Message Scheduling Module API

- Message Scheduling Module(MSM) should know the limit on allowed outstanding messages it can have before the start of any PT(Power Transfer Phase), which has to be set by calling,

  void Set_My_Kmax_And_Phase_Time(_My_KmaxLocal, _Phase_Time);

- Peer Management:
  - A new Peer can by added by a call to,

    void Add_Peer(_PeerID, _Obs_RTT);

    similarly a Peer can be deleted by,

    void Delete_Peer(_PeerID);

    Note: As of now PeerID's are configured to be 'int' type which can be changed to 'UUID' type.
  - To get total number of Peers participating in PT, call

    int Get_Number_Of_Peers();

    or if map(data structure) of all Peers is required for debugging purpose, then call

    map<int, Peer*> *Get_Map_Of_Peers();

Message Scheduling Module API

- Delete all Peers by a call to,

  void Delete_All_Peers();

  Useful at the end of PT

- Current MSM tick can be accessed by,

  MS_Tick_Type Get_Current_Tick()

  will not need it, useful for debugging

- If required wait (delay/sleep) can be done by,

  void Wait(MS_Tick_Type _tick);

  It will wait for desired number of MSM ticks (useful to wait until Invariant gets satisfied)

- Following function should never be called, only for MSM internal use

  void Tick_Changed();

  MS_Agent being singleton class, this function is kept in public.

  MS_Tick calls -> MS_Agent::Tick_Changed() -> which calls Peer::Check_Deadline_Miss()

  Note: MS_Tick, MS_Agent both are singleton class

Message Scheduling Module API

- EVENTS:
  - MSM should be notified of the event whenever a new power message is sent, by a call to,

    void Event_Msg_Sent(_PeerID, _MsgID);

    Note: A message should never be sent without checking the invariant
  - MSM should be notified of the event whenever a acknowledgment is received for a message sent earlier, by a call to,

    void Event_Msg_Ack_Received(_PeerID, _MsgID);
  - MSM should be notified of the event whenever a power granted message is received from the excess power node

    void Event_Msg_Received(_PeerID, _MsgID);

    Note: This function is not implemented yet, this module will work only for the node which has excess power
  - MSM should be notified if the message is lost in the network. If required timeout mechanism can be implemented in MSM module.

    void Event_Msg_Lost(_PeerID, _MsgID);

    Note: Not implemented

Message Scheduling Module API

- Invariant checking

  Before any new power message is sent out, MSM invariant has to be satisfied.

  bool Invariant_Check(_PeerID);

  Note: Sample main.cpp included in the package should give a good idea.

- Important:

  Should take a look at MsgSchedModule_Intrnl_Wrkng.pdf