

Manual do modelo TSCH para ns-3

Versão 1.0

Luis Pacheco

1

`luisbelem@gmail.com`

Sumário

1	Introdução	3
2	Interface	3
2.0.1	LrWpanHelper	3
2.0.2	LrWpanMac	3
3	Exemplo	4

1. Introdução

Este manual descreve a implementação do protocolo Timeslotted Channel Hopping (TSCH) para o ns-3. O código fonte está na pasta "ns3/src/lr-wpan", este módulo é composto por outras funcionalidades do protocolo IEEE 802.15.4 que já estavam presentes no ns-3.

Nem todas as funcionalidades descritas na emenda IEEE 802.15.4e foram implementadas, os seguintes serviços estão disponíveis:

- MLME-SET-SLOTFRAME: responsável por configurar slotframes;
- MLME-SET-LINK: responsável por configurar links;
- MLME-TSCH-MODE: responsável por habilitar e desabilitar o TSCH.

Seguindo o padrão do ns-3, um helper foi implementado para facilitar a criação de simulações. Este helper possui diversas funções que são utilizadas rotineiramente.

Este manual tem o objetivo de apresentar o modelo TSCH, a documentação de instalação e utilização do simulador ns-3 pode ser encontrada no site www.nsnam.org.

2. Interface

A seguir a interface das classes do modelo TSCH é apresentada. Apenas métodos públicos introduzidos pela implementação do TSCH no modelo lr-wpan são descritos.

2.0.1. LrWpanHelper

void AddSlotframe(NetDeviceContainer nodes, uint8_t slotframehandle, uint16_t size)
adiciona o slotframe de identificador *slotframehandle* e tamanho *size* nos dispositivos *nodes*;

void ModSlotframe(NetDeviceContainer nodes, uint8_t slotframehandle, uint16_t size)
modifica o slotframe dos dispositivos *nodes*, setando o identificador para *slotframehandle* e o tamanho para *size*;

void AddLink(Ptr<LrWpanNetDevice> src, Ptr<LrWpanNetDevice> dst, AddLinkParams params)
adiciona um novo link de transmissão do nó *src* ao nó *dst* com os parâmetros *params*;

void ModifyLink(Ptr<LrWpanNetDevice> src, Ptr<LrWpanNetDevice> dst, AddLinkParams params)
modifica o link dos nós *src* e *dst* para os parâmetros *params*;

void DeleteLink(Ptr<LrWpanNetDevice> src, Ptr<LrWpanNetDevice> dst, AddLinkParams params)
remove o link identificado nos parâmetros *params* dos nós *src* e *dst*;

AddLinkParams estrutura com os parâmetros de um link: *slotframeHandle* (inteiro) identificador do slotframe, *linkHandle* (inteiro) identificador do link, *timeslot* (inteiro) número do timeslot a conter o link e *channelOffset* (inteiro) *offset* para a seleção do canal da comunicação.

2.0.2. LrWpanMac

void AddPktToQueue(Ptr<Packet> pkt, uint8_t msdu) adiciona o pacote *pkt* de tamanho *msdu* ao buffer;

void SetDefaultHoppingSequence(uint16_t sequenceLength) utiliza a lista de saltos padrão de tamanho *sequenceLength*;

void SetHoppingSequence(std::vector<uint8_t> sequence, uint8_t id) utiliza a lista de saltos *sequence* e identificador *id*;

void TschRemovePktFromBuffer() remove o primeiro pacote do buffer;

void TschClearQueue() limpa o buffer;

int TschQueueSize() retorna a quantidade de pacotes presentes no buffer;

void SetTschAttributes(LrwpanMacTschPibAttributes att) configura os atributos relacionados ao TSCH com valores em *att*;

void MlmeTschModeRequest (MlmeTschModeRequestParams params) realiza uma requisição para habilitar ou desabilitar o TSCH, de acordo com *params*;

void MlmeSetSlotframeRequest (MlmeSetSlotframeRequestParams params) adiciona, modifica ou remove um slotframe, de acordo com *params*;

void MlmeSetLinkRequest (MlmeSetLinkRequestParams params) adiciona, modifica ou remove um link, de acordo com *params*;

void SetMlmeSetSlotframeConfirmCallback (MlmeSetSlotframeConfirmCallback c) registra o callback *c* do tipo *MlmeSetSlotframeConfirmCallback*;

void SetMlmeTschModeConfirmCallback (MlmeTschModeConfirmCallback c) registra o callback *c* do tipo *MlmeTschModeConfirmCallback*;

void SetMlmeSetLinkConfirmCallback (MlmeSetLinkConfirmCallback c) registra o callback *c* do tipo *MlmeSetLinkConfirmCallback*;

