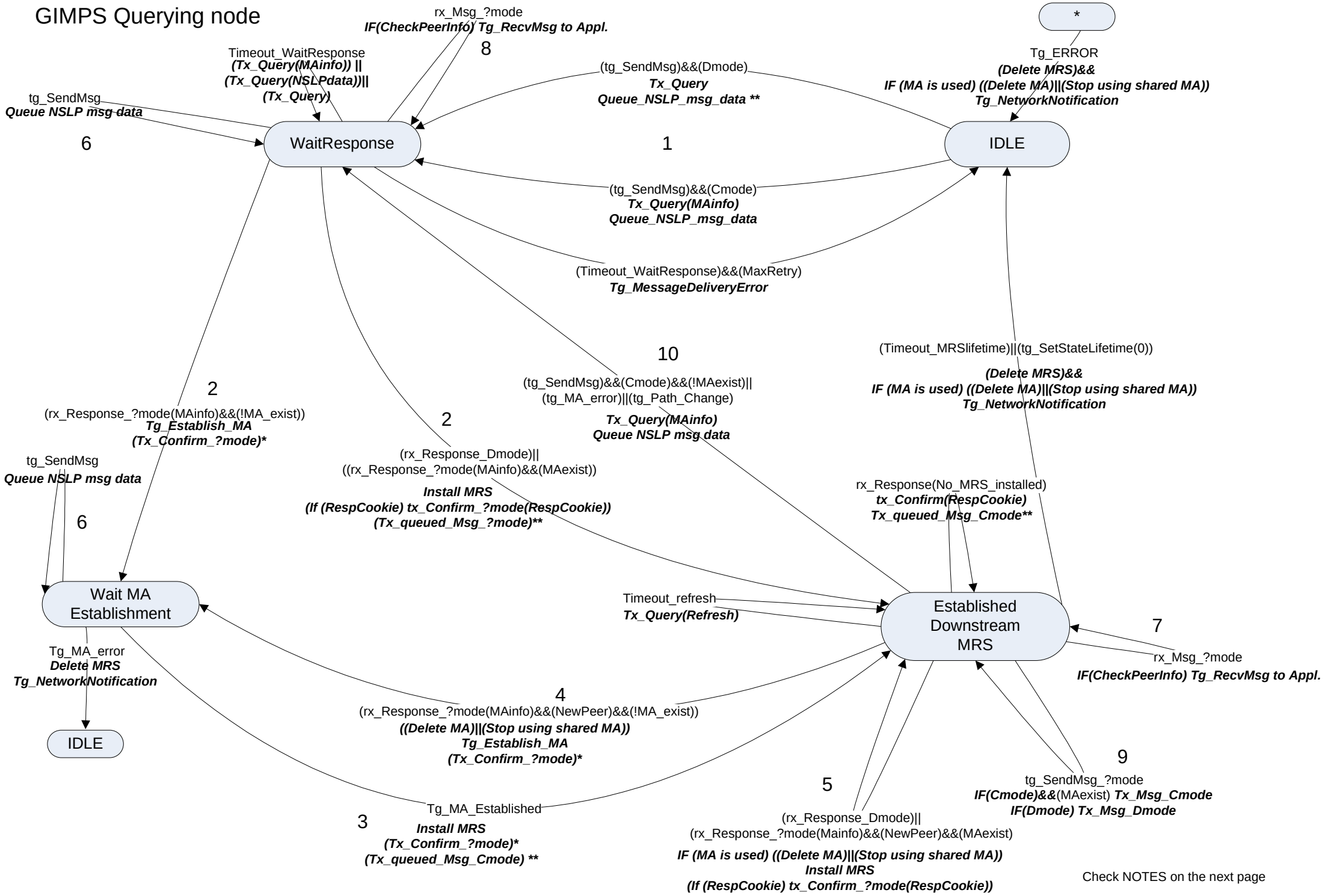
GIMPS Querying node
rx_Msg_?mode
IF(CheckPeerInfo) Tg_RecvMsg to Appl.
