TREK

DELAY TOLERANT NETWORKING (DTN) TUTORIAL

April 2016

Approved for Public Release; Distribution is Unlimited.

TABLE OF CONTENTS

PARAGRAPH PAGE

1 Welcome ... 1

1.1 System Requirements ... 1

2 Technical Support ... 1

3 Delay Tolerant Networking (DTN) ... 2

4 Interplanetary Overlay Network (ION) ... 5

4.1 ION Administration Programs ... 5

4.1.1 ionadmin ... 6

4.1.2 ionsecadmin ... 6

4.1.3 bpadmin ... 6

4.1.4 ipnadmin ... 6

4.1.5 cfdpadmin ... 7

4.1.6 ltpadmin ... 7

4.2 ION Configuration Files ... 7

4.3 ION Feedback ... 9

4.3.1 The ion.log File ... 9

4.3.2 Standard Output ... 10

4.4 The ION Application Programming Interface ... 10

5 TReK DTN Capabilities ... 10

5.1 HPEG Application ... 10

5.2 IONconfig Application ... 11

5.3 IONizer API ... 11

5.4 IONizer Application ... 11

5.5 CFDP Application (ION CFDP mode) ... 11

5.6 CFDP Console (ION CFDP mode) ... 11

6 The ISS DTN Configuration ... 12

7 Step-by-Step Tutorials ... 13

7.1 Two Nodes ... 13

7.1.1 IONconfig ... 14

7.1.2 IONizer ... 29

7.1.3 CFDP ... 31

7.2 Four Nodes ISS ... 39

7.3 Two Nodes with LTP ... 42

8 Terminology ... 50

FIGURES

FIGURE PAGE

Figure 1 A Simple DTN Example 3
Figure 2 A 4-Node DTN Example 4
Figure 3 ISS DTN Architecture 12
Figure 4 Two Node Example 13
Figure 5 IONconfig 14
Figure 6 IONconfig with Directory 15
Figure 7 IONconfig with Two Nodes 15
Figure 8 IONconfig with Two Connected Nodes 16
Figure 9 Node Properties 17
Figure 10 General Node Properties (Node 1535) 18
Figure 11 Configuration Properties 19
Figure 12 Configuration Properties (Node 1535) 20
Figure 13 Administration Properties 21
Figure 14 Bundle Protocol Properties 22
Figure 15 Bundle Protocol Properties (Node 1535) 23
Figure 16 CFDP Properties 24
Figure 17 LTP Properties 25
Figure 18 Security Properties 26
Figure 19 Duct Properties 28
Figure 20 IONizer Main Window 29
Figure 21 IONizer with Config File Directory 30
Figure 22 IONizer Main Window After Staring ION 31
Figure 23 CFDP Application Main Window 32
Figure 24 CFDP Configure Dialog 33
Figure 25 CFDP with ION CFDP Mode Selected 34
Figure 26 CFDP Configure Dialog Options Tab 36
Figure 27 CFDP Configure Dialog Options Tab with Data 37
Figure 28 CFDP Active in ION CFDP Mode. 38
Figure 29 CFDP Main Window with Complete File Transfer Information 39
Figure 30 Four Node ISS Example 40
Figure 31 IONconfig 4-node ISS 40
Figure 32 Two Node Example with LTP 43
Figure 33 LTP Properties 44
Figure 34 LTP Properties (Node 1535) 45
Figure 35 Duct Properties 46
Figure 36 Duct Properties (LTP Part 1) 48
Figure 37 Duct Properties (LTP Part 2) 49

TABLES

TABLE PAGE

Table 1 ION Configuration Files Generated by IONconfig 8
Table 2 TReK Related Files Generated by IONconfig 9
Table 3 General Node Properties (Node 1535) 18
Table 4 Configuration Properties (Node 1535) 19
Table 5 Bundle Protocol Properties (Node 1535) 22
Table 6 General Node Properties (Node 9000) 26
Table 7 Configuration Properties (Node 9000) 26
Table 8 Bundle Protocol Properties (Node 9000) 27
Table 9 Configuration Parameters for ION CFDP 35
Table 10 Payload Ground Changes 41
Table 11 Payload Flight Changes 42

1 Welcome

The Telescience Resource Kit (TReK) is a suite of software applications and libraries that can be used to monitor and control assets in space or on the ground.

This tutorial describes Delay Tolerant Networking (DTN) from an end user perspective. There is a general discussion on what DTN is and how it can benefit the International Space Station (ISS) payload community and others. A brief introduction is also provided for the Interplanetary Overlay Network (ION) which is the DTN implementation used by TReK. The DTN related capabilities in TReK are described along with step-by-step tutorials.

There are many new terms associated with DTN. These terms are defined when they are first used in the text of the document and also collected in the Section 8 of this document.

1.1 System Requirements

Windows 7, Red Hat Enterprise Linux 6.x.

2 Technical Support

If you are having trouble installing the TReK software or using any of the TReK software applications, please try the following suggestions:

Read the corresponding material in the manual and/or on-line help.

Ensure that you are correctly following all instructions.

Checkout the TReK Web site at http://trek.msfc.nasa.gov/ for Frequently Asked Questions.

If you are still unable to resolve your difficulty, please contact us for technical assistance:

TReK Help Desk E-Mail, Phone & Fax:

E-Mail:	trek.help@nasa.gov
Telephone:	256-544-3521 (7:00 a.m. - 3:30 p.m. Central Time)
Fax:	256-544-9353

TReK Help Desk hours are 7:00 a.m. – 3:30 p.m. Central Time Monday through Friday. If you call the TReK Help Desk and you get a recording please leave a message and someone will return your call. E-mail is the preferred contact method for help. The e-mail message is automatically forwarded to the TReK developers and helps cut the response time.

3 Delay Tolerant Networking (DTN)

Computers attached to the terrestrial Internet are essentially always able to communicate and do so instantaneously. While there can be outages that prevent connectivity, these outages tend to be isolated and relatively short term. For communication between a computer on Earth and one in orbit around Earth or on a deep space mission, the outages are more frequent and of longer duration. Protocols such as TCP/IP that work great for terrestrial networks can have problems in space networks when connectivity is not continuous or there is a long delay in communications.

Some programs such as ISS have implemented IP protocols for the space environment. This provides the ability for payload teams to use many of the same protocols common to the terrestrial Internet with orbiting experiments. Payload teams can use application layer protocols such as secure shell (SSH) to interact with their onboard payload. While communications with the payload are frequent and of relatively long duration for a space environment, they are not continuous.

Delay or disruption in any path can cause problems in the transfer of data. The flight and payload computers of space systems must often build specialized code to store the data during periods of disruption and send it when a transmission link becomes available. If data is identified as lost on the receiving end, a means of requesting the retransmission of lost data must be provided. DTN can address this problem and alleviate the need for this type of work.

A delay tolerant network is composed of multiple computers or nodes. A single computer can run one or more nodes although usually only a single node will exist on a computer. The node is identified by a node number. The node number is unique across the entire DTN. You can consider the node number the DTN equivalent of an IP address. It identifies what node sent the data and what node is to receive it. A computer with multiple nodes is similar to a computer with multiple network cards/IP addresses.

Each node has the knowledge of other nodes with which it may directly communicate and when those communication links are available. For nodes where direct communication is not possible, DTN gateways can be used. Any DTN node that routes data from one DTN node to another DTN node is considered a gateway. Gateways are usually not the final destination for the data, but can be. The data sent between nodes are called bundles. A bundle is the primary transfer block of data over a DTN[1]. A simple example can help illustrate this concept.

Figure 1 shows a simple four node DTN with each node in series. Node 1 only knows how to communicate with Node 2. Node 1 also knows when the communication link is available for Node 2. If Node 1 needs to send bundles to either Node 3 or Node 4, then it must send the bundle through a gateway. In this case the gateway is Node 2. Node 1 does not need to know when the communication links are available from Node 2 to Node

[1]The Bundle Protocol Specification is available if you wish to review the details. Just type “RFC-5050” into your favorite search engine.

3 or Node 3 to Node 4. It only knows that by sending the bundles to Node 2 the bundles will get to the other nodes.

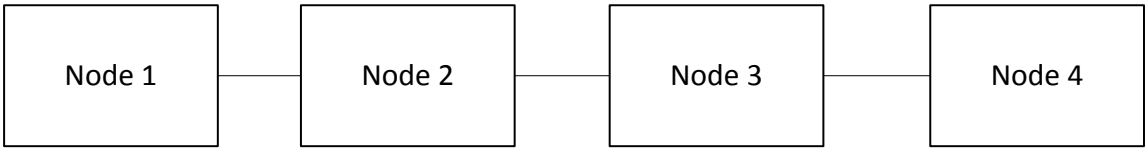

Figure 1 A Simple DTN Example

DTN can be used as a store and forward network since end-to-end communication is not always possible. A node can send bundles to another node that will store the bundles until it is possible to forward it to the next node. If custody transfer is implemented for the store and forward network, the node sending the bundle can safely delete its copy. Custody transfer is a request made from the node sending the data to the node receiving the data. If the receiving node accepts custody of the data, the sending node can delete its copy of the data. By accepting custody of the data the receiving node accepts responsibility for ensuring the data is transferred to the final destination or until it can pass custody to another node. By passing custody in this manner a single copy of the data can be maintained in the DTN.

Often a primary task of payloads is the transfer of files to and from the payload. Prior to the availability of IP protocols onboard ISS, each payload team had to build a unique file transfer protocol. Most often this involved splitting the file across multiple CCSDS packets[2] at the sending end and reassembling the file on the receiving end. If packets were missing a mechanism to retransmit the missing data or the entire file had to be provided.

When IP protocols are available other means of data transfer is possible. If the planned communication period and bandwidth available for contact with the spacecraft is sufficient for the size of the file, standard file transfer mechanisms such as secure file transfer protocol (SFTP) and secure copy (SCP) can be used. These means of file transfer work well, but may have issues if communication is lost due to a planned loss of signal or an unplanned interruption.

The CCDSS File Delivery Protocol[3] (CFDP) helps alleviate the problems experienced by standard network protocols by addressing the issues of delay and disruption. CFDP can handle long delay in communications and also has the means of retransmitting the data that is lost due to disruption. While it is possible to run CFDP over non-IP networks, it became available as a standard service for ISS when IP protocols were enabled. CFDP

[2] The Consultative Commit for Space Data Systems (CCSDS) provides multiple international standards for use in space programs. The Space Packet Protocol is document number CCSDS 133.0-B-1 and is available on the CCSDS website at http://public.ccsds.org/publications/BlueBooks.aspx.
[3] The CFDP document number is CCSDS 727.0-B-4 and is available on the CCSDS website at http://public.ccsds.org/publications/BlueBooks.aspx.

allows for better multiple contact file transfers, but often needs to be knowledgeable about contact periods so that the timeouts and limits associated with the CFDP protocol behave properly. CFDP can also be used with DTN. In these instances the retransmission of lost data is provided by the DTN protocols.

DTN can provide extra services for end users that aren't available with standard IP protocols. The best way to explain some of the advantages of DTN is with a couple of examples. The first example will be a large file transfer from a spacecraft computer system to the ground. The second example will be a file transfer from a ground computer to the spacecraft.

For each of these examples we will use a four node configuration like the one shown in Figure 2. In this example the two gateways can communicate with each other, but the communication is not always available. The end nodes each have continuous contact with their respective gateway.

