1 BMSim: the BM network Simulator

BMSim is a simulation framework for BM networks. the main advantages of BMSim simulator are as follows. BMSim is accurate enough that can be trusted for performance evaluation of BM networks (verified by real experiments). Second, low level details of packet transmission and networking procedures (both medium access and network layers) are truly modeled so that various performance metrics can be estimated. Third, the simulator is modular and flexible so that extensions for inclusion of any channel/radio/mobility/interference models are possible and straightforward. As the last and very important specification, it is possible to configure the network and its parameters during simulation so that dynamic networks with run-time configurations mechanisms can be simulated.

BMSim is an open-source and publicly available event-driven Bluetooth mesh network simulator. Python is used as the programming language for implementation of the simulator. We first introduce the architecture of the simulator, and then discuss its core operation and user interfaces.

Fig. 1 shows the general architecture of BMSim and its inner components. A simulation starts by receiving some high-level network specifications from the user. The *Initializer* module prepares the first snapshot of the node deployment in the specified simulation area according to user wishes. Then the *Updater* module runs to prepare required inputs for the simulator engine. The updater module runs every T_{update} seconds to support network dynamism as well as run-time configuration changes. The *BM simulation engine* is a discrete event simulator, for which the events and their timings are made according to the BLE and BM standard protocols. This engine continues making and processing events until the requested simulation time (T_{sim}) is reached. The *Logger* module creates the required output files while the simulation engine is running, and finally calculates and reports various performance indicators of the simulated BM network.

1.1 Initializer Module

The initialize module is the entry point user interface for the simulator by which the user initial settings are received and compiled into the simulation framework. It gets the number of nodes in the network (network size, N) and the dimensions of the area the nodes should be deployed in. Also, the simulation time (T_{sim}) is the time frame that the user expects the network to be simulated. Initializer distributes N instances of the node model all over the simulation area based on the topology parameters given by user; it may be either uniformly random distribution, a regular grid structure, or specifying the x-y coordinations of all nodes. The last case is especially used when several simulation runs are expected to be simulated starting from the same deployment. Anyways, the initializer only determines the initial locations of nodes, which remain fixed all over the simulation time if no node mobility is requested (static networks). In case of having node mobility, the mobility model block of the network updater module determines the new locations of the mobile nodes at each simulation step.

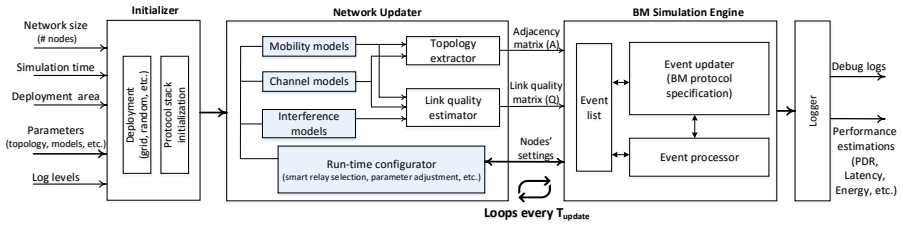

Figure 1: The architecture of the developed Bluetooth mesh event-driven simulator (BMSim)

There are a number of parameters related to different models used in the next components of the simulator that are received from user by the initializer module and are directly passed to the corresponding blocks. Parameters of the mobility/channel/radio/interference models, BM stack initial configuration and node types, and the level of expected logs are other inputs of the initializer module. Note that since the simulator is expected to be easily configurable by plugging in/out various models, the exact list of input parameters will differ. For instance, for link extraction, user can simply specify the communication range of nodes or the Packet Reception Ratio (PRR) of links between each pair of nodes to have a high level modeling of channel behavior without simulating details of channel and radio. However, to have more precise simulation, one can use a path-loss model, for instance, for which several parameters such as transmission power of nodes, receiver sensitivity, and channel path-loss exponent need to be set. The same applies to other models such as mobility and interference models.

1.2 Network Updater Module

This module is essential for supporting dynamism as well as the possibility of having run-time adaptations and (re)configurations. The procedures in this module run every T_{update} to update the network. It is obvious that using lower time interval for updates results in more smooth changes (e.g., smoother mobility patterns), but leads to higher execution time. On the other hand, to have faster simulation, longer intervals can be used especially when dynamism level is low or parameter changes are not very frequent.