8
Timeout_WaitResponse
(Tx_Query(MAinfo)) ||
(Tx_Query(NSLPdata))||
(Tx_Query)
tg_SendMsg
Queue NSLP msg data
6
WaitResponse
(tg_SendMsg)&&(Dmode)
Tx_Query
Queue_NSLP_msg_data **
1
*
Tg_ERROR
(Delete MRS)&&
IF (MA is used) ((Delete MA)||(Stop using shared MA))
Tg_NetworkNotification
IDLE
(tg_SendMsg)&&(Cmode)
Tx_Query(MAinfo)
Queue_NSLP_msg_data
(Timeout_WaitResponse)&&(MaxRetry)
Tg_MessageDeliveryError
10
(tg_SendMsg)&&(Cmode)&&(!MAexist)||
(tg_MA_error)||(tg_Path_Change)
Tx_Query(MAinfo)
Queue NSLP msg data
(Timeout_MRSlifetime)||(tg_SetStateLifetime(0))
(Delete MRS)&&
IF (MA is used) ((Delete MA)||(Stop using shared MA))
Tg_NetworkNotification
2
(rx_Response_?mode(MAinfo)&&(!MA_exist))
Tg_Establish_MA
(Tx_Confirm_?mode)*
2
(rx_Response_Dmode)||
((rx_Response_?mode(MAinfo)&&(MAexist))
Install MRS
(If (RespCookie) tx_Confirm_?mode(RespCookie))
(Tx_queued_Msg_?mode)**
tg_SendMsg
Queue NSLP msg data
6
rx_Response(No_MRS_installed)
tx_Confirm(RespCookie)
Tx_queued_Msg_Cmode**
Wait MA
Establishment
Timeout_refresh
Tx_Query(Refresh)
Established
Downstream
MRS
7
rx_Msg_?mode
IF(CheckPeerInfo) Tg_RecvMsg to Appl.
Tg_MA_error
Delete MRS
Tg_NetworkNotification
4
(rx_Response_?mode(MAinfo)&&(NewPeer)&&(!MA_exist))
((Delete MA)||(Stop using shared MA))
Tg_Establish_MA
(Tx_Confirm_?mode)*
IDLE
9
tg_SendMsg_?mode
IF(Cmode)&&(MAexist) Tx_Msg_Cmode
IF(Dmode) Tx_Msg_Dmode
5
(rx_Response_Dmode)||
(rx_Response_?mode(Mainfo)&&(NewPeer)&&(MAexist)
IF (MA is used) ((Delete MA)||(Stop using shared MA))
Install MRS
(If (RespCookie) tx_Confirm_?mode(RespCookie))
3
Tg_MA_Established
Install MRS
(Tx_Confirm_?mode)*
(Tx_queued_Msg_Cmode) **
Check NOTES on the next page

* Response and Confirm messages might be send either in Dmode or Cmode, before or after MA establishment depending on node's local 3-way handshake policy and the availability of MAs to be reused. See draft for details.
** Depending on the local policy NSLPdata might be send as payload of Query and Confirm messages. (piggybacking)

1) Initial request from NSLP are received, which triggers Query messages requesting either D_mode or C_mode. Dependign on node's local policy NSLP data might be piggybacked in the Query requesting D_mode.
2) Response message is received. If C_mode connections must be established and there is no available MA to be reused, MA establishment is initiated and waited to be completed. If D_mode connection is requested or available MA can be reused if C_mode is requested the MRS is established.
3) New MA is successfully established and MRS, which will use it, is installed.
4) Path change detected events – local recovery procedure, where D_mode or C_mode with available MA must be established. THIS IS VALID ONLY IF THE NODE IS CROSSOVER NODE
5) Path change detected events – local recovery procedure, where new MA must be established for requested C_mode connection. THIS IS VALID ONLY IF THE NODE IS CROSSOVER NODE
6) NSLP data is queued, because downstream peer is not discovered or required MA is still not established.
7) Received Data messages are checked if their sender matches the installed downstream peer info in the MRS and then processed.
8) Received Data messages are checked if their sender matches the installed downstream peer info in the MRS and then processed. In WaitResponse state, this event might happen in the process of MA upgrade, when the downstream peer is still not aware of establishment of the new MA.
9) Depending on the requested transport from NSLP and currently established D_mode or C_mode, NSLP message is sent D_mode if D_mode is requested and C_mode if the features of the used MA covers the required transport (e.g. used MA is reliable and NSLP request reliable but not secure transport)
10) External event notifies for Path Change and discovery procedures is restarted. THIS IS VALID ONLY IF THE NODE IS CROSSOVER NODE OR NSLP requests C_mode transport that is not covered by currently used D_mode or MA (case of MA upgrade) and discovery procedure is restared but current downstream peer info is kept in order to be able to receive messages from it during the upgrade process.