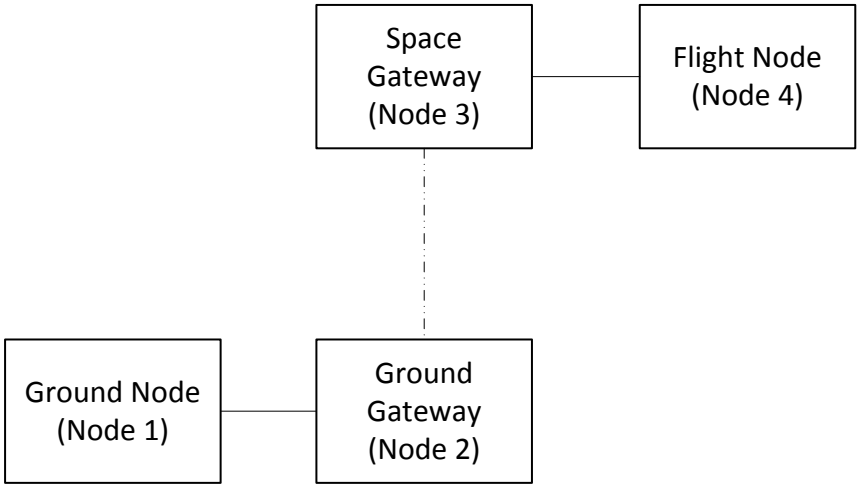

Figure 2 A 4-Node DTN Example

The first example is transfer of a large file from the flight node to the ground node. In our example, the transfer of the file from node 4 to node 3 can occur relatively quickly since the nodes are continuously available for sending and receiving data. As the bundles composing the file arrive at node 3 they can be sent along to node 2 if there is communication available. Otherwise, node 3 stores the data and forwards it once a link becomes available. When the bundles arrive at node 2 they can be immediately forwarded to node 1 which will then reassemble the file.

If node 1 is not connected to node 2 when the bundles arrive from node 3, node 2 will store the bundles and forward them once node 1 is able to connect again. In this example the flight node and ground node are both shielded from knowing when a communication link is available. The gateway nodes handle that for them. Also, if a node is not available, data is stored and forwarded when the node is once again connected.

The second example involves sending a file from the ground node to the flight node. Sending a file from the ground to the spacecraft is often a controlled activity. For this example, let's assume that you need to uplink a configuration file to your flight node. Perhaps the operations team controlling the ground gateway is required to enable you for sending the file. If other users currently have priority, your first opportunity may conflict with watching a very important football game. Instead of missing the game, you could send the file to the ground gateway and it would automatically uplink the file once uplink is enabled by the operations cadre. If the scheduled time available for uplinking the file shifted, the delay tolerant network will automatically be aware of the shift and with no intervention from you take care of the problem preventing you from missing the pregame buildup or postgame celebration.

The disruptions and delays associated with space links are the primary motivations for establishing delay tolerant networks. A properly designed DTN can insulate systems from the knowledge of when contacts are available and provide a storage network that includes the automatic retransmission of missing data. While the above examples are file related the same principles can be applied to other types of data such as telemetry and video.

4 Interplanetary Overlay Network (ION)

Delay tolerant networking is described in RFC-4838 and includes several other specifications. Fortunately, several implementations of the DTN protocols are available including the Interplanetary Overlay Network (ION) which is used by TReK. This section briefly describes ION and introduces a few ideas that are important in understanding how TReK works with ION.

ION is written in C and is compilable on many different operating systems including the ones supported by TReK. TReK provides a Windows installation with .lib files needed for compiling with Visual Studio programs. An RPM file is also available for installation of RedHat Enterprise Linux 6.x systems. The ION source code[4] and TReK provided installation files include the ION design documentation as well as the description of the application programming interface provided by ION.

ION is composed of many small processes and common libraries working together to provide the entire set of DTN capabilities. These processes are managed through a series of administration programs and associated resource files. These resource files allow the user to configure the delay tolerant network.

4.1 ION Administration Programs

The following paragraphs describe the administration programs that are currently used by TReK and the general capabilities that each control. There are other administration programs in ION that are not discussed. The information provided can be helpful in

[4] ION source code is available for download at http://sourceforge.net/projects/ion-dtn/.

understanding some of the TReK applications related to DTN. The administration programs are identified by their executable name.

4.1.1 ionadmin

The ionadmin program configures the DTN node and establishes the node number that will identify the node. Additionally, the simple data recorder (SDR) is created. The SDR in ION can reside in volatile memory, on the file system, or both. The memory option allows for faster data retrieval while the file option provides non-volatile storage. The SDR is sometimes referred to as a data store in this document. One of the important considerations in configuring a DTN node with ION is the size of the SDR. ION uses a configuration item name identified as *heapWords* to determine the size of the SDR. The number of *heapWords* is multiplied by the number of bytes in the processor to determine the actual size of the SDR. If *heapWords* is set to 1,000,000, then the SDR size on a 32-bit computer will be 4 million bytes and on a 64-bit computer 8 million bytes.

4.1.2 ionsecadmin

The ionsecadmin program configures the security policy for ION. At this time, TReK only initializes the security to prevent unwanted messages from appearing in the log file.

4.1.3 bpadmin

The bpadmin program configures the bundle protocol for the ION node. This includes identifying the endpoints used by applications. Endpoints in DTN are used to segregate bundle traffic and route the appropriate data to the correct BP enabled application. The endpoints in ION use the node number and a service number. The service number allows ION to route the bundle to an application that has registered to receive it. Just as you can consider the node number as similar to an IP address, you can think of the service number as a port number. For standard applications such as CFDP that are provided by ION, you don’t need to worry about the endpoints. You will only need to add endpoints for BP enabled applications you provide.

The bpadmin program also establishes which protocols (e.g., TCP) will be used with ION. The protocols identify which convergence layers are used for the ION node.

ION uses the terms ‘duct’ to specify communication paths between nodes. An *induct* is used to receive data from another node. An *outduct* is used to send data to another node. There is typically a single induct per protocol, but some protocols require an outduct for each destination node.

4.1.4 ipnadmin

The ipnadmin program configures the plans for routing DTN bundles. The routes can be either a direct connection or via DTN gateways (static routing).

4.1.5 cfdpadmin

The cfdpadmin program configures the ION provided implementation of CFDP. The default values for CFDP should be sufficient for most users.

4.1.6 ltpadmin

The ltpadmin program configures the Licklider Transmission Protocol (LTP) for ION. LTP requires the knowledge of when transmission of data is available. In ION this knowledge is provided by contact plans. LTP provides an efficient mechanism for the retransmission of missing data over disruptive links by providing checkpoints which trigger the node receiving the data to report on what data has been received and what is missing. For the typical 4-node ISS implementation LTP is implemented at the gateway nodes and is not needed by the ground or payload flight computers. However, if one or both gateway nodes are not available LTP can be enabled on the payload flight or ground computer as needed to ensure efficient retransmission of data.

4.2 ION Configuration Files

The IONconfig application will generate all of the files needed to run ION. Table 1 shows all of the files generated by IONconfig that are used by ION administration programs. Table 2 shows the other files that are generated that are related to TReK operations. Some of these files can also be used without TReK. Items in both tables marked by an asterisk (*) are only applicable for nodes connecting to ISS via the HOSC to Payload Ethernet Gateway (HPEG) which is explained in a later section.

File	Description
contact_plan.ionrc	An ION configuration file that describes contact times between nodes. Only applicable when LTP[5] is used.
proxy.bprc*	If a proxy is used, this file configures ION to work with the proxy. If no proxy is available, the file is identical to the start.bprc file.
proxy.ipnrc*	If a proxy is used, this file configures ION to work with the proxy. If no proxy is available, the file is identical to the start.ipnrc file.
start.bprc	Contains the information for the ION bpadmin program to use when configuring an ION node.
start.cfdprc	Contains the information for the ION cfdpadmin program to use when configuring an ION node.
start.ionconfig	Contains configuration information for the SDR (data store).
start.ionrc	Contains the information for the ION ionadmin program to use when configuring an ION node.
start.ionsecrc	Contains the information for the ION ionsecadmin program to use when configuring an ION node.
start.ipnrc	Contains the information for the ION ipnadmin program to use when configuring an ION node.
start.ltprc	Contains the information for the ION ltpadmin program to use when configuring an ION node.
stopion.ionrc	Contains information needed to successfully stop an ION node.

Table 1 ION Configuration Files Generated by IONconfig

[5] See Section 8 for more information.

File	Description
chk_node_num.bat*	A Windows batch file to ensure that node numbers in the configuration files are the same as those in the current HPEG configuration.
chk_node_num.sh*	A Linux bash script to ensure that node numbers in the configuration files are the same as those in the current HPEG configuration.
deletesdr.bat	A Windows batch file to delete the SDR (data store) associated with the configuration. This should only be executed if the SDR becomes corrupt. You will see a message in the ION log file if that happens.
deletesdr.sh	A Linux bash script to delete the SDR (data store) associated with the configuration. This should only be executed if the SDR becomes corrupt. You will see a message in the ION log file if that happens.
process_temp.txt	A file containing a list of the expected processes for the IONizer to monitor.
proxy_available.txt	A file containing information on if a proxy is available for this node.
renamelog.bat	A Windows batch file that renames the log file. ION must not be running when this batch file is executed.
renamelog.sh	A Linux bash script that renames the log file. ION must not be running when this script is executed.
startion.bat	A Windows batch file that starts all of the ION processes associated with a node.
startion.sh	A Linux bash script that starts all of the ION processes associated with a node.
stopion.bat	A Windows batch file to stop all of the ION processes.
stopion.sh	A Linux bash script to stop all of the ION processes.

Table 2 TReK Related Files Generated by IONconfig

4.3 ION Feedback

ION provides feedback to the user through two primary mechanisms: the *ion.log* file that is written to the operating system directory where ION is started and standard output.[6]

4.3.1 The ion.log File

ION writes several types of messages to a log file named ion.log that can be found in the directory where ION is started. The three most important types of messages are informational, warning, and diagnostic. Informational messages are written for events that are nominal, but significant during the normal operations of ION. Warning messages

[6] TReK provides the ability to start ION and monitor both the log file and standard output in a graphical user interface. These capabilities are described in a later section.

indicate that an off-nominal event occurred, but that the cause is likely due to operational or configuration issues and not problems with the ION software. Diagnostic messages are errors that may be due to ION software errors. The information provided in the message includes the line of code where the error was detected and a description of the error. You should read the description carefully as it may indicate that your SDR is corrupt. If the SDR becomes corrupt, you must stop ION and delete it prior to restarting ION.

ION does not generate pop-up dialogs and such for these errors.[7] It is the responsibility of the ION user to monitor the ION log file for any errors. If ION isn't operating properly, the first thing you should check is the ion.log file.

4.3.2 Standard Output

ION also writes information to standard output. This usually only occurs during startup and shutdown of the ION related processes by the administration programs. There are cases where configuration issues may prevent ION from starting that are only reported to standard output.

A very useful part of standard output is something ION calls watch characters. Watch characters must be enabled to appear in standard output. They provide valuable insight into the inner working of an ION node. Watch characters provide both positive feedback (e.g., a bundle was transmitted) and negative feedback (e.g., custody was refused). These characters can help debug problems with the DTN.

