SICA

Generated by Doxygen 1.6.3

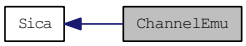
Thu Feb 2 14:37:04 2012

Contents

1 Module Documentation

1.1 ChannelEmu

Collaboration diagram for ChannelEmu:



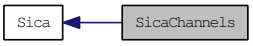
Classes

- class ns3::ChannelEmuHelper

  Helper class that adds channel emulation to channels objects.

1.2 SicaChannels

Collaboration diagram for SicaChannels:



1.3 SicaHelper

Collaboration diagram for SicaHelper:

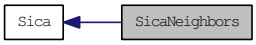
Classes

- class ns3::RoutingHelper

  Helper class that adds Sica channel assignment to nodes.

1.4 SicaNeighbors

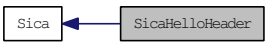
Collaboration diagram for SicaNeighbors:



1.5 SicaHelloHeader

Sica Hello Message defines the header structure of hello messages of Sica. Hello messages used to inform neighboring nodes about channel information and switching attempts.

Collaboration diagram for SicaHelloHeader:



Sica Hello Message defines the header structure of hello messages of Sica. Hello messages used to inform neighboring nodes about channel information and switching attempts. This class defines the hello message format for Sica protocol.

Parameters

{Hello Sequence Number }: Sequence number of message

Parameters

{Originator ID}: Node Id of the originator

Parameters

{Originator Time }:Used to distinguish between old and new hello messages

Parameters

{::R-Radios} : Number of receiving radios

Parameters

{Channel R-R}: current channel of receiving radio

Parameters

{Bx(Channel R-R)}: Estimated consumed bandwidth by external interference over current channel of R-R

Parameters

{R-R -NewChannel}: The next channel for channel switching attempt of R-R, R-R -NewChannel=0 shows no switching