MlmeSetLinkRequestParams estrutura com os parâmetros para a chamada do serviço *MlmeSetLinkRequest*;

MlmeSetSlotframeRequestParams estrutura com os parâmetros para a chamada do serviço *MlmeSetSlotframeRequest*;

MlmeTschModeRequestParams estrutura com os parâmetros para a chamada do serviço *MlmeTschModeRequest*;

LrwpanMacTschPibAttributes estrutura com os parâmetros de configuração do TSCH,

3. Exemplo

A seguir uma simulação utilizando o modelo TSCH é apresentada passo a passo. O exemplo “tsch-simple.cc”, localizado na pasta “scratch”, simula uma rede simples onde um dispositivo envia dados a outro.

```

1 #include <ns3/core-module.h>
2 #include <ns3/lr-wpan-module.h>
3 #include <ns3/mobility-module.h>
4
5 NS_LOG_COMPONENT_DEFINE ("TschSimple");
6
7 using namespace ns3;
```

Listing 1. Primeiramente são incluídos os módulos utilizados na simulação, o modelo referente ao TSCH é o *lr-wpan-module*.

```

1 // Configuration
2 int pktsize = 114; //size of packet in bytes, 114 is the
   maximum allowed
```

```
3 double duration = 100; //duration of simulation
```

Listing 2. Duas variáveis de configuração foram criadas para facilitar a modificação, referentes ao tamanho do pacote e a duração da simulação.

```
1 void SchedulePackets (Ptr<LrWpanNetDevice> dev, Address
   addr) {
2     Ptr<Packet> pkt = Create<Packet> (pktsize);
3     dev->Send(pkt, addr, 0x86DD);
4
5     Simulator::Schedule(Seconds(0.01), &SchedulePackets,
        dev, addr);
6 }
```

Listing 3. Uma função é utilizada para gerar os pacotes, o ns-3 dispõe de módulos que simulam as camadas superiores, mas neste exemplo utilizamos diretamente a função Send de um NetDevice, pois estamos trabalhando apenas com as camadas física e de enlace.

```
1 int main (int argc, char** argv) {
2     bool g_verbose = false;
3     CommandLine cmd;
4     cmd.AddValue ("verbose", "Print_trace_information_",
        if_true", g_verbose);
5     cmd.Parse (argc, argv);
6     LrWpanHelper lrWpanHelper;
7
8     // Logging
9     if (g_verbose) {
10         lrWpanHelper.EnableLogComponents ();
11     }
```

Listing 4. A função main contém o código da simulação, primeiramente um comando é criado para habilitar o log em caso de necessidade de inspecionar o funcionamento dos módulos utilizados.

```
1 NodeContainer lrwpanNodes;
2 lrwpanNodes.Create (2);
```

Listing 5. Dois nós são criados.

```
1 // Mobility
2 MobilityHelper mobility;
3 Ptr<ListPositionAllocator> nodePositionList = CreateObject<
   ListPositionAllocator> ();
4
5 nodePositionList->Add (Vector (5.0, 5.0, 0.0));
6 nodePositionList->Add (Vector (0.0, 0.0, 0.0));
7
8 mobility.SetPositionAllocator (nodePositionList);
9 mobility.SetMobilityModel ("ns3::
   ConstantPositionMobilityModel");
```

```

10
11 mobility.Install (lrwpanNodes);

```

Listing 6. A travez do módulo Mobility os nós são posicionados, o nó 0 está na posição 5,5,0 e o nó 1 está na posição 0,0,0.

```

1 // Configure lrwpan nodes
2 NetDeviceContainer netdevs = lrWpanHelper.Install (
    lrwpanNodes);
3 std::vector<Ptr<LrWpanNetDevice> > devs;
4 for (uint8_t i = 2;i>0;i--)
5     devs.push_back(DynamicCast<LrWpanNetDevice> (
        netdevs.Get(i-1)));
6
7 //Pre associate nodes
8 lrWpanHelper.SetCoordNode(devs[0],1);
9 lrWpanHelper.ForceAssociateNode(devs[0],netdevs);