4.4 The ION Application Programming Interface

ION provides an application programming interface (API) that allows you to write DTN enabled applications. The functions and libraries composing the ION API are described in detail as part of the ION documentation. For Linux users the API documentation is available as both man pages and a PDF file. For Windows users the API documentation is available as a PDF file.

5 TReK DTN Capabilities

The following sections describe the capabilities provided by TReK that are relative to DTN. There is other documentation available that describes in detail these capabilities. The actual DTN capabilities in TReK are provided by ION.

5.1 HPEG Application

The Huntsville Operations Support Center (HOSC) Payload Ethernet Gateway (HPEG) application provides access to the IP capabilities, including DTN, for ISS users. All users

[7] It's important to remember that ION was developed to run on many different platforms including those without nice graphical user interfaces for popup dialogs. Log files, however, are available everywhere.

are required to authenticate with the HOSC prior to using DTN. Information needed to properly configure ION is received from the HOSC as part of the authentication process. If at any time the connection from the HPEG Application to the HOSC is lost, DTN communications will stop. See the HPEG User Guide for details.

5.2 IONconfig Application

The ION Configuration application provides a simple means of defining a delay tolerant network and generating the ION configuration files needed for each node. Template configurations are provided that have the ISS delay tolerant network defined. You will just need to edit a few of the configurable items to get a DTN setup. See the IONconfig User Guide for details.

5.3 IONizer API

The IONizer API provides a set of functions to monitor ION. ION consists of multiple processes and each of those processes write information to a single log file. The IONizer API monitors the log file for new messages and can make these messages available to a user application for action. Additionally, the IONizer API can monitor the system to ensure all ION processes are executing. If an ION application exits unexpectedly, the application integrating the IONizer API can be notified and action taken.

The IONizer API is documented as part of the TReK Help application. You can also use the IONizer Application which already incorporates this functionality.

5.4 IONizer Application

The IONizer application incorporates the IONizer API to allow you to monitor the health of the local ION node via a graphical user interface. See the IONizer User Guide for details.

5.5 CFDP Application (ION CFDP mode)

The CFDP application can run in two modes: Native CFDP and ION CFDP. When configured for ION CFDP mode, the file transfers are automatically sent using the bundle protocol. See the TReK CFDP User Guide for details.

5.6 CFDP Console (ION CFDP mode)

The CFDP console application can also run in two modes: Native CFDP and ION CFDP. This application is suited for use in environments where a graphical user interface is not practical such as a spacecraft. See the TReK CFDP Console User Guide for details.

6 The ISS DTN Configuration

This section provides some details about the ISS DTN configuration. Figure 3 shows the configuration from the perspective of a single payload user. There are four nodes in the diagram. The payload flight computer and payload ground computer must be configured by the payload team. The Payload and HOSC gateways are configured by the ISS program.

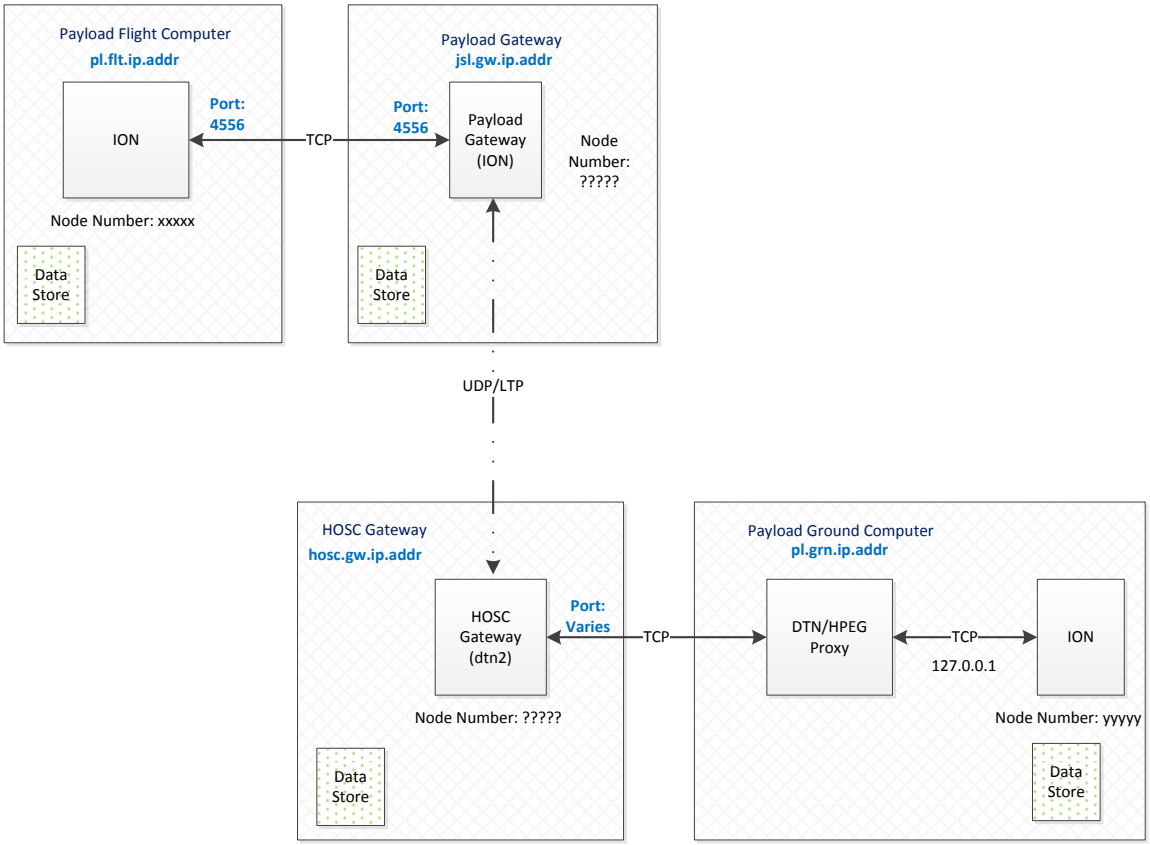

Figure 3 ISS DTN Architecture

The onboard interface between the payload's ION node and the gateways ION node is expected to be TCP[8]. The IP address of the payload flight computer and payload gateway will be known by the payload team at the time configuration is required. It is anticipated that the payload flight computer will be required to limit the data rate when sending bundles to the payload gateway. This is referred to as data throttling and is discussed later in the document.

The ground interface is slightly more complicated. The HOSC is responsible for ensuring that only authorized users have access to the ISS payloads. One of the products used to ensure that a user is valid can cause the IP address of the payload ground computer to change between sessions. Additionally, it is possible for the HOSC IP address, port number, and node number to change as well between sessions. As part of

[8] The onboard interface is currently undocumented.

the authentication process, the user must also select the node number used by the payload ground computer.

To simplify the ION configuration TReK automatically generates information to configure an application referred to as the TReK Proxy. The TReK Proxy hides the IP address and port number changes from ION. Scripts are also provided to ensure that the node numbers for both the HOSC Gateway and the payload ground computer nodes do not change between sessions.

While there is no data throttling required on the ground, the payload's bundle traffic sent to the HOSC gateway for delivery to the payload flight node is only delivered when the user is 'enabled' to send DTN traffic. You should take this into consideration when configuring timeouts associated with file transfers. It is also important to remember that each payload team has limited bandwidth available for both uplink and downlink which will make the transfer of file slower than typical terrestrial Internet file transfers.

7 Step-by-Step Tutorials

This section contains two step-by-step tutorials for configuring and running a delay tolerant network with ION. You will need a computer to run each node in the tutorial. It is possible to run multiple nodes on a single computer, but that requires setting environment variables and starting all applications from the directory ION is started.

The first tutorial contains two nodes and is applicable to all users. The second tutorial contains four nodes and is written for ISS users. If you are an ISS user, don't skip the first tutorial. It will contain information that isn't repeated in the second tutorial. The third tutorial is another two node scenario, but this time uses LTP to ensure data transmission over a disruptive link.

7.1 Two Nodes

Figure 4 shows a simple two node DTN configuration that will use TCP as the convergence layer. For this example, we will assume that DTN has been allocated 50 kbps on the uplink (node 1535 to node 9000) and 1 Mbps on the downlink (node 9000 to node 1535). You will use three TReK applications to setup and run this example: IONconfig, IONizer, and the CFDP Applications.

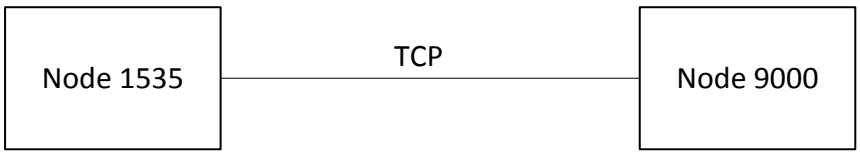

Figure 4 Two Node Example

7.1.1 IONconfig

The IONconfig application will allow you to generate the files needed to run ION.

Step 1 – Start IONconfig

Start the IONconfig application from your systems start menu. The IONconfig application shown in Figure 5 will start. The main window of IONconfig contains several toolbars we will use in the following sections. The main window also has a blank drawing area for placing and connecting nodes to describe the delay tolerant network.

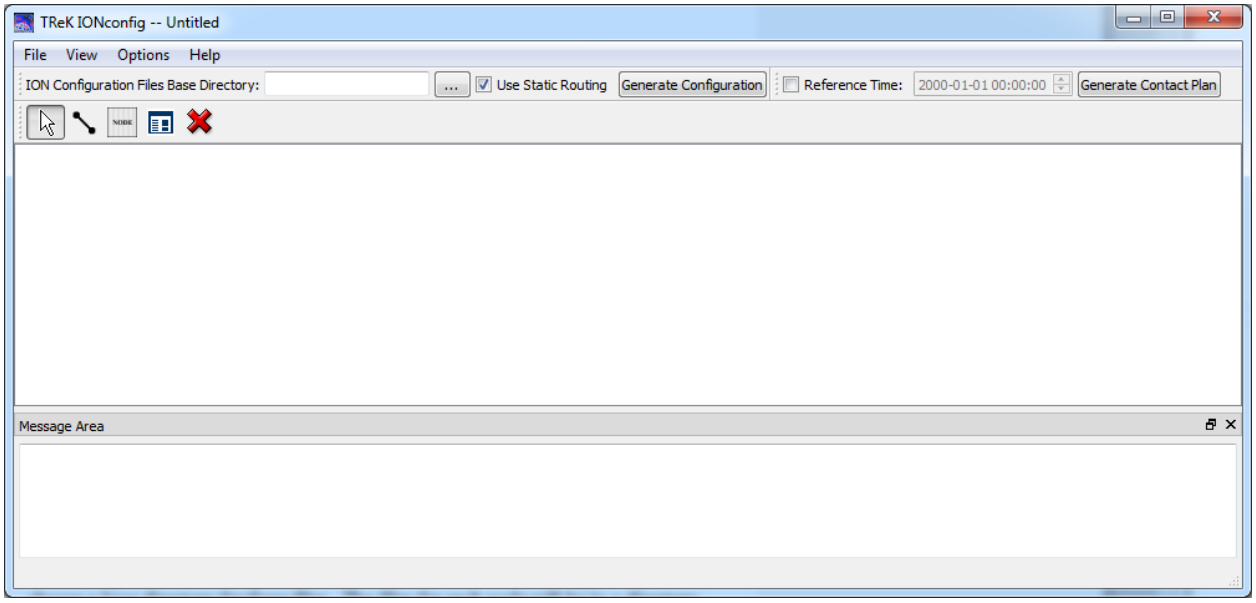

Figure 5 IONconfig

Step 2 – Choose a directory for the generated files

ION configuration files are generated for each of the nodes in the diagram. You must choose a base directory for these files. The files for each node will be in a directory named *node_xxxxx* where the xxxxx is replaced with the node number. You can either type the name of a directory in the "ION Configuration Files Base Directory" text field or use the button to the right of the field to select a directory. When you are finished the IONconfig main window will look similar to Figure 6.

