Manual for Network Manager and the Tester

This manual uses a demo to describe how to run the Network Manager and the Tester to setup a multi-hop WirelessHART network, construct reliable routing graphs, and generate the communication schedules among devices in the network. The Tester plays the role of a physical WirelessHART network. It sends messages to and receives responses back from the Network Manager. The GUI of the Network Manager can display the WirelessHART network topology, device information, messages exchanged between the Network Manager and the Tester, reliable routing graphs and communication schedules, and the bandwidth usage for each device.

**Step 1.** Install Java SDK and IDE: NetBeans 6.8 + jdk1.6.0_18 or jdk1.6.0_20

You can download Java SE and NetBeans Co-bundle (JDK 6u20 and NB 6.8) from here: http://java.sun.com/javase/downloads/widget/jdk_netbeans.jsp

**Step 2.** Open the Mesh project using the NetBeans IDE. Run the MeshApp.java (Network Manager) and TestFrame.java (Tester) respectively.

Figure 1. The Overview of Network Manager
Step 3. Setup the connection between Network Manager and the Tester.

The messages send to Network Manager is displayed here

Users can send specific commands to the Network Manager by inputing corresponding parameters

The messages received from Network Manager is displayed here

Figure 2. The Overview of the Tester
**Step 4.** Start the demo to setup a multi-hop WirelessHART network with one Gateway, 2 Access Points, and 8 field devices. In the demo, each field device is configured with a scan period for bursting data to the Gateway periodically. The message sequence in the demo is collected by manually generating the commands in the Tester. The messages are sent to Network Manager every 5 seconds.
Figure 4. Start the Demo

Figure 5. The network topology and the device information in the Demo
Step 5. Construct the global reliable uplink graph, global reliable broadcast graph, and the reliable downlink graph for each field device.
Figure 7. Construct the global reliable uplink graph

In the reliable uplink graph, each device has at least two neighbors to forward the messages to the Gateway.

Figure 8. Construct the reliable downlink graph: The downlink graph for device 10 is shown

In the reliable downlink graph, each intermediate field device has at least two neighbors to forward the messages to the destination.
Figure 9. Construct the global reliable broadcast graph

**Step 6.** Based on the network topology and the constructed reliable routing graphs in step 5, we can further generate the communication schedules among devices.
Construct the communication schedules in the network

The global schedule of the network: it is a \(<\text{time slot, channel}>\) matrix

Figure 10. The global schedule of the network based on the routing graphs, network topology, and network statistics

The communication schedule for each device

Figure 11. The communication schedule for each device in the network
Figure 12. The bandwidth usage for each device in the network

**Step 7.** Take snapshots for both the routing graphs and the communication schedules based on the current network topology.
Figure 13. Take the snapshots for both the routing graphs and the communication schedules

**Step 8.** Change the network topology by either send command from the Tester or directly make change in Network Manager GUI.
Figure 14. Change the network topology by sending commands from the Tester

Send command 788: path failure to Network Manager. The Network Manager will remove the link from dev 0004 to dev 000a.

In the Network Manager GUI, delete the link from dev 000a to dev 0004

Figure 15. Manually change the network topology in the Network Manager GUI
Figure 16. The updated network topology after the two changes

**Step 8.** Compare the routing graphs and communication schedules before and after the network topology change. Notice here that you have to construct the new reliable uplink graph, downlink graphs, broadcast graph, and the communication schedules before we conduct the comparisons.

The two links between dev 0004 and dev 000a are removed.
Both of dev 000a and dev 000a are removed from the reliable uplink graph.

Figure 17. Comparison of the uplink graph before and after the change of the network topology

The reliable broadcast graph is changed.

Figure 18. Comparison of the broadcast graph after the network topology changes
Figure 19. Comparison of the downlink graph of dev 000a after the network topology changes

The reliable downlink graph of dev 000a cannot be constructed

Figure 20. Comparison of the downlink graph of dev 0009 after the network topology changes

The reliable downlink graph of dev 0009 remains the same after the topology changes
Figure 21. Comparison of the communication schedule of dev 000a after the topology changes.

Dev 000a is isolated because it is not in the reliable uplink graph. Its configured links are removed.

Figure 22. Comparison of the communication schedule of dev 0005 after the topology changes.

The communication schedule of Dev 0005 is changed, and the updates on the links are listed as below.