Description of code flow for incoming and outgoing packets

Iyappan Ramachandran

September 16, 2005

Description of packet flow

Since no real documentation exists on the NS-2 implementation, we provide here a fundamental description of the flow of code when a packet outgoing or an incoming packet is received by the MAC.

Outgoing

- The upper layer hands down a packet to MAC by calling Mac802_15_4::recv(Packet *p, Handler *h).
- recv() then calls Mac802_15_4::mcps_data_request( ). mcps_data_request() works by using a variable called step, which is initialized to 0. Every time the function needs to pass control to a different function it increments step so that when the control returns to it, it will know where to proceed next.
- For direct transmission mcps_data_request() then calls Max802_15_4::csmacaBegin(pktType) after incrementing step to 1.
- csmacaBegin in turn calls CsmaCA802_15_4::start(). start() calculates the random backoff time, determines if it can proceed and starts the macBackoffTimer with the backoff time determined.
- On expiry of the timer, CsmaCA802_15_4::backoffHandler() is called. backoffHandler then turns on the receiver (through Phy802_15_4::PLME_SET_TRX_STATE_request()) and requests a CCA by calling Phy802_15_4::PLME_CCA_request().
- CCA is done exactly at the end of the 8th symbol time (i.e. 192μs) by starting the CCAH timer with 8 symbol durations, which on expiry calls Phy802_15_4::CCAHandler. CCAHandler determines if the channel is idle and reports its finding by calling Mac802_15_4::PLME_CCA_confirm().

- If the channel is found to be idle and if CW≠0, CsmaCA802_15_4::CCA_confirm() called by PLME_CCA_confirm() decrements CW and in turn calls backoffHandler to perform CCA again. If CW=0, it calls Mac802_15_4::csmacaCallback(). If the channel is found busy, CsmaCA802_15_4::start() is called to go to the next backoff stage.
- csmacaCallback() subsequently returns control to Mac802_15_4::mcps_data_request(), which enables the transmitter by calling Phy802_15_4::PLME_SET_TRX_STATE_request() after incrementing step to 2.
- Mac802_15_4::PLME_SET_TRX_STATE_confirm() then passes the data to Mac802_15_4::txBcnCmdDataHandler(), which uses Mac802_15_4::sendown() to give the data to Phy802_15_4::recv(Packet *p, Handler *h)
- recv() then calls Phy802_15_4::PD_DATA_request(), which in turn uses WirelessPhy::sendDown() (Phy802_15_4 is a sub-class of WirelessPhy) to decrement energy and transmit the data to Channel::recv().

Incoming

- Channel::recv() then gives one copy of the packet to each node using WirelessChannel::sendUp(), which subsequently passes the packet to Phy802_15_4::recv() after propagation delay.
- Phy802_15_4::recv() uses WirelessPhy::sendUp() to decrement energy and indicates packet reception to MAC using Phy802_15_4::PD_DATA_indication(). Phy802_15_4::recvOverHandler() involved here drops packets not intended for the node. PD_DATA_indication() then calls Mac802_15_4::recv(Packet *p, Handler *h).
- recv() drops the packet if there is a collision or calls Mac802_15_4::recvData() if there is no collision. recvData() then calls Mac802_15_4::MCPA_DATA_indication(), which passes the data to the upper layer.

In the following the changes made to the NS-2 mode are described along with the reasons for the modifications and list of files affected.