Note: The directory you choose for the ION Configuration Files Base Directory must exist on the system.

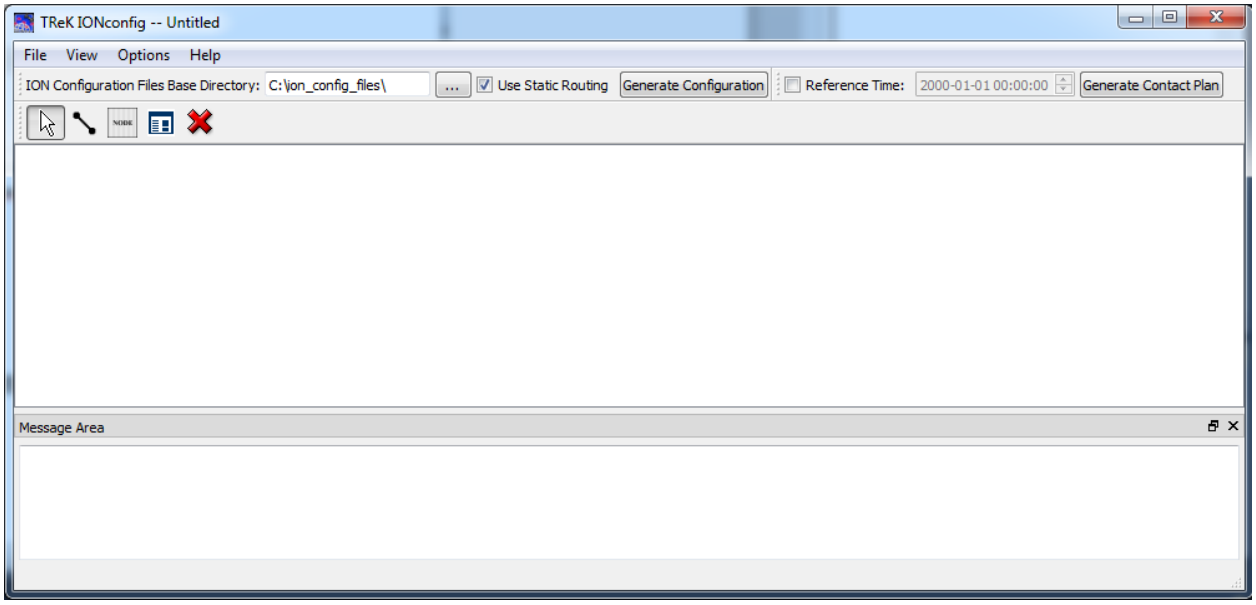

Figure 6 IONconfig with Directory

Step 3 – Draw the node diagram

To place nodes on the diagram, select the Node tool in the toolbar. Once you have selected the Node tool, just single click on the drawing area to place a node. Figure 7 shows the two nodes on the drawing area. The Node tool is selected in the toolbar. *Hint: You are selecting where to put the upper left hand corner of the node when you click on the drawing area. Once selected, the tool in the toolbar will remain selected until you select a different tool.*

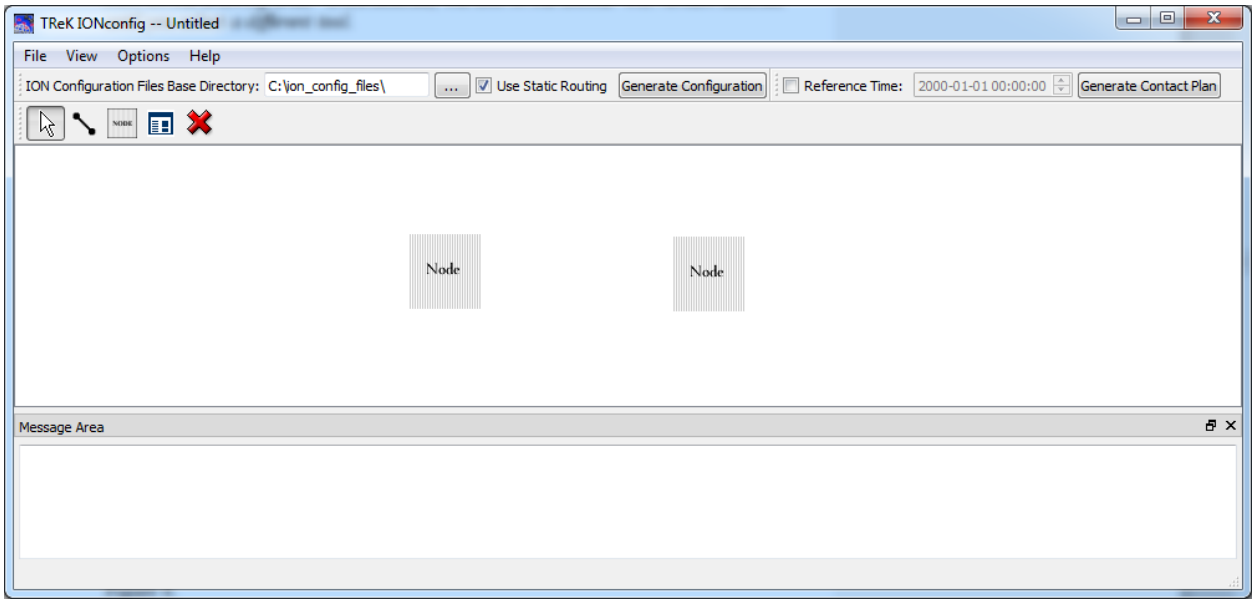

Figure 7 IONconfig with Two Nodes

Step 4 – Connect the nodes

To connect the nodes on the diagram, select the Duct tool in the toolbar. Once you have selected the Duct tool, click on a node and hold the mouse button down. Drag the line over to the other node and release the mouse button. If you miss the node when dragging the line, just try again. Your diagram should look similar to the one in Figure 8.

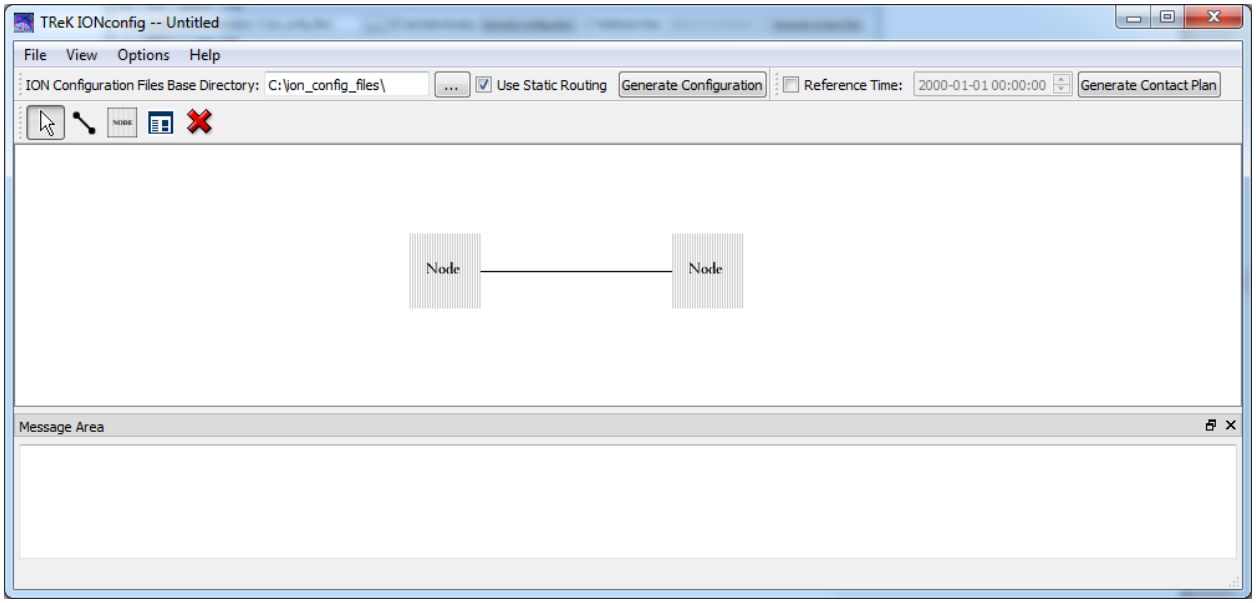

Figure 8 IONconfig with Two Connected Nodes

Step 5 – Set the properties for the first node.

Now that the diagram is complete, you will need to edit properties for each of the nodes. This tutorial will not go into detail about all of the properties available. You can read the IONconfig User Guide for properties not discussed in the tutorial. Click on the Arrow tool in the toolbar and then double click on the first node you placed in the diagram. You should see the dialog shown in Figure 9.