Parameters

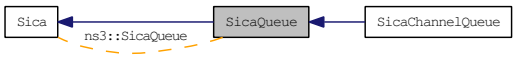
{Time To Switch R interface} : Time in miliseconds until R-R switch

```
0                   1                   2
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Hello Sequence Number
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Originator ID
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Originator Time
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  #Radios    |  Channel R-R  |Bx(Channel R-R)|R-R -Ne
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Receiving Radio #1  MAC Address (part
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Receiving Radio #1  MAC Address (part
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Time To Switch R interface
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Time To Sense the current channel of R
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
| # Neighbors   |Channel R-R N#1|Channel R-R N#2|Chann
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Neighbor #1 ID
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Neighbor #2 ID (if exists)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Neighbor #3 ID (if exists)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     CLCPF (for Urbanx protocol)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     CLCPF (for Urbanx protocol)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     TTL (for Urbanx protocol)
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

1.6 SicaQueue

Collaboration diagram for SicaQueue:



Classes

- class ns3::SicaQueue

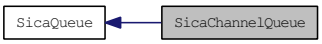
  SicaQueue is used to handle more than one data and signal queue for each node.

Modules

- SicaChannelQueue

1.7 SicaChannelQueue

Collaboration diagram for SicaChannelQueue:



1.8 RTABLE

Routing table structure and functions here we need to manage the routing information using a global routing protocol.

Collaboration diagram for RTABLE:

Classes

- class ns3::RoutingHelper

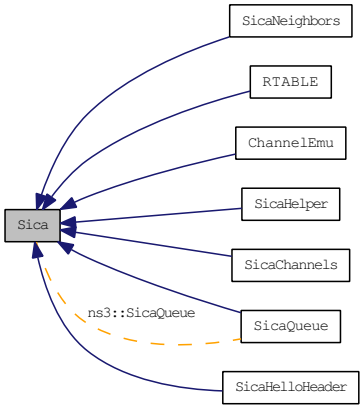
  Helper class that adds Sica channel assignment to nodes.

1.8.1 Detailed Description

Routing table structure and functions here we need to manage the routing information using a global routing protocol.

1.9 Sica

Collaboration diagram for Sica:



Classes

- class ns3::SicaQueue

  SicaQueue is used to handle more than one data and signal queue for each node.

Modules

- ChannelEmu
- SicaChannels
- SicaHelper
- SicaNeighbors
- SicaHelloHeader

  Sica Hello Message defines the header structure of hello messages of Sica. Hello messages used to inform neighboring nodes about channel information and switching attempts.

- SicaQueue
- RTABLE

  Routing table structure and functions here we need to manage the routing information using a global routing protocol.

2 Class Documentation

2.1 ns3::ChannelEmu Class Reference

A channel emulation which emulate the external interference over channels using some random variables, this object would be aggregated to each channel.

`#include <channel-emulation.h>`

Public Types

- enum Status { Idle_State = 1, Busy_State = 2 }
  used to differentiate packets for services.

Public Member Functions

- ChannelEmu ()
  c-tor
- virtual ~ChannelEmu ()
  d-tor
- void SetChannelNumber (uint32_t chId)
  set the channel number for which this emulator works
- uint32_t GetChannelNumber ()
  Return the channel number for which this emulator works.
- void SetBusyDuration (Time duration)
  set the duration for channel busy status
- bool IsBusy ()
  return true if the current status is busy otherwise false
- bool IsIdle ()
  return true if the current status is idle otherwise false
- void ChangeStatus ()
  Changes the current status and set the timer for next time.
- void NotifyStatusChanged ()
  Public method used to fire a trace for a status changes.

Static Public Member Functions

- static TypeId GetTypeId (void)
  Makes it possible for user to change emulation parameters through calling SetAttribute.

2.1.1 Detailed Description

A channel emulation which emulate the external interference over channels using some random variables, this object would be aggregated to each channel.

2.1.2 Member Enumeration Documentation

2.1.2.1 enum ns3::ChannelEmu::Status

used to differentiate packets for services.

Enumerator:

Idle_State Channel is idle.
Busy_State Channel is Busy with external nodes.

The documentation for this class was generated from the following file:

- src/mrmc/sica/channel-emulation.h

2.2 ns3::ChannelEmuContainer Class Reference

Container which holds channel emulators.

`#include <channel-emulation.h>`

Public Types

- typedef std::vector< Ptr< ChannelEmu > >::const_iterator Iterator

  Iterator for vector of channel emulators.

Public Member Functions

- ChannelEmuContainer ()
- Iterator Begin (void) const

  the pointer to the first element in the container

- Iterator End (void) const

  the pointer to the last element of the container

- uint32_t GetN (void) const

  Return the number of elements in the container.

- Ptr< ChannelEmu > Get (uint32_t i) const

  Return one element in the container.

- Ptr< ChannelEmu > GetId (uint32_t chId) const

  Return one of the elements in the container according to the channel id.

- void Add (Ptr< ChannelEmu > c)
  Add one element to the container.

2.2.1 Detailed Description

Container which holds channel emulators.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 ns3::ChannelEmuContainer::ChannelEmuContainer ()

Create an empty ChannelEmuContainer

2.2.3 Member Function Documentation

2.2.3.1 void ns3::ChannelEmuContainer::Add (Ptr< ChannelEmu > *c*)

Add one element to the container.

Parameters

c The pointer to the channel emulator object

2.2.3.2 Ptr<ChannelEmu> ns3::ChannelEmuContainer::Get (uint32_t *i*) const

Return one element in the container.

Parameters

i the place of the element to be returned

2.2.3.3 Ptr<ChannelEmu> ns3::ChannelEmuContainer::GetId (uint32_t *chId*) const

Return one of the elements in the container according to the channel id.

Parameters

chId the channel id corresponding to the channel emulator object

The documentation for this class was generated from the following file:

- src/mrmc/sica/channel-emulation.h

2.3 ns3::ChannelEmuHelper Class Reference

Helper class that adds channel emulation to channels objects.

`#include <channel-emulation.h>`

Public Member Functions

- Ptr< ChannelEmu > Create (uint32_t chId) const
- void Set (std::string name, const AttributeValue &value)
- ChannelEmuContainer Install (std::vector< uint32_t > channels) const

2.3.1 Detailed Description

Helper class that adds channel emulation to channels objects.

2.3.2 Member Function Documentation

2.3.2.1 Ptr<ChannelEmu> ns3::ChannelEmuHelper::Create (uint32_t *chId*) const

Parameters

chId the wifi channel on which emulation will run

Returns

a newly-created channel emulator

2.3.2.2 ChannelEmuContainer ns3::ChannelEmuHelper::Install (std::vector< uint32_t > *channels*) const

For each channel in the input container,implements ns3::ChannelEmu

Parameters

channels that holds the set of channels for which to install the new agent.

2.3.2.3 void ns3::ChannelEmuHelper::Set (std::string *name*, const AttributeValue & *value*)

Parameters

name the name of the attribute to set

Parameters

value the value of the attribute to set.

This method controls the attributes of ns3::ChannelEmu::ChannelEmu

The documentation for this class was generated from the following file:

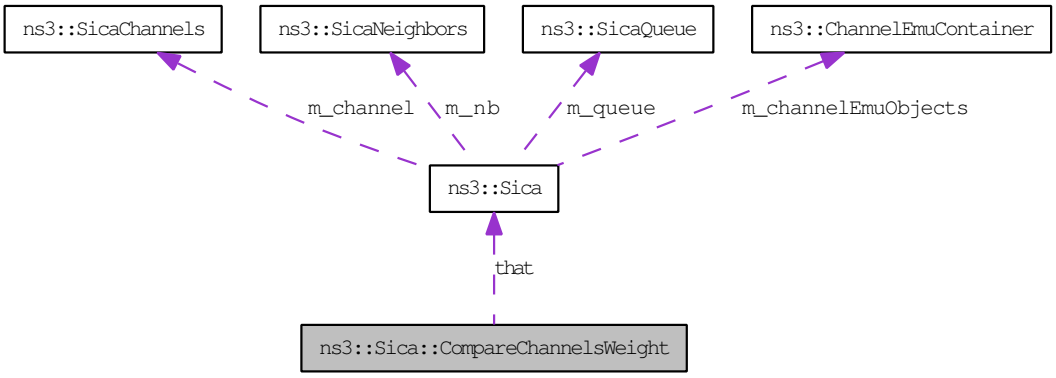
- src/mrmc/sica/channel-emulation.h

2.4 ns3::Sica::CompareChannelsWeight Class Reference

Compare two channels based on the weights assigned to them, used to sort the list of channels based on their weight.

`#include <sica.h>`

Collaboration diagram for ns3::Sica::CompareChannelsWeight:



Public Member Functions

- CompareChannelsWeight (Sica &s)

  c-tor

- bool operator() (uint32_t i, uint32_t j) const

  Get two channels id and compare them based on the weighs assigned to each channel.

2.4.1 Detailed Description

Compare two channels based on the weights assigned to them, used to sort the list of channels based on their weight.

2.4.2 Member Function Documentation

2.4.2.1 bool ns3::Sica::CompareChannelsWeight::operator() (uint32_t *i*, uint32_t *j*) const [inline]

Get two channels id and compare them based on the weighs assigned to each channel.

Parameters

i channel ID

Parameters

j channel ID

Returns

true if the weight of first channel is less than the weight of the second channel. the channel with the bigger weight is more suitable for data transmission

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica.h

2.5 ns3::RoutingHelper Class Reference

Helper class that adds Sica channel assignment to nodes.

`#include <sica-rtable.h>`

Public Member Functions

- Ptr< RTable > Create (Ptr< Node > node, const char *fileName) const
- void Set (std::string name, const AttributeValue &value)

  This method controls the attributes of ns3::Sica::Sica.

- void Install (NodeContainer c, const char *fileName)

  For each node in the input container,implements ns3::Rtable The program will assert if this method is called on a container with a node that already has a Rtable object aggregated to it.

2.5.1 Detailed Description

Helper class that adds Sica channel assignment to nodes. A routing helper which reads a routing file and attaches a routing table to a node

2.5.2 Member Function Documentation

2.5.2.1 Ptr<RTable> ns3::RoutingHelper::Create (Ptr< Node > *node*, const char * *fileName*) const

Parameters

node the node on which this routing will run

Parameters

fileName the file which contains routing information

Returns

a newly-created routing table