```

Listing 7. A classe LrWpanHelper possui vários métodos que facilitam a criação e configuração de dispositivos TSCH. Primeiramente o método Install é utilizado para adicionar as camadas física e de enlace nos nós. Em seguida um array de ponteiros para o LrWpanNetDevice é criado. O método SetCoordNode atribui o dispositivo 0 como coordenador e o identificador da rede como 1. O método ForceAssociateNode associa os dispositivos LrWpan à rede em que o dispositivo 0 é coordenador.

```

1 //Set channel hopping
2 int channel_hopping_size = 16; //number of channels
3 int channel_hopping_list[] =
    {11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26};
4
5 LrWpanMacChannelHopping chparams;
6 chparams.m_macHoppingSequenceList.clear();
7
8 chparams.m_macHoppingSequenceList.resize(
    channel_hopping_size);
9
10 for (unsigned char i = 0;i<channel_hopping_size;i++)
11     chparams.m_macHoppingSequenceList[i] =
        channel_hopping_list[i];
12
13 for (uint8_t i = 0;i<2;i++)
14     devs[i]->GetMac()->SetHoppingSequence(chparams.
        m_macHoppingSequenceList,1); //id, anything but
    0

```

Listing 8. Em seguida a lista de canais a serem utilizados é configurada, esta etapa é realizada diretamente nos dispositivos. Primeiramente um vetor de inteiros é criado com os canais utilizados, em seguida o tipo de dados LrWpanMacChannelHopping é instanciado e configurado, e por fim atribuído aos dois dispositivos.

```

1 //Add slotframe and links
2 lrWpanHelper.AddSlotframe(netdevs, 0,1);
3 AddLinkParams alparams;
4 alparams.slotframeHandle = 0;
5 alparams.channelOffset = 0;
6 alparams.linkHandle = 0;
7 alparams.timeslot = 0;
8 lrWpanHelper.AddLink (devs[1],devs[0],alparams);

```

Listing 9. O método `AddSlotframe` é responsável por criar slotframes nos dispositivos, o segundo argumento é referente à identificação do slotframe, e o terceiro ao número de timeslots.

```

1 // Tracing
2 lrWpanHelper.EnablePcapAll ("tsch-simple", true);
3 AsciiTraceHelper ascii;
4 Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("
    tsch-simple.tr");
5 lrWpanHelper.EnableAsciiAll (stream);

```

Listing 10. O método `EnablePcapAll` habilita a saída do arquivo .pcap, neste caso é criado um arquivo para cada interface de cada dispositivo. O método `EnableAsciiAll` habilita a saída em formato de texto, neste caso apenas um arquivo é criado.

```

1 //Run scheme
2 Ptr<Packet> pkt = Create<Packet> (pktsize);
3 devs[1]->Send(pkt,devs[0]->GetAddress(),0x86DD);
4 Simulator::Schedule(Seconds(0.01), &SchedulePackets,devs
    [1], devs[0]->GetAddress());

```

Listing 11. Os pacotes são agendados, um pacote é enviado a cada 0.01 segundos, referente a duração do timeslot, dessa forma todo timeslot terá um pacote a ser enviado.

```

1 //Activate TSCH
2 MlmeTschModeRequestParams modeRequest;
3 modeRequest.TSCHMode = MlmeTschMode_ON;
4
5 for (uint8_t i = 0;i<2;i++)
6     Simulator::Schedule(Seconds(0.0), &LrWpanMac::
        MlmeTschModeRequest,devs[i]->GetMac(),
        modeRequest);
7
8 //End TSCH
9 MlmeTschModeRequestParams modeRequestoff;
10 modeRequestoff.TSCHMode = MlmeTschMode_OFF;
11
12 for (uint8_t i = 0;i<2;i++)

```

```
13      Simulator::Schedule(Seconds(duration), &LrWpanMac::  
        MlmeTschModeRequest, devs[i]->GetMac(),  
        modeRequestoff);
```

Listing 12. O TSCH é habilitado no início da simulação e desabilitado no final.

```
1 // Start and finish the simulation  
2 Simulator::Run ();  
3 Simulator::Destroy ();
```

Listing 13. Por fim os métodos de início e fim da simulação são invocados.