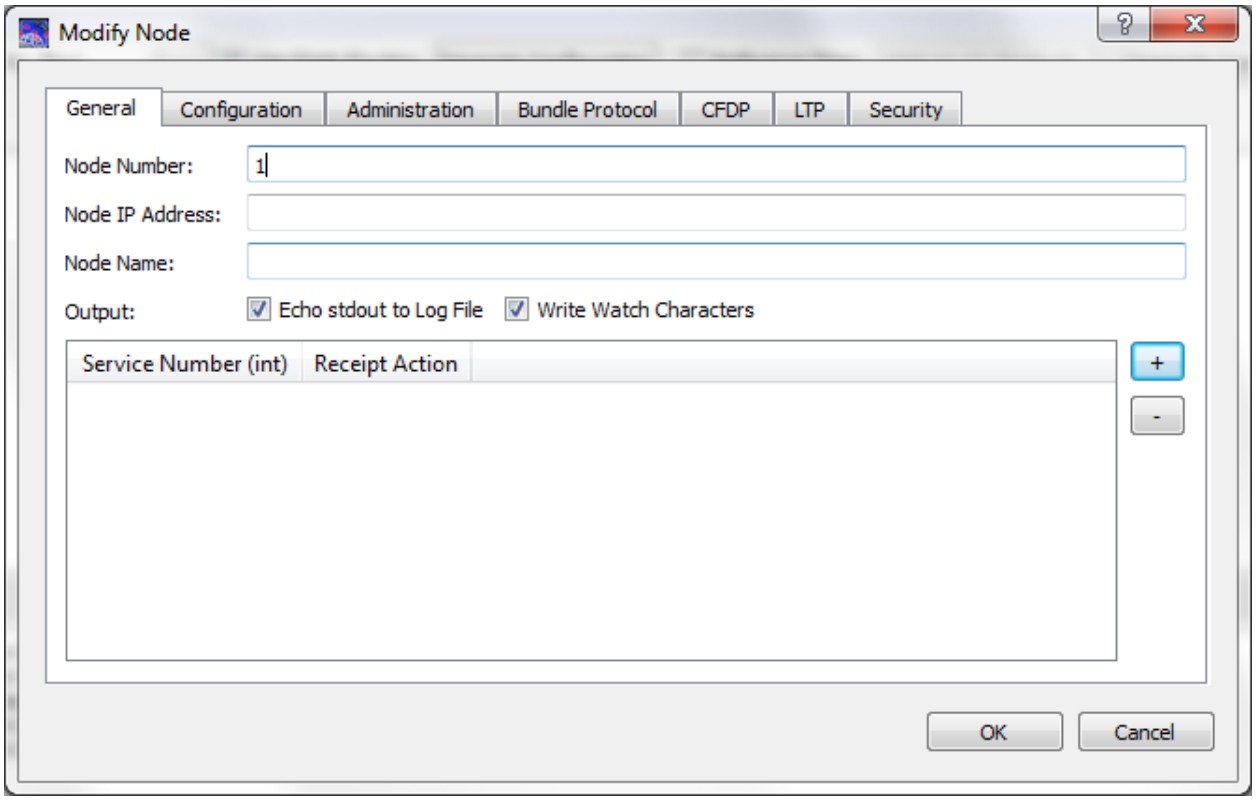

Figure 9 Node Properties

The General tab contains information that isn't necessarily tied to a single ION configuration file. The *Node Number* is what makes this node unique in the delay tolerant network. The *Node IP Address* defines the IP address to use for DTN communications.[9] The *Node Name* is a user defined name that is only used for display purposes in the diagram.[10] ION can generate watch characters that give visual indications of the underlying protocol activity. These can be very helpful if there are ever any errors transferring bundles. ION watch characters are sent to standard output and can be seen in the IONizer application. One or more services can be added for DTN enabled applications. The services needed to use CFDP are automatically added when you enable CFDP so there is no need to add the CFDP service numbers to the list.

To add a service number[11], you click the "+" button to the right of the service list. This will place a row in the service list with a default receipt action of "queue". Just type in the service number and change the receipt action if needed (the other option is "drop" which will drop the incoming bundle if a DTN enabled application isn't running to process the bundle).

[9] It is possible to use communications means other than IP for DTN, but TReK only supports generating IP configurations at this time.
[10] The nodes are not resizable at this time, so short names are best for now.
[11] You only need to add service numbers if you need to use BP other than the standard services (bpecho and CFDP).

Enter the information for the node as shown in Table 3. When you are finished, the Node properties dialog should look similar to Figure 4.

Property	**Value**
Node Number	1535
Node IP Address	Your IP address
Node Name	GRND
Services	3 (queue), 4 (drop)

Table 3 General Node Properties (Node 1535)

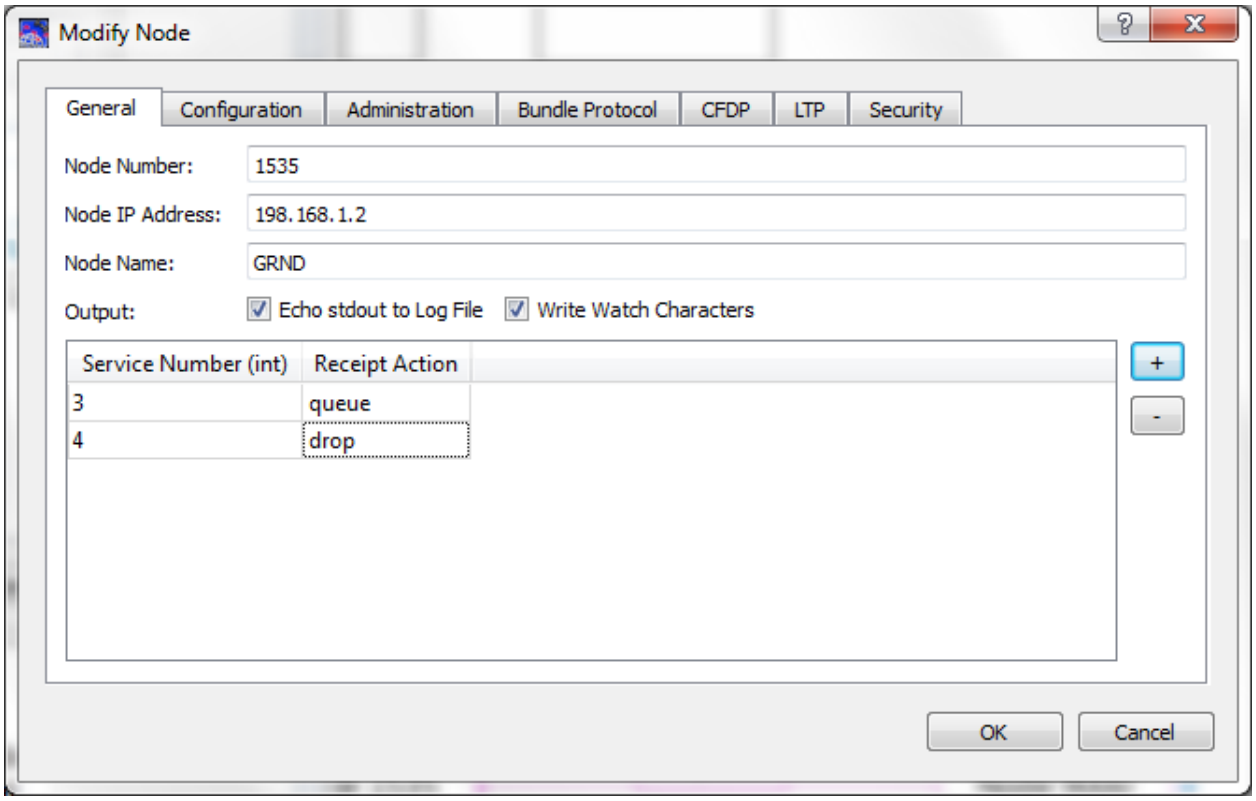

Figure 10 General Node Properties (Node 1535)

Click on the Configuration tab in the dialog as shown in Figure 11. You will need to edit some of the properties in the dialog. Only the properties that are relevant for this tutorial are described in the following paragraph. Everything else can remain with the defaults shown.

The *SDR File Path* is the directory where your data store will be located. It needs to be a directory where you have write privilege. The *Heap Words* describe how large to make the data store. ION calculates the number of bytes for the data store by multiplying the value you provide here with the number of bytes in the processor. For a 64-bit computer that will be 8x250,000 or 2 million bytes if you keep the default. You will want to

increase this value for operations. A recommendation will be provided in a later TReK release.

Note: The *SDR File Name* is the name of the data store. You should use the default value of *ion*. It is required unless you have set up a computer to run multiple nodes (not part of this tutorial-see ION documentation on how to do that). The *wmKey* must also have the default value of 65281 when using a single node per computer configuration.

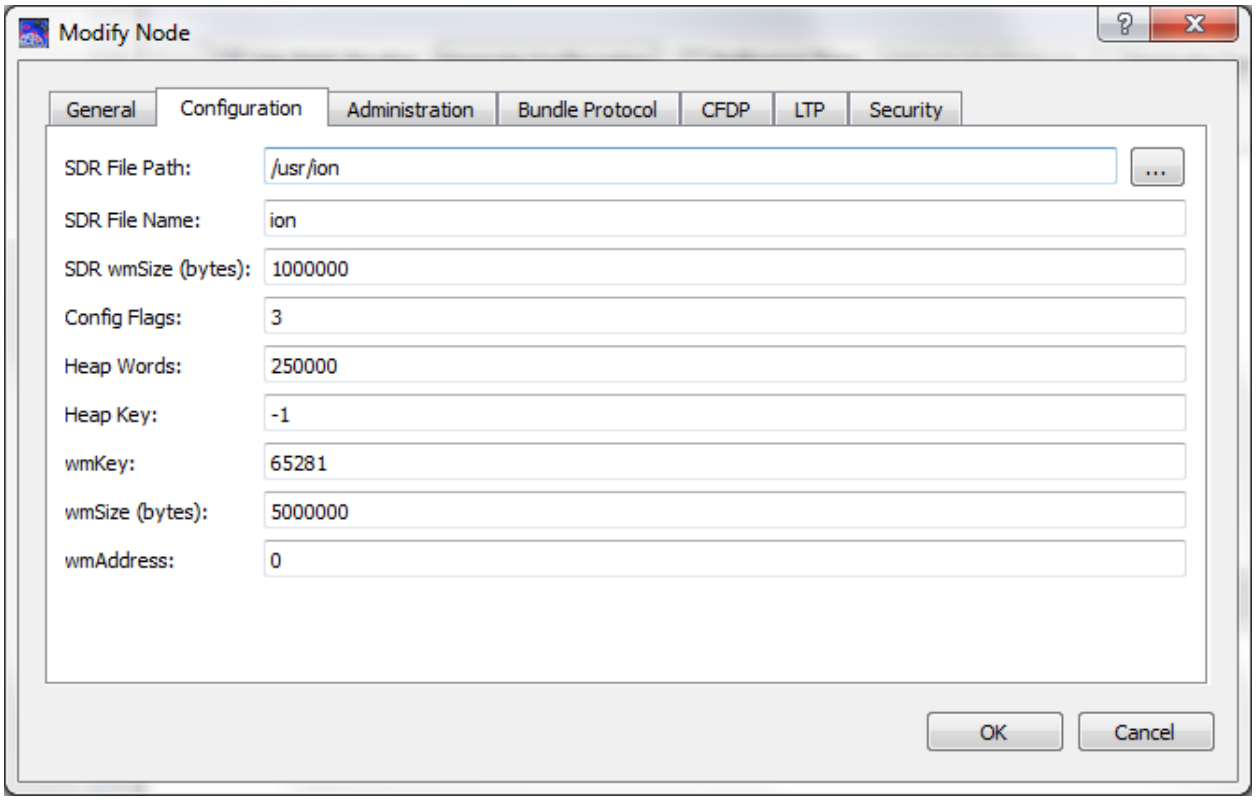

Figure 11 Configuration Properties

Enter the information for the node as shown in Table 4. Items not listed should not be changed. When you are finished, the Node properties dialog should look similar to Figure 12.

