Definitions

The following definitions are useful in reviewing this document:

N – the number of nodes in the simulation

Overall System Requirements

Each requirement includes its source from the slides that are being passed around in parenthesis. The version used to generate this document is Ch11DDSimulation9 KFV1.pptx – another version.

1. Support a 100x100 grid (slide 12)
2. Number of sensor types is fixed from 1 to N (slide 18); this corresponds to 1 to N data types (slide 13)
3. Each node is randomly given a sensor type from 1 to N, assigned by the interest packet.
4. Support for one source node (derived from slide 18)
5. Support for one sink node (derived from slide 18)
6. The simulator will use individual threads for executing the model (slide 13)
   1. The RDSendFlag will be set to tell a node to reinforce the least delay gradient by sending a reinforced data packet
7. Each thread will have various flags that are checked to determine status checking, exploratory data, etc at each interval. (reference?)
8. Each nodes is capable of sensing 1 to N different type
9. The user sets a sink Node with the UI (slide 13).
10. The user sets a data interest from the UI (slide 13). \*\*\* This one does not seem very clear right now; ie how do we set the data interest? It is listed on the Node Information, so per-node?.\*\*\*

# Node

## Requirements

**PacketTransmitter** to send packets to other nodes

**PacketReceiver** to receive packets from other nodes

**Residual energy**

**Node** slide 4 (Node Model) defines *resEnergy*

**DataInterest** for storing the list of data interests that the node has

Must be able to add a data interest

1. Residual Energy
2. Data Interests:
3. Node type: normal, source, sink [slide 18]
   1. Source nodes cannot be sink node, and vice versa
   2. One node is the source node
   3. One node is the sink node

Gradient

Slide 4 – do not know about DataInterest objects

senderId (id of the node that sent the gradient)

rate (?)

PacketTransmitter

Requires: a PacketToTransmit, containing the destination list and the packet that is meant to be sent. In order to model the PacketToTransmit, the PacketTransmitter will have to be able to send unicast and multicast diagrams.

Follows the basic operations outlined on slide

PacketReceiver

Methods:

receivePacket (delegate of Node.receivePacket)

Diagrams:

Requires:

ReceivedPacket (w/ *thePacket* that was received). It supports a method called getPacket() to be able to get the packet that was received.

Associated with the Nodes:

PacketTransmitter

PacketReceiver

Sending Packets:

1. nodeSender (node i) invokes nodeReceiver.receivePacket()
2. nodeReceiver delegates the call to it’s receiver (*nodeReceiver.myPacketReceiver.receivePacket*)
3. The nodeReceiver.myPacketReceiver.receivePacket method calls it’s createPacket method to create a new ReceivedPacket instance.
4. The instance of ReceivedPacket from #3 above is stored with a call to storePacket()

Data Interests

This must model the concept of the data interest by identifying the data type, [interval, duration]????

The gradient is required to identify the paths from the

Requirements from the slides (slide 4):

1. Nodes have zero to many data interests
2. Nodes can have many neighbors
3. DataInterest can have many gradients
4. Gradient objects do not know about the DataInterest objects

From 1: The nodes (Node instances) get associated to their data interests by a DataInterest object being registered into the Node::myDataInterest list.

# Packets

The class framework features one general packet class with the features:

1. packetType
2. senderID
3. interval
4. seqNumber

# Reinforced Data

This is the model of the data reinforcements that happen along the gradients. It is implemented by the class Rnf

## Flags

From slide 33:

*RnfFlag*: set means that node receives a reinforcement message.

When the node supports the interest, it creates a reinforced data object with a timer. The node’s timer is then used to signal the node’s thread at the specified interval (DataInterest::interval) to arrange to send reinforced data.

From slide 34:

When set, the thread’s flag (RDSendFlag: bool) indicates that a node must sent reinforced data packet to its least delay gradient.

The RDSendFlag gets set by the RnfTimer (see section below); the thread then calls a sendRD method to send reinforced data packet to the lowest timestamp gradient linked to the data interest. After completion, the RDSendFlag is cleared by the thread.

## Timer

To support the mechanism of data reinforcement, there is a timer in place that allows for the implementation of the occasional reinforced data send.

As per slide 5: implemented in the class RnfTimer.

The timer is associated to a ReinforcedData interest, and therefore timed using the interval specified in the DataInterest.

# Simulator Initialization

[reference: NodeTest.java, slide 16]

To setup the simulator:

1. Establish the number of dimensions. This defaults to 3 dimensions.
2. Establish the number of nodes. This defaults to 2.

# Tasks

## Coding

Node

Add PacketTransmitter myPacketTransmitter

Add PacketReceiver member - myPacketReceiver

Sending Packets (PacketTransmitter)

Implement

DataInterest:

Sensor types

Overall

1. Node placement: place each of the nodes at a random point in a 100x100 grid.

## User interface

As of right now this has no design.

Requirements:

1. Choose the number of nodes – between 1 and 10000. [slide 12]
2. Identify the radio range *r* in terms of the postman metric. [slide 12]
3. Choose sink, data interest

Suggestions/Ideas

* A simple user interface for
* provide wrapping for the classes of the simulator so that they are abstracted from GUI specifics;
  + The NodeUI class would inherit from the class Node, and provide specific class operations for using the class in the UI

Other notes:

For collaboration and sketching – could use some sort of tool?