This method will be called by ns3::RoutingHelper::Install

2.5.2.2 void ns3::RoutingHelper::Install (NodeContainer *c*, const char * *fileName*)

For each node in the input container,implements ns3::Rtable The program will assert if this method is called on a container with a node that already has a Rtable object aggregated to it.

Parameters

c NodeContainer that holds the set of nodes on which to install the new agent.

Parameters

fileName the name of the input file which contain the routing information (SrcId DestId NextHopId Metric)

2.5.2.3 void ns3::RoutingHelper::Set (std::string *name*, const AttributeValue & *value*)

This method controls the attributes of ns3::Sica::Sica.

Parameters

name the name of the attribute to set

Parameters

value the value of the attribute to set.

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-rtable.h

2.6 ns3::RTable Class Reference

A routing table which contains a list of routes for static routing in Sica.

`#include <sica-rtable.h>`

Public Member Functions

- RTable ()

  c-tor

- ~RTable ()

  d-tor

- bool AddRouteToTable (SicaRoutingTableEntry *route)

  Add route to the list.

- void MakeRoute (uint32_t srcId, uint32_t dstId, uint32_t nextHopId, double metric)

  Make a route to a node with destId from nexthop node and add it to the list.

- SicaRoutingTableEntry * FindRoute (uint32_t srcId, uint32_t dstId)
  Find the route entry for the destination node with dstId.
- int FindNextHop (uint32_t srcId, uint32_t dstId)
  Find the id of the nexthop node to the destination node with dstId.
- void ReadRoutesFromFile (const char *fileName)
  Fill the routing tables from file.
- void PrintRTable (std::ostream &os)
  Print all content of the routing table.

Static Public Member Functions

- static TypeId GetTypeId (void)
  Used to set the parameters of the class.

2.6.1 Detailed Description

A routing table which contains a list of routes for static routing in Sica.

2.6.2 Member Function Documentation

2.6.2.1 bool ns3::RTable::AddRouteToTable (SicaRoutingTableEntry * *route*)

Add route to the list.

Parameters

route the pointer to the routing table entry

2.6.2.2 int ns3::RTable::FindNextHop (uint32_t *srcId*, uint32_t *dstId*)

Find the id of the nexthop node to the destination node with dstId.

Parameters

srcId the Id of the source node

Parameters

dstId the Id of the destination node

2.6.2.3 SicaRoutingTableEntry* ns3::RTable::FindRoute (uint32_t *srcId*, uint32_t *dstId*)

Find the route entry for the destination node with dstId.

Parameters

srcId the Id of the source node
Parameters

dstId the Id of the destination node

2.6.2.4 void ns3::RTable::MakeRoute (uint32_t *srcId*, uint32_t *dstId*, uint32_t *nextHopId*, double *metric*)

Make a route to a node with destId from nexthop node and add it to the list.

Parameters

srcId the Id of the source node
Parameters

dstId the Id of the destination node
Parameters

nextHopId the Id of the next hop node on the path to the destination
Parameters

metric the metric of the path (reserved for future work if necessary)

2.6.2.5 void ns3::RTable::PrintRTable (std::ostream & *os*)

Print all content of the routing table.

Parameters

os the output stream

2.6.2.6 void ns3::RTable::ReadRoutesFromFile (const char * *fileName*)

Fill the routing tables from file.

Parameters

fileName the name of the input file contains static routes with the following format (srcId dstId nextHop Id metric)

The documentation for this class was generated from the following file:

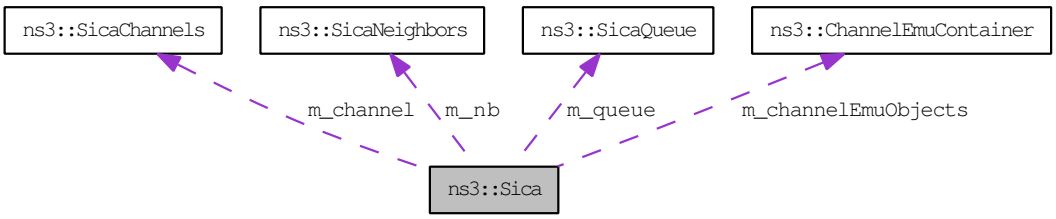
- src/mrmc/sica/sica-rtable.h

2.7 ns3::Sica Class Reference

A channel assignment mechanism for multi-channel multi-radio mesh network semi-dynamic interference avoidance channel assignment It select a channel for the receiving radio of each node and switch the transmitter radio to send data to neighbors. hello messages are used to coordinate neighbors. For each node we do the following: 1- Add one data queue for each channel 2- Add one hello-queue for each channel Push every Hello to hello-queue of the channels we have neighbors on it Push every received data to the data-queue of the corresponding channel Each time we got a packet the module check the header and try to find the responsible neighbor for that, then it would check the channel of the neighbor and push the packet to the corresponding channel. during the sending period the module check all channels and send packets corresponding to each channel.

`#include <sica.h>`

Collaboration diagram for ns3::Sica:



Classes

- class CompareChannelsWeight

  Compare two channels based on the weights assigned to them, used to sort the list of channels based on their weight.

Public Member Functions

- Sica ()

  c-tor

- virtual ~Sica ()

  d-tor

- void NotifyRxReceived (Ptr< Packet > packet)

  Public method used to fire a data-received trace for a data packet being received.

- void NotifyTxSent (Ptr< Packet > packet)

  Public method used to fire a data-sent trace for a data packet being sent.

- void NotifyHelloSent (Ptr< Packet > packet)

  Public method used to fire a hello-sent trace for a data packet being sent.

- void NotifyRxDropped (Ptr< const Packet > packet, double snr)

  Public method used to fire a data-dropped trace for a data packet during the receiving process.

- void NotifyChannelProbability (std::vector< double > channelProb)

  Public method used to fire a probability changes trace for channels.

- void Initialize ()

  Initialize Sica, call Sica::InitializeQueues, Sica::InitializeInterfaces, Sica::InitializeChannel and Sica::InitializeTimers.

- void InitializeQueues ()

  Create and initialize channel queues.

- void InitializeInterfaces ()

  Check the number of interfaces over a node and select one transmitter and one receiver interface.

- void InitializeChannel ()

  set the receiving channel for the node and decide a new channel for the initial phase.

- void InitializeTimers ()

  Set the timers for hello interval, sensing period and channel assignment.

- void SetChannelsEmulationObject (ChannelEmuContainer channelsEmu)

  Set the channel emulators for all channels.

- Ptr< ChannelEmu > GetChannelEmu (uint32_t id)

  Return the channel emulators for the given channel id.

- uint32_t GetRChannel ()

  return the receiving channel of the R interface

- uint32_t GetTChannel ()

  return the transmitting channel of the T interface

- uint32_t GetRNewChannel ()

  return the future receiving channel to which the R interface will switch after a period of time

- uint32_t GetId ()

  return the Id of the attaching node to the Sica object

- SicaNeighbors * GetSicaNeighbors ()

  Return the address of Sica::m_nb.

- SicaQueue * GetSicaQueue ()

  Return the address of Sica::m_queue.

- SicaChannels * GetSicaChannel ()

  Return the address of Sica::m_channel.

- void PrintNeighborTable (std::ostream &os)

  It calls the same function in neighbor class to print neighbor table of node.

- void PrintChannelTable (std::ostream &os)

It calls the same function in channels class to print the channel information table.

- bool ChannelIsBusy (uint32_t chId)

  Check whether the channel associated to the net device is busy or not.

- void ModifySenseChannelFlag (uint32_t c)

  Set the Sense channel flag to prevent other functions from sending data during the sensing period.

- void StartSenseCurrentChannel ()

  Senses the current channel of receiving interface and estimate the external bandwidth.

- void SenseCurrentChannel ()

  Start checking the channel situation every Sica::ChannelSensePeriod and count the number of times that a channel is busy, is called by Sica::StartSenseCurrentChannel.

- void EndSenseCurrentChannel ()

  Estimate the channel busy time at the end of the sensing period and write it in the channel table is called by Sica::StartSenseCurrentChannel.

- void ReScheduleTimer (Timer *t, Time minDelay)

  Re-schedule timer with minDelay if minDelay is less than the delay left for timer t.

Sica Data Delivery methods.

- bool RecvPacket (Ptr< NetDevice > dstDevice, Ptr< const Packet > packet, uint16_t protocolNumber, const Address &srcAddr)

  Listen to device and receives packets.

- void ProcessRcvHello (Ptr< Packet > p, Address srcAddr)

  Gathers information from hello and call Sica::UpdateNeighborTable and Sica::UpdateChannelTable .

- void ProcessRcvData (Ptr< Packet > p)

  Evaluate the received data packet, send it to Sica::NotifyRxReceived if the node is the destination of the packet or call Sica:: DistributeDataPacket to send it to the receiving channel of the next hop node.

- bool UpdateNeighborTable (SicaHelloHeader sicaHelloHeader, Address srcAddr)

  Update neighbor information based on information in hello message.

- bool UpdateChannelTable (SicaHelloHeader sicaHelloHeader)

  Update channel information based on information in hello message.

- void CreateHello ()

  Produces hello send it to all neighbor channel for sending, call Sica::CreateHelloHeader for producing hello header.

- SicaHelloHeader CreateHelloHeader ()

  Produces hello header information based on neighbor table and current information and return it.

- void DistributeHello (Ptr< Packet > p)

distribute hello message to all channels over which the node has a neighbor

- void CreateData (uint32_t dstId, uint32_t pSize)

  Produces uni-cast data packet with given packet size and destination address and push it to neighbor channel.

- int AddDataHeaders (Ptr< Packet > p, uint32_t srcId, uint32_t dstId, Time originTime)

  Create a sica header and add it to the packet this function may be called from a source node or in any of the forwarding nodes on the path to the destination.

- void DistributeDataPacket (Ptr< Packet > p, uint32_t nextHopChannel)