Property	**Value**
SDR File Path	Valid directory on your computer
Heap Words	1,000,000 (no commas)

Table 4 Configuration Properties (Node 1535)

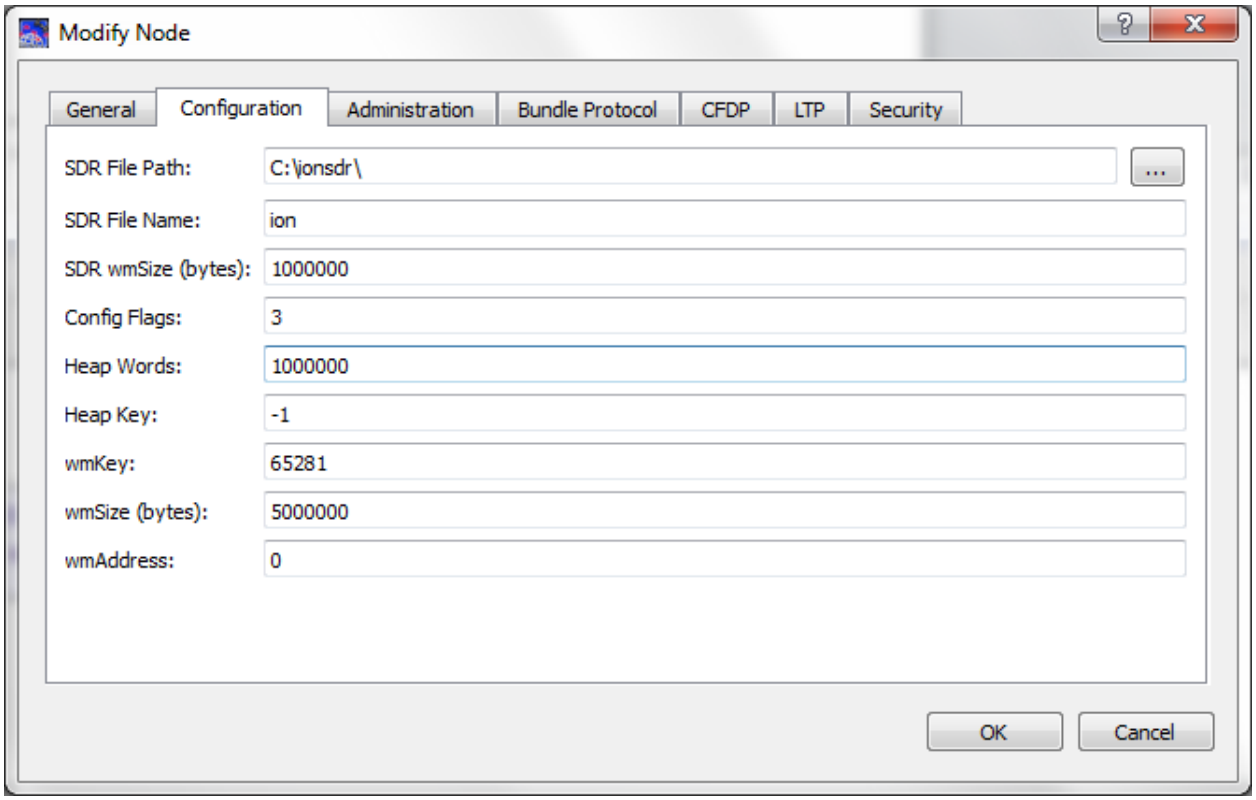

Figure 12 Configuration Properties (Node 1535)

Click on the Administration tab on the Node Properties dialog. There are no changes to make to the dialog shown in Figure 13.

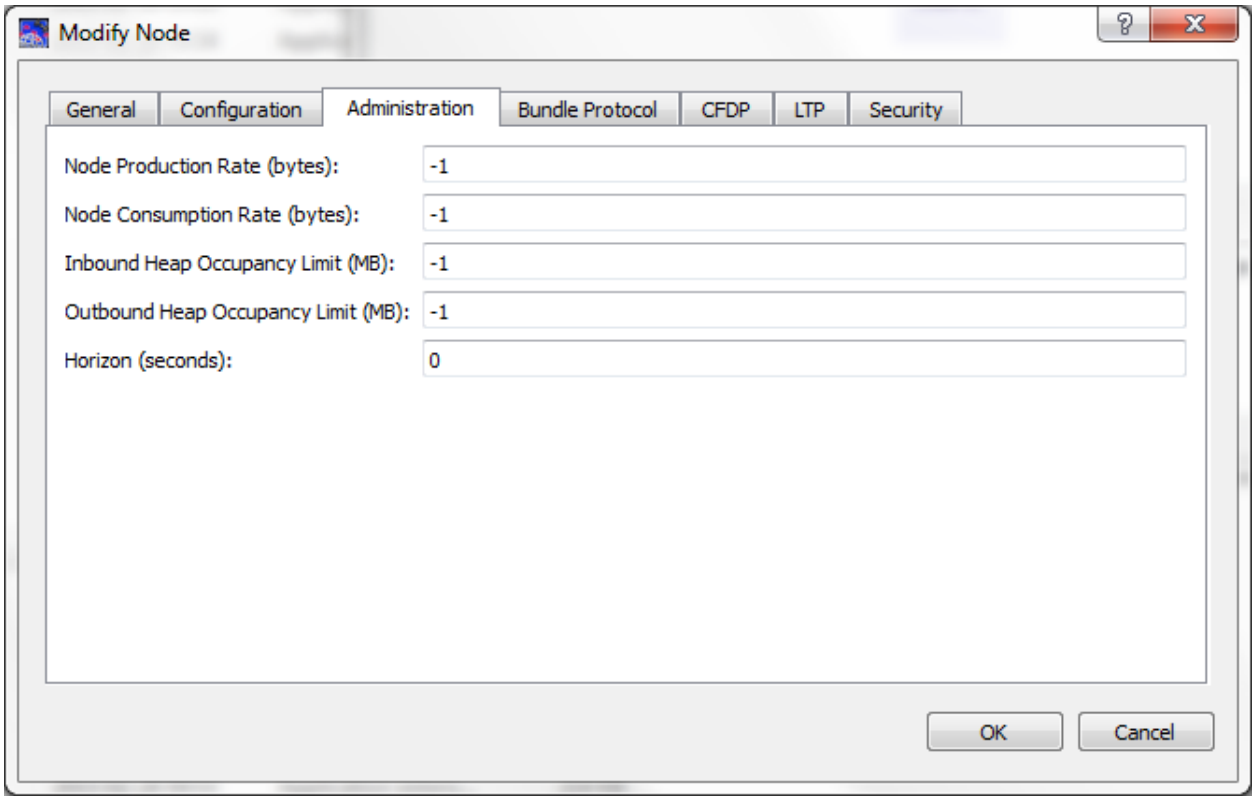

Figure 13 Administration Properties

Click on the Bundle Protocol tab of the Node Properties dialog. It will look like Figure 14. TReK can automatically start the bpecho application available in ION. This allows other ION instances to "ping" this node with the ION bping application. The echo service is on by default. You can select a different service number for the echo service. It is automatically added to the services list.[12]

There are four supported convergence layers: TCP, UDP, STCP, and LTP. The STCP convergence layer is unique to ION and cannot be used with other implementations of the bundle protocol (DTN). For each protocol you can select the number of *Payload Bytes Per Frame* and *Overhead Bytes Per Frame* that are used internally by ION to calculate estimated transmission capacity for each bundle and aid in congestion forecasting. It is probably just best to use the default values. The *Nominal Data Rate* can be used to throttle the amount of data sent from an ION node for TCP, UDP, and STCP. If you use zero the data is sent as fast as the convergence layer will allow. The nominal data rates are in bytes per second. The data rate for LTP is controlled in the contact plan.

[12] The echo service number defaults to the ISS designated service number. At this time the service number is undocumented and will change prior to operational use.

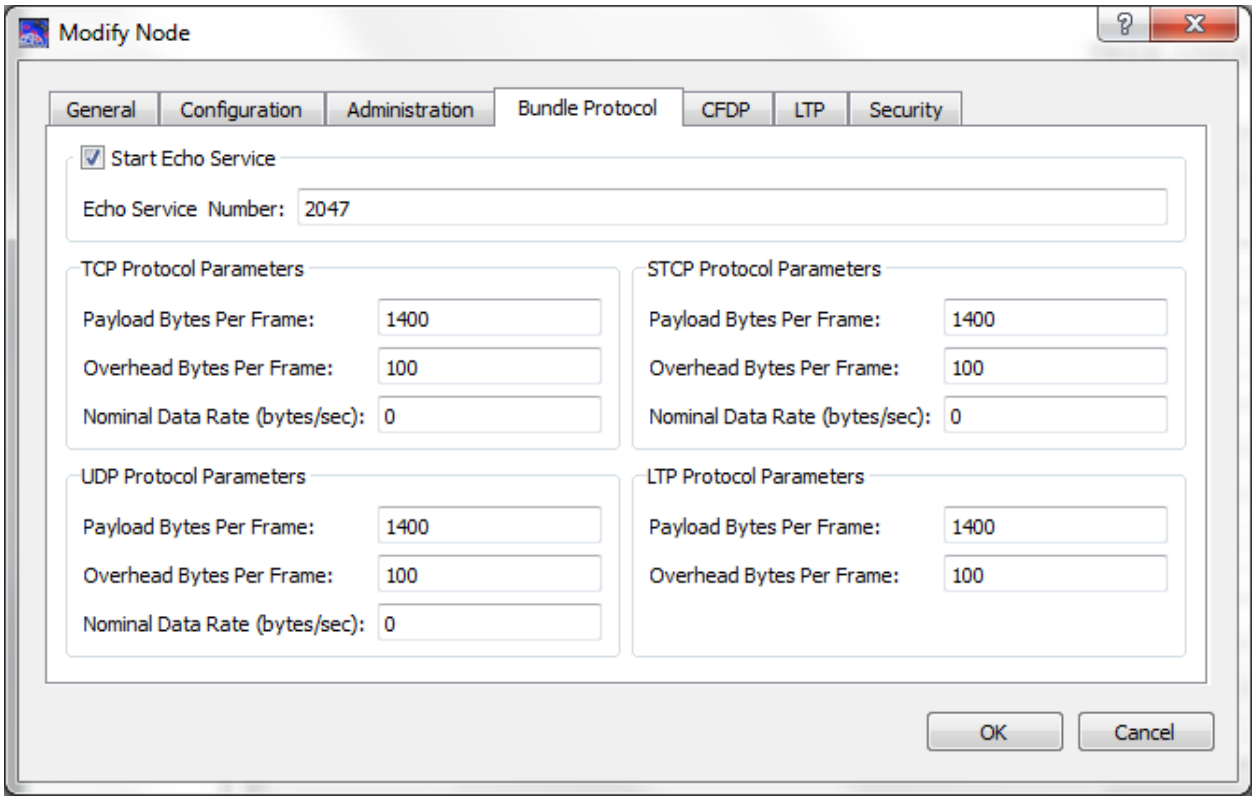

Figure 14 Bundle Protocol Properties

Enter the information for the node as shown in Table 5. Items not listed should not be changed. When you are finished, the Node properties dialog should look similar to Figure 15.