There are two categories of outputs from the network updater module to the simulation engine. The first category of outputs is related to the network topology and link quality, namely the adjacency ($A_{N\times N}$) and link quality ($Q_{N\times N}$) matrices. Each entry $a_{i,j} \in A$ is a binary value that determines if there is a link between nodes i and j while $q_{i,j} \in Q$ gives the probability of successful packet delivery over that link. The topology extractor and link quality estimator blocks receive inputs from the mobility (nodes' positions), channel, and interference models and produce matrices A and Q for the next simulation step. The other set of outputs generated by the network updater module is the BM protocol parameter settings of each individual node in the network. The settings for a

node include node's features (packet generator, relay, low-power, friend), packet generation interval (T_{gen}), advertising interval (T_{adv}), scan interval ($T_{scanInt}$), scan window ($T_{scanWin}$), network and relay retransmit count, Ntis, Rris, heartbeat message transmission interval (T_{hb}) of the destination nodes, data poll interval for low-power nodes, etc. The value of these parameters are set for the next simulation step based on the adaptation mechanisms that are implemented within the run-time configurator block of the network updater. As an example, determining a subset of nodes in a BM network that are best candidates for being relay is of paramount importance for achieving the required network performance. Relay nodes should be in the locations such that isolated nodes are minimized. A large number of relay nodes leads to unnecessary traffic and thus collisions, degrading the network performance. On the other hand, low number of relays affect the functionality and robustness of the flooding mechanism for data delivery. Thus, some intelligence can be used here to select the best nodes for the relay role; it may be design-time for static networks or run-time for dynamic networks. The current version of BMSim is fortified with a design-time relay selection algorithm based on graph betweenness and closeness centrality algorithm [1]. Like the other blocks, this can be replaced or extended with other mechanisms for relay selection or adaptations of the other parameters.

1.3 BM Simulation Engine

The core of the BM simulator is a discrete event simulator that is composed of three blocks. The event updater block runs BM protocol specification, generates events (event id and expiry time) and adds them to the event list. The event processor block controls the simulation time and allows it to elapse only if all events at the current simulation time are processed. The event processor picks an event from the top of the list (the event with the earliest expiry time) and processes it, which may lead to generation of more events (performed in line with the event updater block). Some of the defined and processed events are data packet generation event (made periodically every T_{gen} by each source node), heartbeat message generation event (made periodically every T_{hb} by destination nodes), relay event (every T_{adv} by each relay node if it has data packet in its buffer), Advertising37, Advertising38, Advertising39, Scan37, Scan38, Scan39, and channel switch event.

1.4 Logger Module

To closely study the behavior of BM networks and for post-processing towards performance extraction, various log files may be generated during a simulation run. Since the simulator is open-source for public, users can add logs for whatever parameters and wherever in the protocol execution they aim for. However, There are already some logs produced by the simulator to report important parameters that have interest of researchers. Such logs have two modes, determined by user, that controls the level of details being logged. If the detailed log flag is on, a separate log file is produces per node that contains details of the

behavior of the packet generator nodes and the relay nodes. In the generator nodes' log file, the source ID, packet destination ID, packet generation time, sequence number, and the type of packets (heartbeat or data main packet) are logged for each generated packet. In the relay nodes' log, the advertiser node, the source node, packet TTL and sequence number, advertising and receiving and generation time, type of packets, and the number of packets in the node's buffer are logged. This information help users to track the packet's path across the network. If the detailed log flag is off, per node log files are not produced. Anyways, a general log file is generated in every simulation run that contains required information for evaluating the performance metrics (i.e., PDR, latency, burst packet loss, and energy consumption). This log file includes the following items per delivered packet as source and destination node IDs, packet generation time, packet delivery time to destination, packets type, and packets sequence number. For calculating energy consumption, the time duration that each node has spent in each operation mode (transmission, reception, sleep, and switch) and the source node, is logged. Then energy consumption is calculated based on this timing logs as well as the power consumption profile of the used radio transceiver (provided by user).

References

[1] A. Syarif, A. Abouaissa, and P. Lorenz, "Operator calculus approach for route optimizing and enhancing wireless sensor network," *Journal of Network and Computer Applications*, vol. 97, pp. 1–10, 2017.