  Distribute the data message to the appropriate channel.

- void SendPacket (Ptr< Packet > packet, Ptr< NetDevice > device, uint32_t protocolNumber)

  Get packet and the device through which the node wants to use to send data and set the sending parameters.

- Time EstimateTxDuration (uint32_t pSize, Ptr< WifiPhy > wifiphy)

  Estimate the transition duration for physical layer.

- void DeviceSend (Ptr< NetDevice > device, Ptr< Packet > packet, Address dstAddr, uint32_t protocolNumber)

  Get packet and device address to send packet through it.

Sica Channel switching methods.

- void HandleNeighborSwitchChannel ()

  It handles the process necessary when a node discover that its neighbor has switched its channel.

- void ScheduleSwitchRInterface ()

  Check whether the interface is busy, if so predict the idle time then add it to a random switch delay and schedule a timer for switching the R interface.

- void SwitchRInterface ()

  Switch the r interface to new channel (Sica::m_rNewChannel).

- bool SwitchTInterface (uint32_t c)

  Switch the T interface to a channel for sending data.

- void TInterfaceStartSend (uint32_t ch)

  Switch T interface to channels one by one and send data.

- void TInterfaceSend (uint32_t ch)

  Send data over the given channels (Sica::channelsToPoll)using the T interface.

- bool TInterfaceReadyToSend (uint32_t ch, Time txEstimation)

  Check whether all conditions are held before start sending with T-Interface over the given channel.

- void RInterfaceCheckChannelQueue ()

  Check whether there is any packet for send in receiving channel and send it if the R interface is ready to send, call Sica::RInterfaceReadyToSend.

- bool RInterfaceReadyToSend (Time txEstimation)

  Check whether it is possible to send data over R interface or not, check the sense flag and remaining time to the upcoming switch.

Sica Channel Decision making mechanisms

- void GameChannelAssignment ()

  Select a channel for R interface.

- void UpdateLossMatrix ()

  Update the loss matrix based on different formulas which is indicated by Sica::LossFormulaNum.

- double ComputeStageLoss (uint32_t c)

  Compute the loss for the given channel based on its weight.

- void UpdateChannelWeight ()

  Compute the new weight for all available channels based on their previous weight and the given loss vector.

- uint32_t SelectRandomChannel ()

  select a random channel based on the probability assigned to each channel

- std::vector< double > ComputeCumulativeProbability (std::vector< double > p)

  compute the cumulative probability of channels

- void PrintGameData (std::ostream &os, std::vector< uint32_t > channels, std::vector< double > prob, std::vector< double > weight, std::vector< double > cProb)

  Print game information, it is called by Sica::GameChannelAssignment this function must be called by Sica::GameChannelAssignment.

Static Public Member Functions

- static TypeId GetTypeId (void)

  Makes it possible for user to change protocol parameters through calling SetAttribute.

Public Attributes

Sica Parameters

- uint32_t **LossFormulaNum**
- uint32_t SICA_DATA_PORT

  protocol id used to send sica data packets (Default=550)

- uint32_t SICA_HELLO_PORT

  protocol id used to send sica broadcast packets (Default=551)

- uint32_t Max_CH

Maximum Number of available channels (Default=8).

- uint32_t Min_CH

  The first Index of the available channels (Default=1).

- uint32_t Max_BW

  Maximum available bandwidth of each channel.

- double m_gamma

  Gamma parameter for channel decision.

- double m_beta

  Beta parameter for calculating the channel weights in decision game.

- double m_alpha
- Time HelloInterval

  Hello message interval.

- Time DataExpireTime

  The maximum period of time that Sica is allowed to buffer a data packet for 2000 seconds.

- Time HelloExpireTime

  The maximum period of time that Sica is allowed to buffer a hello packet for 10 seconds.

- Time NeighborExpireTime

  The maximum period of time that Sica keep neighbor information if it does not receive any new information from neighbor. We set it to at least three times of the HelloInterval.

- Time SwitchingDelay

  Switching delay for interface, we set it to a fixed number for simplicity, it is equal to the switching delay in default 802.11b radio interface 250 microseconds.

- Time ChannelAssignmentInterval

  The period between channel assignment runs.

- Time ChannelSenseInterval

  Sensing channel interval.

- Time ChannelSensePeriod

  The duration of sensing receiving channel.

- Time ChannelSenseRate

  Sica sense channel using sampling, this time control the sampling rate.

- Time BxExpireTime

  The maximum period of time that Sica keeps estimated external bandwidth consumption for a channel entry in channel list. If it does not receive any updated information it will reset external interference estimation to zero after this time, We set it to 40s.

- Time ChannelBusyBackoffTime

The delay that Sica differ sending over a channel if it found it busy.

- Time QueuePollTime

  The interval that Sica waits before check an empty queue (used only when all queues are empty or for R interface to check the corresponding queue of the receiving channel).

2.7.1 Detailed Description

A channel assignment mechanism for multi-channel multi-radio mesh network semi-dynamic interference avoidance channel assignment It select a channel for the receiving radio of each node and switch the transmitter radio to send data to neighbors. hello messages are used to coordinate neighbors. For each node we do the following: 1- Add one data queue for each channel 2- Add one hello-queue for each channel Push every Hello to hello-queue of the channels we have neighbors on it Push every received data to the data-queue of the corresponding channel Each time we got a packet the module check the header and try to find the responsible neighbor for that, then it would check the channel of the neighbor and push the packet to the corresponding channel. during the sending period the module check all channels and send packets corresponding to each channel.

2.7.2 Member Function Documentation

2.7.2.1 int ns3::Sica::AddDataHeaders (Ptr< Packet > *p*, uint32_t *srcId*, uint32_t *dstId*, Time *originTime*)

Create a sica header and add it to the packet this function may be called from a source node or in any of the forwarding nodes on the path to the destination.

Parameters

p The packet to which header must be added

Parameters

dstId the id of the destination node

Parameters

srcId the id of the source node

Parameters

originTime the time of origination packet

2.7.2.2 std::vector<double> ns3::Sica::ComputeCumulativeProbability (std::vector< double > *p*)

compute the cumulative probability of channels

Parameters

p the probability vector assigned to all channels

Returns

The cumulative probability

2.7.2.3 double ns3::Sica::ComputeStageLoss (uint32_t *c*)

Compute the loss for the given channel based on its weight.

Parameters

c the id of a channel for which the loss will be computed

2.7.2.4 void ns3::Sica::CreateData (uint32_t *dstId*, uint32_t *pSize*)

Produces uni-cast data packet with given packet size and destination address and push it to neighbor channel.

Parameters

dstId the id of the destination node

Parameters

pSize the size of the data packet

2.7.2.5 void ns3::Sica::DistributeDataPacket (Ptr< Packet > *p*, uint32_t *nextHopChannel*)

Distribute the data message to the appropriate channel.

Parameters

p The packet produced by create Sica::CreateData

Parameters

nextHopChannel The id of the channel where the data must be sent

2.7.2.6 void ns3::Sica::DistributeHello (Ptr< Packet > *p*)

distribute hello message to all channels over which the node has a neighbor

Parameters

p hello packet

2.7.2.7 Time ns3::Sica::EstimateTxDuration (uint32_t *pSize*, Ptr< WifiPhy > *wifiphy*)

Estimate the transition duration for physical layer.

Parameters

pSize The size of packet for sending

Parameters

wifiphy the pointer to the physical layer

2.7.2.8 Ptr<ChannelEmu> ns3::Sica::GetChannelEmu (uint32_t *id*) [inline]

Return the channel emulators for the given channel id.

Parameters

id the channel id

Returns

the channel emulation object related to the given channel id

2.7.2.9 SicaChannels* ns3::Sica::GetSicaChannel ()

Return the address of Sica::m_channel.

Returns

Sica::m_channel

2.7.2.10 SicaNeighbors* ns3::Sica::GetSicaNeighbors ()

Return the address of Sica::m_nb.

Returns

Sica::m_nb

2.7.2.11 SicaQueue* ns3::Sica::GetSicaQueue ()

Return the address of Sica::m_queue.

Returns

Sica::m_queue

2.7.2.12 void ns3::Sica::NotifyChannelProbability (std::vector< double > *channelProb*)

Public method used to fire a probability changes trace for channels.

Parameters

channelProb a vector containing the channel probability

2.7.2.13 void ns3::Sica::NotifyHelloSent (Ptr< Packet > *packet*)

Public method used to fire a hello-sent trace for a data packet being sent.

Parameters

packet a copy of a packet which was sent

2.7.2.14 void ns3::Sica::NotifyRxDropped (Ptr< const Packet > *packet*, double *snr*)

Public method used to fire a data-dropped trace for a data packet during the receiving process.

Parameters

packet a pointer to the packet

Parameters

snr The signal to noise ratio detected on channel

2.7.2.15 void ns3::Sica::NotifyRxReceived (Ptr< Packet > *packet*)

Public method used to fire a data-received trace for a data packet being received.

Parameters

packet Received packet

2.7.2.16 void ns3::Sica::NotifyTxSent (Ptr< Packet > *packet*)

Public method used to fire a data-sent trace for a data packet being sent.

Parameters

packet a copy of a packet which was sent