Property	**Value**
Nominal Data Rate (TCP)	50,000 (no commas)

Table 5 Bundle Protocol Properties (Node 1535)

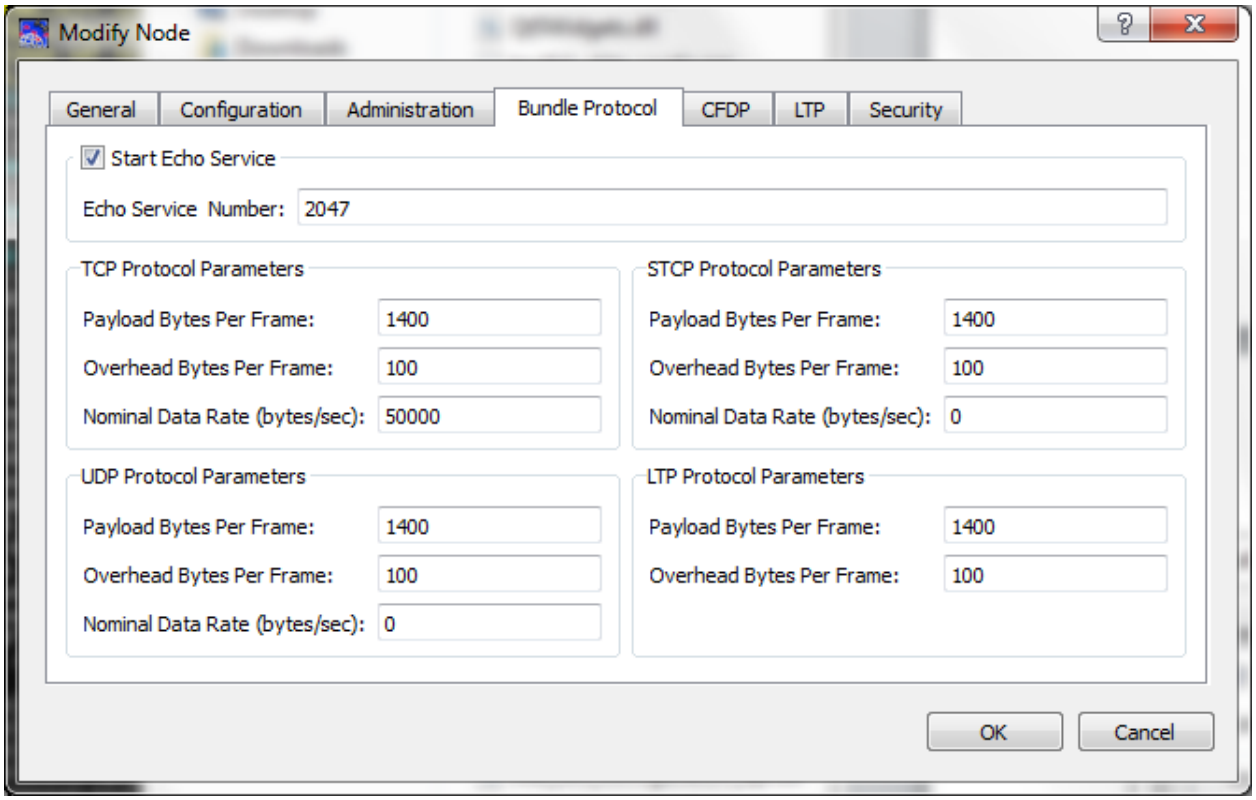

Figure 15 Bundle Protocol Properties (Node 1535)

Click on the CFDP tab on the Node Properties dialog. The dialog will look like Figure 16. There are no changes to make on this dialog, but it is worthwhile explaining a few of the fields on the dialog. The CFDP checkbox enables/disables CFDP for the ION node. If it is enabled, the service numbers (64 and 65) used by CFDP will be automatically added. The *File Fill Character* is the fill pattern used by the ION CFDP engine for missing data. The valid values are 0-255 (0x00-0xff). The *PDU Segment Size* defines how large the CFDP Protocol Data Unit (PDU) will be. It is best to leave this value unchanged. Small PDU sizes can cause performance problems with the CFDP engine. The *Inactivity Period* defines how long a file transfer can be idle before the file is canceled.

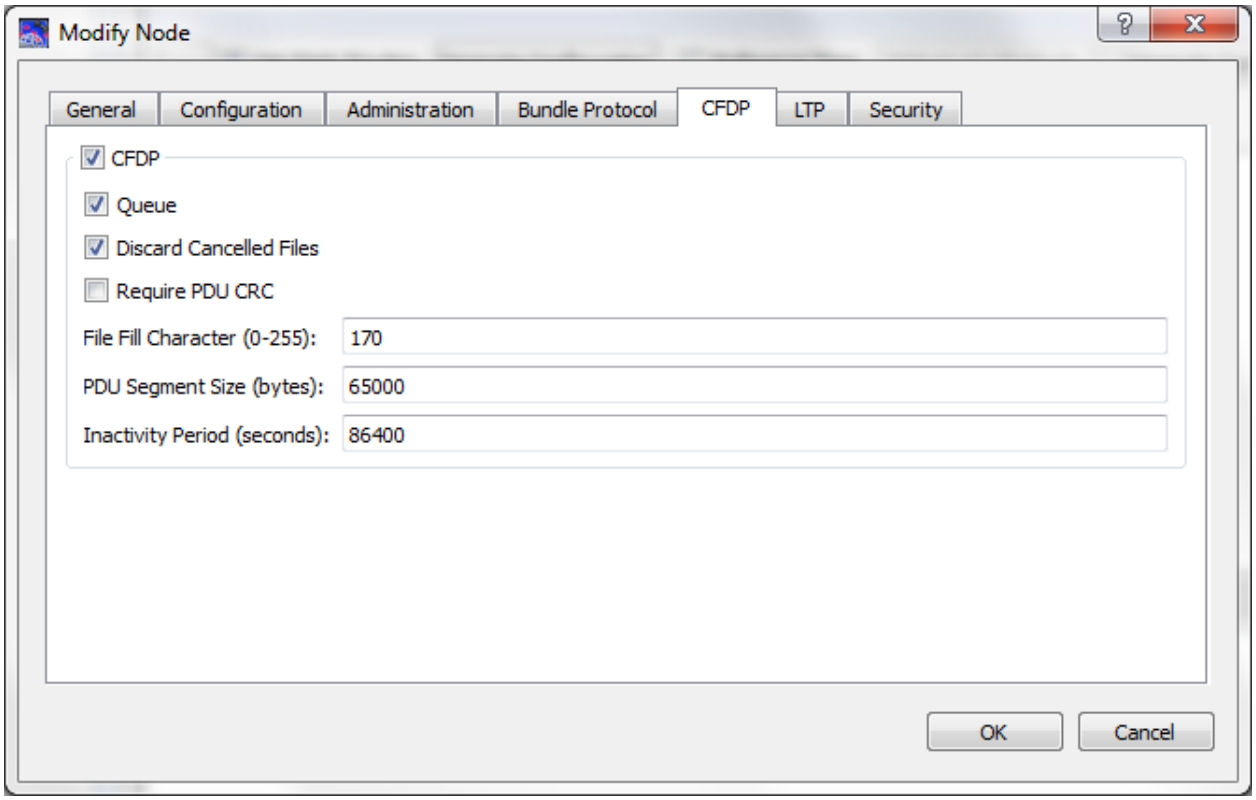

Figure 16 CFDP Properties

Click on the LTP tab on the Node Properties dialog. The dialog looks like Figure 17. There are no changes to be made on this tab. The example in Section 7.3 will use LTP.

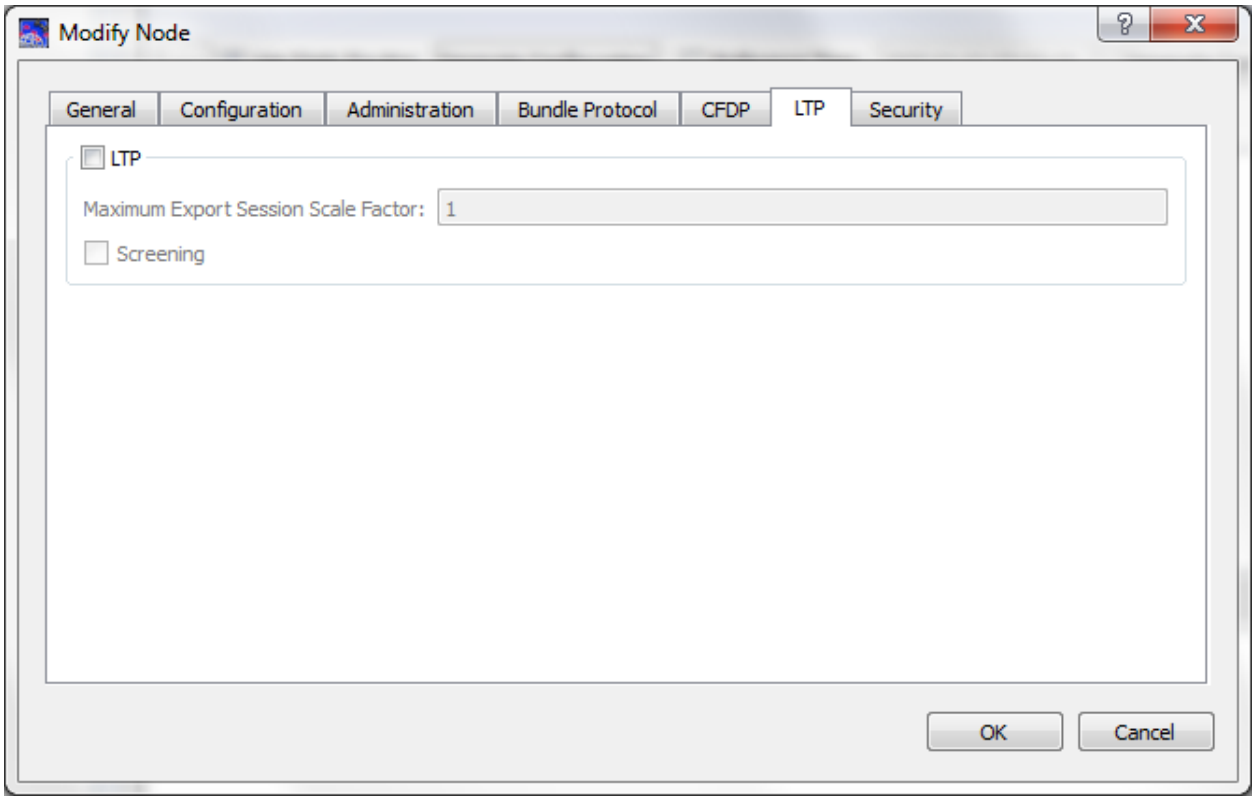

Figure 17 LTP Properties

Click on the Security tab on the Node Properties dialog. The dialog looks like Figure 18. There are no changes to be made on this tab. Turning on the security prevents unwanted messages appearing in the ION log file. Press OK to save the changes for the node.