2.7.2.17 void ns3::Sica::PrintGameData (std::ostream & *os*, std::vector< uint32_t > *channels*, std::vector< double > *prob*, std::vector< double > *weight*, std::vector< double > *cProb*)

Print game information, it is called by Sica::GameChannelAssignment this function must be called by Sica::GameChannelAssignment.

Parameters

os the output media

Parameters

channels the id of channels

Parameters

prob the probability vector assigned to channels

Parameters

weight the weight assigned to channels

Parameters

cProb the cumulative probability

2.7.2.18 void ns3::Sica::ProcessRcvData (Ptr< Packet > *p*)

Evaluate the received data packet, send it to Sica::NotifyRxReceived if the node is the destination of the packet or call Sica:: DistributeDataPacket to send it to the receiving channel of the next hop node.

Parameters

p received packet

2.7.2.19 bool ns3::Sica::RecvPacket (Ptr< NetDevice > *dstDevice*, Ptr< const Packet > *packet*, uint16_t *protocolNumber*, const Address & *srcAddr*)

Listen to device and receives packets.

Parameters

dstDevice the address of the device which receives a packet

Parameters

packet received packet

Parameters

protocolNumber the address of receiving protocol here is SICA_PORT

Parameters

srcAddr the address of sender device

2.7.2.20 void ns3::Sica::ReScheduleTimer (Timer * *t*, Time *minDelay*)

Re-schedule timer with minDelay if minDelay is less than the delay left for timer t.

Parameters

t the pointer to the timer

Parameters

minDelay the time to which the timer will be rescheduled if it is less than the remaining time to the end of the timer

2.7.2.21 bool ns3::Sica::RInterfaceReadyToSend (Time *txEstimation*)

Check whether it is possible to send data over R interface or not, check the sense flag and remaining time to the upcoming switch.

Parameters

txEstimation the estimation time necessary for sending data or hello packets

Returns

true if the R interface is ready to send, false otherwise

2.7.2.22 uint32_t ns3::Sica::SelectRandomChannel ()

select a random channel based on the probability assigned to each channel

Returns

the random channel

2.7.2.23 void ns3::Sica::SendPacket (Ptr< Packet > *packet*, Ptr< NetDevice > *device*, uint32_t *protocolNumber*)

Get packet and the device through which the node wants to use to send data and set the sending parameters.

Parameters

packet data to send

Parameters

device the interface through which the data would be sent

Parameters

protocolNumber used to distinguish hello and data packets

2.7.2.24 void ns3::Sica::SetChannelsEmulationObject (ChannelEmuContainer *channelsEmu*) [inline]

Set the channel emulators for all channels.

Parameters

channelsEmu the channels emulation container which contains emulation objects to be added to the Sica object

2.7.2.25 bool ns3::Sica::SwitchTInterface (uint32_t *c*)

Switch the T interface to a channel for sending data.

Parameters

c The channel to which the T interface will switch

Returns

true if the switching is successful, false otherwise

2.7.2.26 bool ns3::Sica::TInterfaceReadyToSend (uint32_t *ch*, Time *txEstimation*)

Check whether all conditions are held before start sending with T-Interface over the given channel.

Parameters

ch The id of the channel over which the T interface will send data

Parameters

txEstimation The time estimation for sending one packet

2.7.2.27 void ns3::Sica::TInterfaceSend (uint32_t *ch*)

Send data over the given channels (Sica::channelsToPoll)using the T interface.

Parameters

ch The channel for which the T interface start the transmission

2.7.2.28 void ns3::Sica::TInterfaceStartSend (uint32_t *ch*)

Switch T interface to channels one by one and send data.

Parameters

ch the channel to which the T interface will switch to send data Sica::TMax maximum amount of time that node will stay on one channel to send data

2.7.2.29 bool ns3::Sica::UpdateChannelTable (SicaHelloHeader *sicaHelloHeader*)

Update channel information based on information in hello message.

Parameters

sicaHelloHeader Hello header which is passed to process

2.7.2.30 bool ns3::Sica::UpdateNeighborTable (SicaHelloHeader *sicaHelloHeader*, Address *srcAddr*)

Update neighbor information based on information in hello message.

Parameters

sicaHelloHeader Hello header which is passed to process

Parameters

srcAddr The header of the transmitter from which the hello packet was received

Returns

true if the information is not stale

2.7.3 Member Data Documentation

2.7.3.1 double ns3::Sica::m_alpha

Alpha parameter for calculating the loss in decision game

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica.h

2.8 ns3::SicaChannels::SicaChannel Struct Reference

Structure which holds channel information for Sica or Urbanx protocols.

`#include <sica-channel.h>`

Public Member Functions

- SicaChannel (uint32_t chId, uint32_t bw, uint32_t bx, uint32_t niNo, Time bxExpTime)
  c-tor of the SicaChannel struct

Public Attributes

- uint32_t m_cId
  Channel ID.
- uint32_t m_bw
  Channel's total bandwidth in mbps.
- uint32_t m_bx
  Estimated external interference bandwidth consumption for this channel.
- uint32_t m_neighborsNo
  Number of neighboring interface on the channel.
- double m_weight
  Channel weights according to decision game.
- Time m_bxExpireTime
  Reset Bx after a long period of time if there is no update happens.
- bool m_senseFalg

2.8.1 Detailed Description

Structure which holds channel information for Sica or Urbanx protocols.

2.8.2 Member Data Documentation

2.8.2.1 bool ns3::SicaChannels::SicaChannel::m_senseFalg

Used to show whether this channel is being sensed or not, during the sense period no data transmission is done

The documentation for this struct was generated from the following file:

- src/mrmc/sica/sica-channel.h

2.9 ns3::SicaQueue::SicaChannelQueue Struct Reference

Sica Channel Queue is a structure which stores data and signal queue for one channel.

`#include <sica-queue.h>`

Public Member Functions

- SicaChannelQueue (uint32_t ch)
  c-tor
- virtual ~SicaChannelQueue (void)
  d-tor

- void SetChannelQueue (uint32_t ch)

  Set channel number.

- void OpenChannelQueue ()

  Open a recently closed queue.

- bool IsExpired (SicaQueueEntry ent)

  Remove expired entries in the queue.

Public Attributes

- std::vector< SicaQueueEntry > m_helloQueue

  Store Hello packets targeted to one channel in a vector.

- std::vector< SicaQueueEntry > m_dataQueue

  Store Data packets targeted to one channel in a vector.

- uint32_t m_ch

  Channel number.

- bool m_close

  If this queue is active or not.

2.9.1 Detailed Description

Sica Channel Queue is a structure which stores data and signal queue for one channel.

The documentation for this struct was generated from the following file:

- src/mrmc/sica/sica-queue.h

2.10 ns3::SicaChannels Class Reference

Sica ChannelTable defines the table structure for saving information about channels in Sica.

`#include <sica-channel.h>`

Classes

- struct SicaChannel

  Structure which holds channel information for Sica or Urbanx protocols.

Public Member Functions

- SicaChannels ()

  c-tor

- virtual ~SicaChannels ()

  d-tor

- uint32_t GetChannelID (SicaChannel *ch)

  Return the Id of the channel table object.

- SicaChannel * FindChannel (uint32_t chId)

  Return the pointer to the channel information with ID id.

- void UpdateChannel (uint32_t chId, uint32_t bw, uint32_t bx, uint32_t niNo, Time bxExpTime)

  Update or insert information in channel list.

- void SetChannelBandwidth (uint32_t chId, uint32_t bw)

  Set the total bandwidth of the channel with ID id.

- uint32_t GetChannelBandwidth (uint32_t chId)

  Return the total bandwidth of the channel with ID id.

- void SetChannelExtBandwidth (uint32_t chId, uint32_t bx, Time bxExpTime)

  Set the external bandwidth consumption for channel with ID id.

- uint32_t GetChannelExtBandwidth (uint32_t chId)

  Return the external bandwidth consumption for channel with ID id.

- Time GetExtBandwidthExpireTime (uint32_t chId)

  Return the expiration time for estimated external bandwidth for channel with ID id.

- void SetChannelNeighbors (uint32_t chId, uint32_t niNo)

  Set the number of neighbors that have one receiving radio on channel with ID id.

- uint32_t GetChannelNeighbors (uint32_t chId)

  Return the number of neighbors that have one receiving radio on channel with ID id.

- void IncChannelNeighbors (uint32_t chId)

  Increase by one the number of neighbors that have one receiving radio on channel with ID id.

- void DecChannelNeighbors (uint32_t chId)

  Decrease by one the number of neighbors that have one receiving radio on channel with ID id.

- void SetChannelWeight (uint32_t chId, double w)

  Set the weight of a channel with ID id.

- double GetChannelWeight (uint32_t chId)

  Return the weight of a channel with ID id.

- void SetChannelSenseFlag (uint32_t chId)
- void ResetChannelSenseFlag (uint32_t chId)

  Reset the sense flag for the channel.

- bool IsBeingSensed (uint32_t chId)

  Return true if any neighbor is sensing this channel, otherwise false.

- void PrintChannel (std::ostream &os)

  Print Channel Table.

- void Clear ()

  Clear channel list.

- double CalculateCLCPF (uint32_t ccc)

  calculate and return clcpf for Urbanx::Urbanx protocol

- int FindMaxWeightChannel (uint32_t ccc)

  find channel with the maximum weight

2.10.1 Detailed Description

Sica ChannelTable defines the table structure for saving information about channels in Sica.

2.10.2 Member Function Documentation

2.10.2.1 double ns3::SicaChannels::CalculateCLCPF (uint32_t *ccc*)

calculate and return clcpf for Urbanx::Urbanx protocol

Returns

channel with minimum weight

Parameters

ccc the Id of the common channel to be avoided from the calculation

2.10.2.2 void ns3::SicaChannels::DecChannelNeighbors (uint32_t *chId*)

Decrease by one the number of neighbors that have one receiving radio on channel with ID id.

Parameters

chId the Id of the channel

2.10.2.3 SicaChannel* ns3::SicaChannels::FindChannel (uint32_t *chId*)

Return the pointer to the channel information with ID id.

Parameters

chId the id of the channel

2.10.2.4 int ns3::SicaChannels::FindMaxWeightChannel (uint32_t *ccc*)

find channel with the maximum weight

Parameters

ccc the Id of the common channel to be avoided from the calculation

2.10.2.5 uint32_t ns3::SicaChannels::GetChannelBandwidth (uint32_t *chId*)

Return the total bandwidth of the channel with ID id.

Parameters

chId the Id of the channel

2.10.2.6 uint32_t ns3::SicaChannels::GetChannelExtBandwidth (uint32_t *chId*)

Return the external bandwidth consumption for channel with ID id.

Parameters

chId the Id of the channel

2.10.2.7 uint32_t ns3::SicaChannels::GetChannelID (SicaChannel * *ch*) `[inline]`

Return the Id of the channel table object.

Parameters

ch the pointer to the channel table object

2.10.2.8 uint32_t ns3::SicaChannels::GetChannelNeighbors (uint32_t *chId*)

Return the number of neighbors that have one receiving radio on channel with ID id.

Parameters

chId the Id of the channel

2.10.2.9 double ns3::SicaChannels::GetChannelWeight (uint32_t *chId*)

Return the weight of a channel with ID id.

Parameters

chId the Id of the channel

2.10.2.10 Time ns3::SicaChannels::GetExtBandwidthExpireTime (uint32_t *chId*)

Return the expiration time for estimated external bandwidth for channel with ID id.

Parameters

chId the Id of the channel

2.10.2.11 void ns3::SicaChannels::IncChannelNeighbors (uint32_t *chId*)

Increase by one the number of neighbors that have one receiving radio on channel with ID id.

Parameters

chId the Id of the channel

2.10.2.12 bool ns3::SicaChannels::IsBeingSensed (uint32_t *chId*)

Return true if any neighbor is sensing this channel, otherwise false.

Parameters

chId the Id of the channel

2.10.2.13 void ns3::SicaChannels::PrintChannel (std::ostream & *os*)

Print Channel Table.

Parameters

os the output stream

2.10.2.14 void ns3::SicaChannels::ResetChannelSenseFlag (uint32_t *chId*)

Reset the sense flag for the channel.

Parameters

chId the Id of the channel

2.10.2.15 void ns3::SicaChannels::SetChannelBandwidth (uint32_t *chId*, uint32_t *bw*)

Set the total bandwidth of the channel with ID id.

Parameters

chId the Id of the channel

Parameters

bw the bandwidth of the channel in Mb

2.10.2.16 void ns3::SicaChannels::SetChannelExtBandwidth (uint32_t *chId*, uint32_t *bx*, Time *bxExpTime*)

Set the external bandwidth consumption for channel with ID id.

Parameters

chId the Id of the channel

Parameters

bx the amount of channel bandwidth consumed by external interference

Parameters

bxExpTime the expiration time of new information

2.10.2.17 void ns3::SicaChannels::SetChannelNeighbors (uint32_t *chId*, uint32_t *niNo*)

Set the number of neighbors that have one receiving radio on channel with ID id.

Parameters

chId the Id of the channel

Parameters

niNo number of neighboring nodes radio interface tuned over the channel

2.10.2.18 void ns3::SicaChannels::SetChannelSenseFlag (uint32_t *chId*)

\ Set the sense flag for the channel

Parameters

chId the Id of the channel

2.10.2.19 void ns3::SicaChannels::SetChannelWeight (uint32_t *chId*, double *w*)

Set the weight of a channel with ID id.

Parameters

chId the Id of the channel

Parameters

w the weight of the channel

2.10.2.20 void ns3::SicaChannels::UpdateChannel (uint32_t *chId*, uint32_t *bw*, uint32_t *bx*, uint32_t *niNo*, Time *bxExpTime*)

Update or insert information in channel list.

Parameters

chId the Id of the channel

Parameters

bw the bandwidth of the channel in Mb

Parameters

bx the amount of channel bandwidth consumed by external interference

Parameters

niNo number of neighboring radio interface tuned to the channel

Parameters

bxExpTime the expire time of the information about channel

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-channel.h

2.11 ns3::SicaContainer Class Reference

Container which holds Sica objects.

`#include <sica-helper.h>`

Public Types

- typedef std::vector< Ptr< Sica > >::const_iterator Iterator

  The iterator over a vector of Sica objects.

Public Member Functions

- SicaContainer ()
- Iterator Begin (void) const

  Return the pointer of first element in the container.

- Iterator End (void) const

  Return the pointer of last element in the container.

- uint32_t GetN (void) const

  Return the number of elements in the container.

- Ptr< Sica > Get (uint32_t i) const

  Return one element in the container.

- void Add (Ptr< Sica > s)

  Add one element to the end of the container.

2.11.1 Detailed Description

Container which holds Sica objects.

2.11.2 Constructor & Destructor Documentation

2.11.2.1 ns3::SicaContainer::SicaContainer ()

Create an empty SicaContainer.

2.11.3 Member Function Documentation

2.11.3.1 void ns3::SicaContainer::Add (Ptr< Sica > *s*)

Add one element to the end of the container.

Parameters

s the pointer of the Sica object

2.11.3.2 Ptr<Sica> ns3::SicaContainer::Get (uint32_t *i*) const

Return one element in the container.

Parameters

i the id of the element to be returned

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-helper.h

2.12 ns3::SicaHeader Class Reference

Header used for static routing.

`#include <sica-packet.h>`

Public Member Functions

- SicaHeader (uint32_t sqNo=0, uint32_t origin=0, uint32_t dest=0, uint32_t nextHop=0, Time originTime=Simulator::Now())

  c-tor

- TypeId GetInstanceTypeId () const

  return the type id

- uint32_t GetSerializedSize () const

  return the serialize size of the header

- void Serialize (Buffer::Iterator i) const

  Serialize the header in to bits.

- uint32_t Deserialize (Buffer::Iterator start)

  Deserialize the header.

- void Print (std::ostream &os) const

  Print the content of the header.

- bool IsValid ()
  Check whether the header has valid data or not.
- void SetSeqNo (uint32_t seqno)
  Set the sequence number of the hello header.
- uint32_t GetSeqNo ()
  Return the sequence number of the hello header.
- void SetOrigin (uint32_t origin)
  Set the originator Id.
- uint32_t GetOrigin ()
  Return the originator Id.
- void SetDest (uint32_t dest)
  Set the destination Id.
- uint32_t GetDest ()
  Return the destination Id.
- void SetNextHop (uint32_t nextHop)
  Set the nexthop Id.
- uint32_t GetNextHop ()
  Return the nexthop Id.
- void SetOriginTime (Time originTime)
  Set the originating time.
- Time GetOriginTime ()
  Return the originating time of packet.

Static Public Member Functions

- static TypeId GetTypeId ()
  Used to set parameters of the class.

2.12.1 Detailed Description

Header used for static routing.

2.12.2 Member Function Documentation

2.12.2.1 void ns3::SicaHeader::Print (std::ostream & *os*) const

Print the content of the header.

Parameters

os the output stream

2.12.2.2 void ns3::SicaHeader::SetDest (uint32_t *dest*) [inline]

Set the destination Id.

Parameters

dest the id of the destination node

2.12.2.3 void ns3::SicaHeader::SetNextHop (uint32_t *nextHop*) [inline]

Set the nexthop Id.

Parameters

nextHop the id of the next hop node

2.12.2.4 void ns3::SicaHeader::SetOrigin (uint32_t *origin*) [inline]

Set the originator Id.

Parameters

origin the id of the source

2.12.2.5 void ns3::SicaHeader::SetOriginTime (Time *originTime*) [inline]

Set the originating time.

Parameters

originTime the time of originating the message

2.12.2.6 void ns3::SicaHeader::SetSeqNo (uint32_t *seqno*) [inline]

Set the sequence number of the hello header.

Parameters

seqno the sequence number

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-packet.h

2.13 ns3::SicaHelloHeader Class Reference

Hello Header for Sica or Urbanx hello messages.

`#include <sica-packet.h>`

Public Member Functions

- SicaHelloHeader (uint32_t sqNo=0, uint32_t origin=0, Time originTime=Simulator::Now(), uint8_t radios=1, uint8_t rCh=1, uint8_t extBw=0, uint8_t rNewCh=0, Address rAddr=Address(), Time rSwitchTime=MilliSeconds(0), Time rSenseTime=MilliSeconds(0))

  c-tor

- bool IsValid ()

  Check whether the header has valid data or not.

- void SetSeqNo (uint32_t seqno)

  Set the sequence number of the hello header.

- uint32_t GetSeqNo ()

  Return the sequence number of the hello header.

- void SetOrigin (uint32_t origin)

  Set the originator IP address.

- uint32_t GetOrigin ()

  Return the originator IP address.

- void SetRadios (uint8_t radios)

  Set the number of available radios.

- uint8_t GetRadios ()

  Return the number of available radios.

- void SetRChannel (uint8_t rCh)

  Set the receiving channel.

- uint8_t GetRChannel ()

  Return the receiving channel.

- void SetExtBw (uint8_t extBw)

  Set the external bandwidth over a channel in Mbps.

- uint8_t GetExtBw ()

  Return the external bandwidth over a channel.

- void SetRNewChannel (uint8_t rNCh)

  Set new channel for receiving interface.

- uint8_t GetRNewChannel ()

  Return New channel for receiving interface.

- void SetRAddress (Address rAddr)

  Set address of the originator receiving radio.

- Address GetRAddress ()

  Return address of the originator receiving radio.

- void SetOriginTime (Time originTime)

  Set the originating time.

- Time GetOriginTime ()

  Return the originating time of hello.

- void SetTimeToSwitch (Time sT)

  Set the switching time for receiving interface.

- Time GetTimeToSwitch ()

  Return the switching time of receiving interface.

- void SetSenseTime (Time ssT)

  Synchronize a timer for sensing R-R channel for neighbors over this channel.

- Time GetSenseTime ()

  Return the timer for sensing R-R channel for neighbors over this channel.

- void SetCLCPF (double clcpf)

  Set the CLCPF.

- double GetCLCPF ()

  return the clcpf

- uint32_t GetTTL ()

  return the TTL

- void SetTTL (uint32_t ttl)

  Set ttl.

- void DecreaseTTL ()
  decrease the ttl by one
- uint8_t GetNiNo () const
  Return the number of neighbors.
- bool AddNiRChannel (uint32_t ni, uint8_t rCh)
  Add neighbor receiving channel information to the message.
- bool RemoveNiRChannel (std::pair< uint32_t, uint8_t > &ni)
  Peek and remove the first element of neighbor receiving channel information from the message and return it in a pair.
- bool DeleteNiRChannel (uint32_t ni)
  Delete neighbor receiving channel information from the message.
- void Clear ()
  Cleare Header.

Header serialization/de-serialization

- TypeId **GetInstanceTypeId** () const
- uint32_t **GetSerializedSize** () const
- void **Serialize** (Buffer::Iterator i) const
- uint32_t **Deserialize** (Buffer::Iterator start)
- void **Print** (std::ostream &os) const
- static TypeId **GetTypeId** ()

2.13.1 Detailed Description

Hello Header for Sica or Urbanx hello messages.

2.13.2 Member Function Documentation

2.13.2.1 bool ns3::SicaHelloHeader::AddNiRChannel (uint32_t *ni*, uint8_t *rCh*)

Add neighbor receiving channel information to the message.

Returns

false if we have already add maximum number of neighbors(255) to the message

Parameters

ni the id of neighbor node

Parameters

rCh the receiving channel of the neighbor

2.13.2.2 bool ns3::SicaHelloHeader::DeleteNiRChannel (uint32_t *ni*)

Delete neighbor receiving channel information from the message.

Returns

false if there is no such information in the message

Parameters

ni the id of neighbor node

2.13.2.3 bool ns3::SicaHelloHeader::RemoveNiRChannel (std::pair< uint32_t, uint8_t > & *ni*)

Peek and remove the first element of neighbor receiving channel information from the message and return it in a pair.

Returns

false if there is no such information in the message

Parameters

ni the id of neighbor node

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-packet.h

2.14 ns3::SicaHelper Class Reference

Helper class that adds Sica channel assignment to nodes.

`#include <sica-helper.h>`

Public Member Functions

- Ptr< Sica > Create (Ptr< Node > node, ChannelEmuContainer channelsEmu) const

  This method will be called by ns3::SicaHelper::Install to create and return one Sica object.

- void Set (std::string name, const AttributeValue &value)

  This method controls the attributes of ns3::Sica::Sica.

- SicaContainer Install (NodeContainer c, ChannelEmuContainer channelsEmu) const

  For each node in the input container,implements ns3::Sica The program will assert if this method is called on a container with a node that already has a Sica object aggregated to it.

2.14.1 Detailed Description

Helper class that adds Sica channel assignment to nodes.

2.14.2 Member Function Documentation

2.14.2.1 Ptr<Sica> ns3::SicaHelper::Create (Ptr< Node > *node*, ChannelEmuContainer *channelsEmu*) const

This method will be called by ns3::SicaHelper::Install to create and return one Sica object.

Parameters

node the node on which Sica will run

Parameters

channelsEmu the channel emulators which are simulated the channel situations for scenario

Returns

a newly-created Sica

2.14.2.2 SicaContainer ns3::SicaHelper::Install (NodeContainer *c*, ChannelEmuContainer *channelsEmu*) const

For each node in the input container,implements ns3::Sica The program will assert if this method is called on a container with a node that already has a Sica object aggregated to it.

Parameters

c NodeContainer that holds the set of nodes on which to install the new agent.

Parameters

channelsEmu the channel emulators which are simulated the channel situations for scenario

2.14.2.3 void ns3::SicaHelper::Set (std::string *name*, const AttributeValue & *value*)

This method controls the attributes of ns3::Sica::Sica.

Parameters

name the name of the attribute to set

Parameters

value the value of the attribute to set.

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-helper.h

2.15 ns3::SicaNeighbors::SicaNeighbor Struct Reference

Neighbor description.

`#include <sica-neighbor.h>`

Public Member Functions

- SicaNeighbor (uint32_t id, uint32_t h, uint32_t r, uint32_t ch, Address rAddr, Address tAddr, Time utime, Time stime, uint32_t nch)

  c-tor

Public Attributes

- uint32_t m_id

  Node Id of the neighbor.

- uint32_t m_hopCount

  Distance to the neighbor.

- uint32_t m_neighborRadio

  Number of radio interface it has.

- uint32_t m_neighborChannel

  Channel of the receiving radio.

- Address m_rAddr

  < Address of the receiving radio interfaces

- Address m_tAddr

  < Address of the transmitting radio interfaces

- Address m_cAddr

  < Address of the common radio if any

- Time m_updateTime

  Time stamp, which shows the moment that the information is updated.

- Time m_switchTime

  Shows that when a neighbor will switch its receiving interface to another channel.

- uint32_t m_neighborNewChannel

  New channel for neighboring node where it will switch after m_switchTime.

- bool close

2.15.1 Detailed Description

Neighbor description.

2.15.2 Member Data Documentation

2.15.2.1 bool ns3::SicaNeighbors::SicaNeighbor::close

Variable for future need!!

The documentation for this struct was generated from the following file:

- src/mrmc/sica/sica-neighbor.h

2.16 ns3::SicaNeighbors Class Reference

SicaNeighbors defines the table structure for saving neighboring nodes' information in Sica.

`#include <sica-neighbor.h>`

Classes

- struct SicaNeighbor
  Neighbor description.

Public Member Functions

- SicaNeighbors ()
  c-tor
- virtual ~SicaNeighbors ()
  d-tor
- SicaNeighbor * FindNeighbor (uint32_t id)
  Find neighbor and node with ID id return the pointer.
- int32_t FindDeviceAddr (Address addr)
  Find Id of neighbor which has a device with the address addr.
- Time GetNiUpdateTime (uint32_t id)
  Return expire time for neighbor node with ID id, if exists, else return 0.
- void SetNiUpdateTime (uint32_t id, Time uTime)
  Set the update time of a neighbor.
- Time GetNiSwitchTime (uint32_t id)
  Return switching time for neighbor node with ID id, if exists, else return 0.
- void SetNiSwitchTime (uint32_t id, Time sTime)
  Set the update time of a neighbor.
- uint32_t GetNiHops (uint32_t id)
  Return the hop counts of a neighbor.

- void SetNiHops (uint32_t id, uint32_t hops)

  Set the hop counts of a neighbor.

- bool Update (uint32_t id, uint32_t hops, uint32_t radio, uint32_t channel, Address rAddr, Address tAddr, Time updateTime, Time switchTime, uint32_t newChannel)

  Update the information for the neighbor with ID id, return false if the received information is older than the current.

- void RmvNeighbor (uint32_t id)

  Remove neighbor by with ID id.

- int32_t GetNiChannel (uint32_t id)

  Return the receiving channel of the neighbor with ID id.

- void SetNiChannel (uint32_t id, uint32_t nCh)

  Set neighbor's channel.

- int32_t GetNiTChannel (uint32_t id)

  Return the transmitting channel of the neighbor with ID id.

- void SetNiTChannel (uint32_t id, uint32_t nCh)

  Set neighbor's channel.

- int32_t GetNiNewChannel (uint32_t id)

  Return the new channel that the neighbor will switch to.

- void SetNiNewChannel (uint32_t id, uint32_t nCh)

  Set neighbor New channel.

- Address GetNiRAddress (uint32_t id)

  Return the address of the receiving interface of the neighbor.

- void SetNiRAddress (uint32_t id, Address rAddr)

  Set the address of the receiving interface of the neighbor.

- Address GetNiTAddress (uint32_t id)

  Return the address of the transmitting interface of the neighbor.

- void SetNiTAddress (uint32_t id, Address rAddr)

  Set the address of the transmitting interface of the neighbor.

- Address GetNiCAddress (uint32_t id)

  Return the address of the c interface of the neighbor.

- void SetNiCAddress (uint32_t id, Address cAddr)

  Set the address of the c interface of the neighbor.

- bool IsDirectNeighbor (uint32_t id)

  Check that node with ID id is direct neighbor.

- bool IsDirectNeighborByIndex (uint32_t i)
  Check that node in i th place of neighbor table is direct neighbor.
- uint32_t GetNiOnChannel (uint32_t channel)
  Return the number of neighbors on a specific channel.
- uint32_t GetNiOnChannelByHops (uint32_t channel, uint32_t fromHopC, uint32_t toHopC)
  Return the number of neighbors which are far as hopC and have a radio a specific channel.
- int32_t GetNeighborIdByIndex (uint32_t i)
  Return the IP address of the receiving radio of i th neighbor from neighbor list.
- int32_t GetNiChannelByIndex (uint32_t i)
  Return the receiving channel of the i th neighbor from neighbor list.
- void PrintNeighborTable (std::ostream &os)
  Print the information stored in neighbor table.
- uint32_t GetNiNo ()
  Return number of entris in neighbor table.
- uint32_t GetNiNobyHops (uint32_t fromHopC, uint32_t toHopC)
  Return the number of neighbors from distance fromHopC to toHopC.
- void RmvExpiredNi (Time maxExpireTime)
  Remove expired neighbor information.
- void Clear ()
  Cleare neighbor list.

2.16.1 Detailed Description

SicaNeighbors defines the table structure for saving neighboring nodes' information in Sica.

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-neighbor.h

2.17 ns3::SicaQueue Class Reference

SicaQueue is used to handle more than one data and signal queue for each node.

`#include <sica-queue.h>`

Classes

- struct SicaChannelQueue
  Sica Channel Queue is a structure which stores data and signal queue for one channel.

Public Member Functions

- SicaQueue ()

  c-tor

- virtual ~SicaQueue ()

  d-tor

- SicaChannelQueue * FindChannelQueue (uint32_t ch)

  Return the pointer to the queue associated to the channel "ch".

- uint32_t GetSize (uint32_t ch, SicaQueueEntry::PacketType ptype)

  Number of entries in the channel queue (hello queue or data queue).

- void CpQueuEntry (SicaQueueEntry *cpy, SicaQueueEntry origin)

  Copy the entry fields of.

- bool Enqueue (uint32_t ch, SicaQueueEntry *ent)

  Push entry in channel queue, if there is no entry with the same packet and destination address in queue.

- SicaQueueEntry * Dequeue (uint32_t ch, SicaQueueEntry::PacketType ptype)

  Return first entry of channel queue for given packet type (hello or data) This function does not remove packets from queue.

- SicaQueueEntry * DequeueWithDest (uint32_t ch, uint32_t dst)

  Return the earliest entry found for the given destination from data queue of the given channel.

- bool EraseWithDest (uint32_t ch, uint32_t dst)

  Remove the entry for the given destination address from data queue of the given channel.

- std::vector< SicaQueueEntry >::iterator FindQueueEntryForDest (uint32_t ch, uint32_t dst)

  Return the pointer to the entry for the given destination address from data queue of the given channel,, Return queue.end if there is no entry.

- void EraseFront (uint32_t ch, SicaQueueEntry::PacketType ptype)

  Remove the first packet in the queue corresponding to the given packet type (hello or data) Close the channel queue on which we have no neighbor.

- uint32_t Purge (uint32_t ch, SicaQueueEntry::PacketType ptype)

  Remove the expired packets from the channel queue of the given channel and given type (hello or data).

- void ShuffleData (uint32_t originCh, uint32_t targetCh, uint32_t addr)

  Move all data packets related to the node with the given address from one channel to another channel queue.

- void ShuffleDataALL (uint32_t originCh, uint32_t targetCh)

  Move all data packets from one channel to another channel queue.

- void CloseQueue (uint32_t ch)

  Close a queue and erase all packets.

- SicaChannelQueue * CreatQueue (uint32_t ch)

Create the queue of a channel on which we have one neighbor.

- double ComputeFlowNumber (uint32_t ch)

  find how many flows are in one channel queue (packets with different sources)

2.17.1 Detailed Description

SicaQueue is used to handle more than one data and signal queue for each node. It would attached to each node and maintain all tasks about hello and data packets queuing. It stores/retrieves packets to/from the specified channel queue.

2.17.2 Member Function Documentation

2.17.2.1 void ns3::SicaQueue::CpQueuEntry (SicaQueueEntry * *cpy*, SicaQueueEntry *origin*)

Copy the entry fields of.

Parameters

origin into

Parameters

cpy Queue entry

2.17.2.2 SicaQueueEntry* ns3::SicaQueue::Dequeue (uint32_t *ch*, SicaQueueEntry::PacketType *ptype*)

Return first entry of channel queue for given packet type (hello or data) This function does not remove packets from queue.

Parameters

ch the channel ID

Parameters

ptype Hello or Data type

Returns

A pointer which contains a copy of queue entry

2.17.2.3 SicaQueueEntry* ns3::SicaQueue::DequeueWithDest (uint32_t *ch*, uint32_t *dst*)

Return the earliest entry found for the given destination from data queue of the given channel.

Parameters

ch The channel ID

Parameters

dst ID of the destination

Returns

a pointer to a queue entry

2.17.2.4 bool ns3::SicaQueue::EraseWithDest (uint32_t *ch*, uint32_t *dst*)

Remove the entry for the given destination address from data queue of the given channel.

Parameters

ch The channel ID

Parameters

dst ID of the destination

2.17.2.5 std::vector<SicaQueueEntry>::iterator ns3::SicaQueue::FindQueueEntryForDest (uint32_t *ch*, uint32_t *dst*)

Return the pointer to the entry for the given destination address from data queue of the given channel,, Return queue.end if there is no entry.

Parameters

ch The channel ID

Parameters

dst ID of the destination

Returns

an iterator which refer to the queue entry

2.17.2.6 uint32_t ns3::SicaQueue::Purge (uint32_t *ch*, SicaQueueEntry::PacketType *ptype*)

Remove the expired packets from the channel queue of the given channel and given type (hello or data).

Returns

(Queue size)

Parameters

ch The id of the channel

Parameters

ptype the type of the queue selected from SicaQueueEntry::PacketType

2.17.2.7 void ns3::SicaQueue::ShuffleData (uint32_t *originCh*, uint32_t *targetCh*, uint32_t *addr*)

Move all data packets related to the node with the given address from one channel to another channel queue.

Parameters

originCh the source channel

Parameters

targetCh the destination channel

Parameters

addr ID of the destination node

2.17.2.8 void ns3::SicaQueue::ShuffleDataALL (uint32_t *originCh*, uint32_t *targetCh*)

Move all data packets from one channel to another channel queue.

Parameters

originCh the source channel

Parameters

targetCh the destination channel

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-queue.h

2.18 ns3::SicaQueueEntry Class Reference

This class defines the entry format for either hello or packet queue. A time stamp is used to delete old entries.

`#include <sica-queue.h>`

Public Types

- enum PacketType { Hello_Type = 1, Data_Type = 2 }
  used to differenciate packets for services.

Public Member Functions

- SicaQueueEntry (Ptr< Packet > p, PacketType ptype)
  c-tor

- virtual ~SicaQueueEntry ()

  d-tor

- bool operator== (SicaQueueEntry const &o) const

  Compare queue entries.

- PacketType GetPacketType ()

  Return the type packet in queue entry.

- void SetPacketType (PacketType ptype)

  Return the packet type (hello or data) in queue entry.

- Ptr< Packet > GetPacket ()

  Return the pointer to the packet in queue entry.

- void SetPacket (Ptr< Packet >p)

  Put a packet in queue entry.

- void SetExpireTime (Time exp)

  Set the expire time of the queue entry equal to the simulation time plus the expiration time in second.

- Time GetExpireTime ()

  Get the expire time of the queue entry, if the return value is less than zero the packet should be removed.

2.18.1 Detailed Description

This class defines the entry format for either hello or packet queue. A time stamp is used to delete old entries.

2.18.2 Member Enumeration Documentation

2.18.2.1 enum ns3::SicaQueueEntry::PacketType

used to differenciate packets for services.

Enumerator:

Hello_Type SicaQueueEntry contains hello message.
Data_Type SicaQueueEntry contains data message.

2.18.3 Member Function Documentation

2.18.3.1 bool ns3::SicaQueueEntry::operator== (SicaQueueEntry const & *o*) const `[inline]`

Compare queue entries.

Parameters

o the queue entry for comparison

Returns

true if equal

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-queue.h

2.19 ns3::SicaRoutingTableEntry Class Reference

A record of a routing table entry for static routing in Sica.

`#include <sica-rtable.h>`

Public Member Functions

- SicaRoutingTableEntry ()
  This constructor does nothing.
- SicaRoutingTableEntry (SicaRoutingTableEntry const &route)
  Copy Constructor.
- SicaRoutingTableEntry (SicaRoutingTableEntry const *route)
  Copy Constructor.
- SicaRoutingTableEntry (uint32_t srcId, uint32_t dstId, uint32_t nextHopId, double metric)
  Copy Constructor.
- uint32_t GetSrc (void) const
- void SetSrc (uint32_t srcId)
  Set the node id of the source of this route.
- uint32_t GetDest (void) const
- void SetDest (uint32_t dstId)
  Set the node id of the destination of this route.
- uint32_t GetNextHop (void) const
- void SetNextHop (uint32_t nextHopId)
  Set the node id of the next hop of this route.
- double GetMetric (void) const
- void SetMetric (double metric)
  Set the metric of this route.

2.19.1 Detailed Description

A record of a routing table entry for static routing in Sica.

2.19.2 Constructor & Destructor Documentation

2.19.2.1 ns3::SicaRoutingTableEntry::SicaRoutingTableEntry (SicaRoutingTableEntry const & *route*)

Copy Constructor.

Parameters

route The route to copy

2.19.2.2 ns3::SicaRoutingTableEntry::SicaRoutingTableEntry (SicaRoutingTableEntry const * *route*)

Copy Constructor.

Parameters

route The route to copy

2.19.2.3 ns3::SicaRoutingTableEntry::SicaRoutingTableEntry (uint32_t *srcId*, uint32_t *dstId*, uint32_t *nextHopId*, double *metric*)

Copy Constructor.

Parameters

srcId source Id

Parameters

dstId destination Id

Parameters

nextHopId Id of next hop node to the destination

Parameters

metric route metric

2.19.3 Member Function Documentation

2.19.3.1 uint32_t ns3::SicaRoutingTableEntry::GetDest (void) const

Returns

The node id of the destination of this route

2.19.3.2 double ns3::SicaRoutingTableEntry::GetMetric (void) const

Returns

The metric of this route

2.19.3.3 uint32_t ns3::SicaRoutingTableEntry::GetNextHop (void) const

Returns

The node id of the next hop of this route

2.19.3.4 uint32_t ns3::SicaRoutingTableEntry::GetSrc (void) const

Returns

The node id of the source of this route

2.19.3.5 void ns3::SicaRoutingTableEntry::SetDest (uint32_t *dstId*)

Set the node id of the destination of this route.

Parameters

dstId the id of destination node

2.19.3.6 void ns3::SicaRoutingTableEntry::SetMetric (double *metric*)

Set the metric of this route.

Parameters

metric the metric of the path

2.19.3.7 void ns3::SicaRoutingTableEntry::SetNextHop (uint32_t *nextHopId*)

Set the node id of the next hop of this route.

Parameters

nextHopId the id of the next-hop node

2.19.3.8 void ns3::SicaRoutingTableEntry::SetSrc (uint32_t *srcId*)

Set the node id of the source of this route.

Parameters

srcId the id of the source node

The documentation for this class was generated from the following file:

- src/mrmc/sica/sica-rtable.h