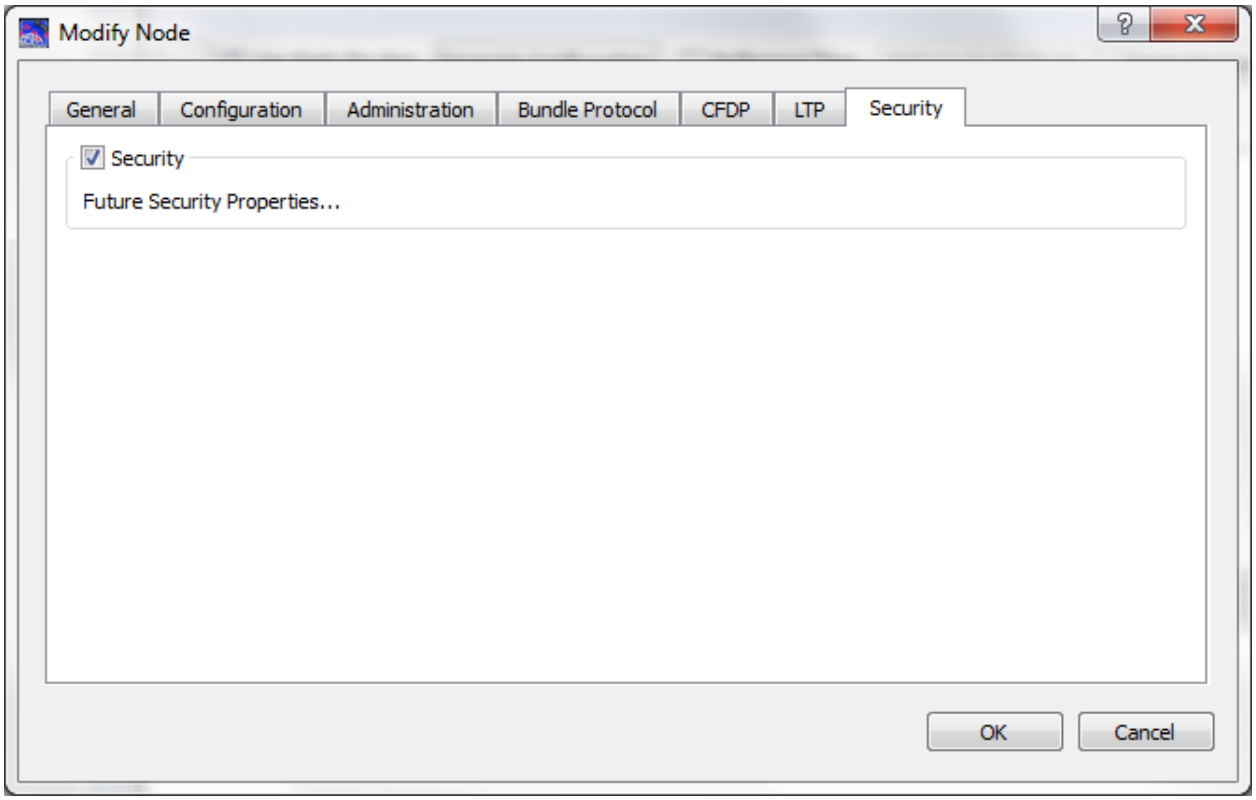

Figure 18 Security Properties

Step 6 – Set the properties for the second node.
Now you need to edit the properties of the second node. The tables that follow describe the values to use on each tab. You will need a second computer for this node, so make sure you enter that computer's IP address.

Property	**Value**
Node Number	9000
Node IP Address	Another Computer's IP address
Node Name	HAL
Services	3 (queue), 4 (drop)[13]

Table 6 General Node Properties (Node 9000)

Property	**Value**
SDR File Path	Valid directory on your computer
Heap Words	1,000,000 (no commas)

Table 7 Configuration Properties (Node 9000)

[13] The service numbers used for both computers are the same, but that isn't required.

Property	Value
Nominal Data Rate (TCP)	1,000,000 (no commas)

Table 8 Bundle Protocol Properties (Node 9000)

Step 6 – Set the properties for the connection

Double click on the connector line between the two nodes to display the dialog in Figure 19. There are two tabs in the dialog one each for each direction of data flow. For this configuration there are no changes necessary. The port numbers all default to the 'official' bundle protocol port, but you can change it as needed.[14] The *From Port* is the port associated with the first node in the tab title. The *To Port* is the port associated with the second node in the tab title.

The *Contacts* are only valid when using LTP.

[14] If you decide to run multiple nodes on a single computer, changing the port number is necessary.

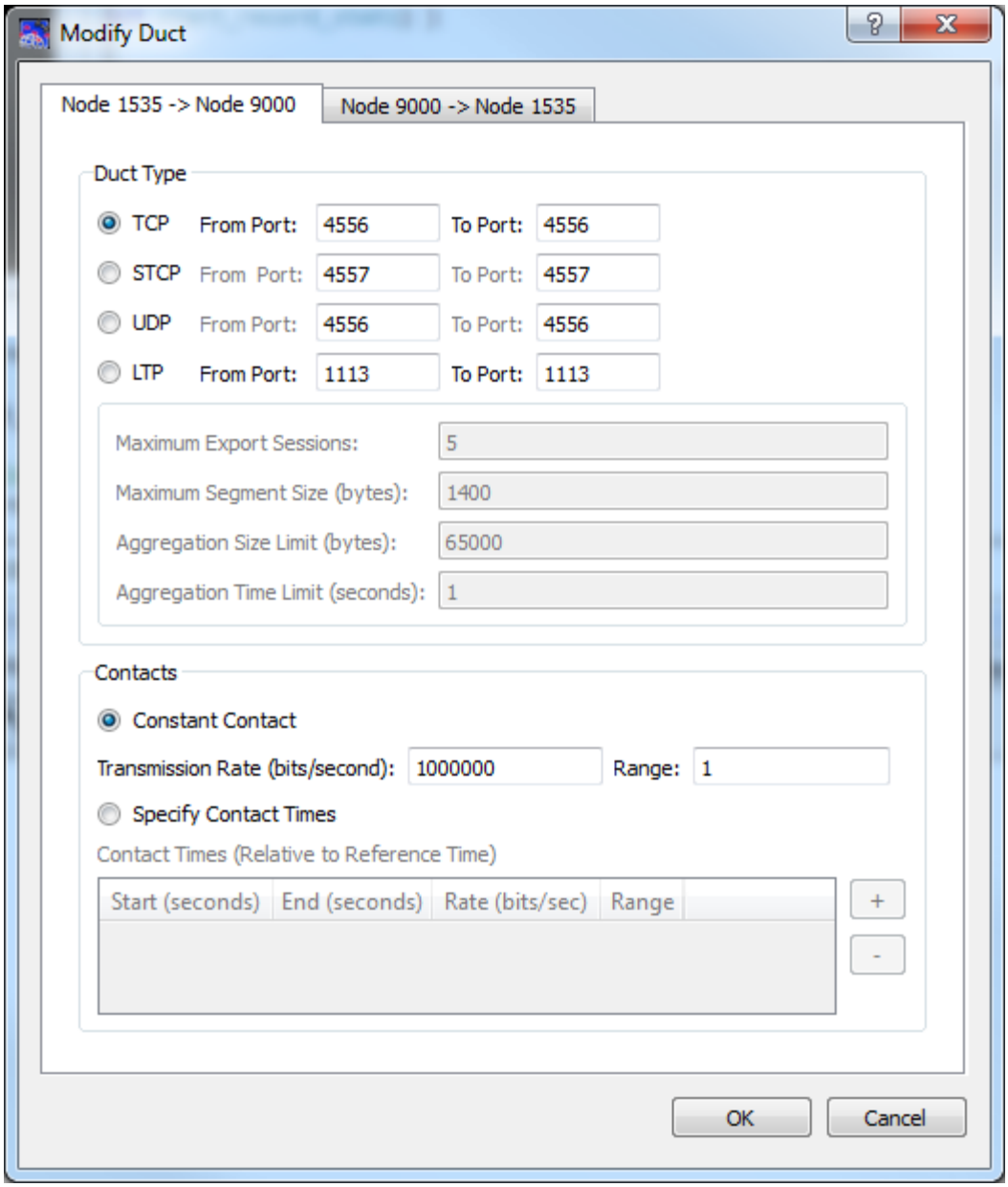

Figure 19 Duct Properties

Step 7 – Save the configuration

To save the configuration select Save from the File menu. Just select a directory that allows you write access and push the Save button. An extension isn't required for the file. The file format is XML.

Step 8 – Generate the configuration files

Press the *Generate Configuration* push button from the toolbar. This will generate all of the files and scripts needed to run ION on each node in the diagram. These files will be placed in the directory specified by *ION Files Base Directory*. An explanation of all of the files generated by IONconfig can be found in Section 4.2.

7.1.2 IONizer

The IONizer application will start an ION node with the configuration files generated by the IONconfig application in the previous section. Once the node is started, IONizer will monitor the log file for errors and report any process that crashes.[15]

Step 1 – Start IONizer

Start the IONizer application from your systems start menu. The IONizer application shown in Figure 20 will start. The main window of IONizer contains a toolbar with the options we will use in the following steps. The main window contains two tabs. The Monitor tab will show messages written by ION and the TReK Proxy when applicable. The Output tab is the messages written by ION to standard output as well as any watch characters generated during ION operations.

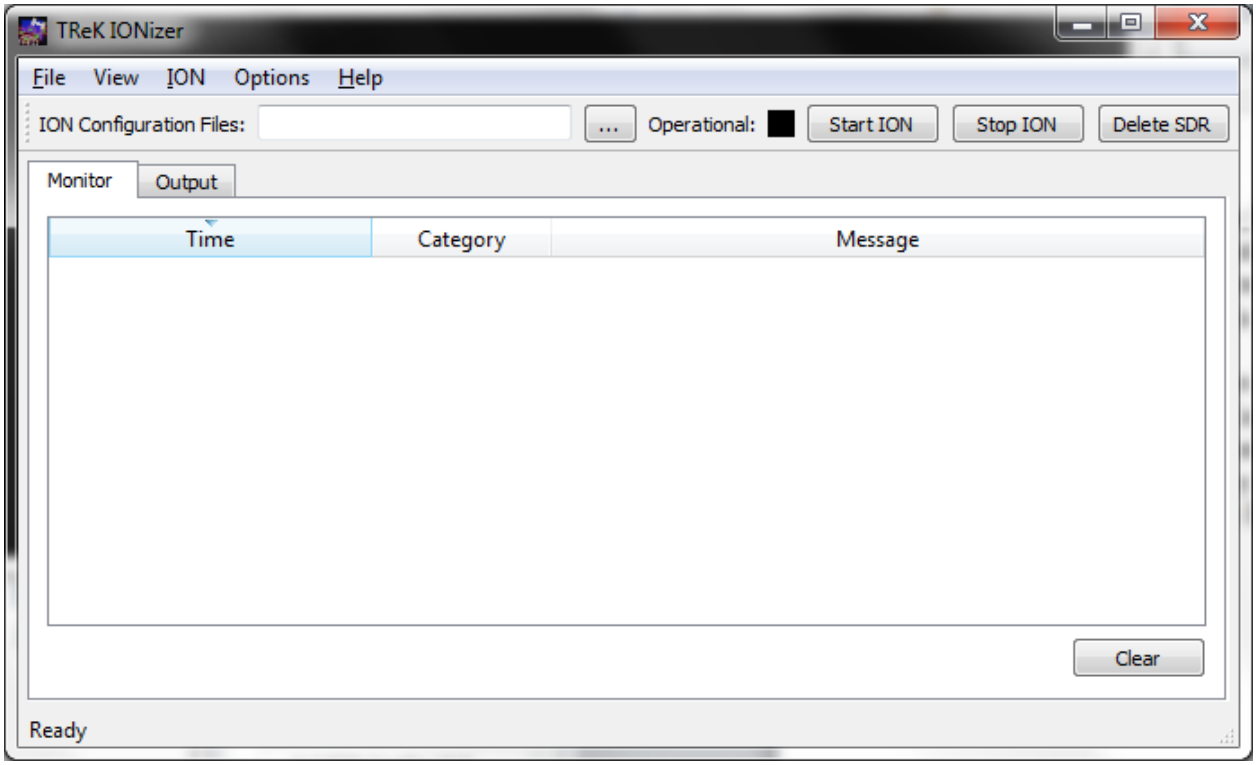

Figure 20 IONizer Main Window

[15] Additional monitoring capabilities are anticipated in future releases.

Step 2 – Select the ION configuration file directory

In the toolbar enter the *ION Configuration Files* directory in the text field or select the directory using the button to the right of the text field. You should select the node_1535 directory you used in the IONconfig setup. Your main window should be similar to Figure 21.

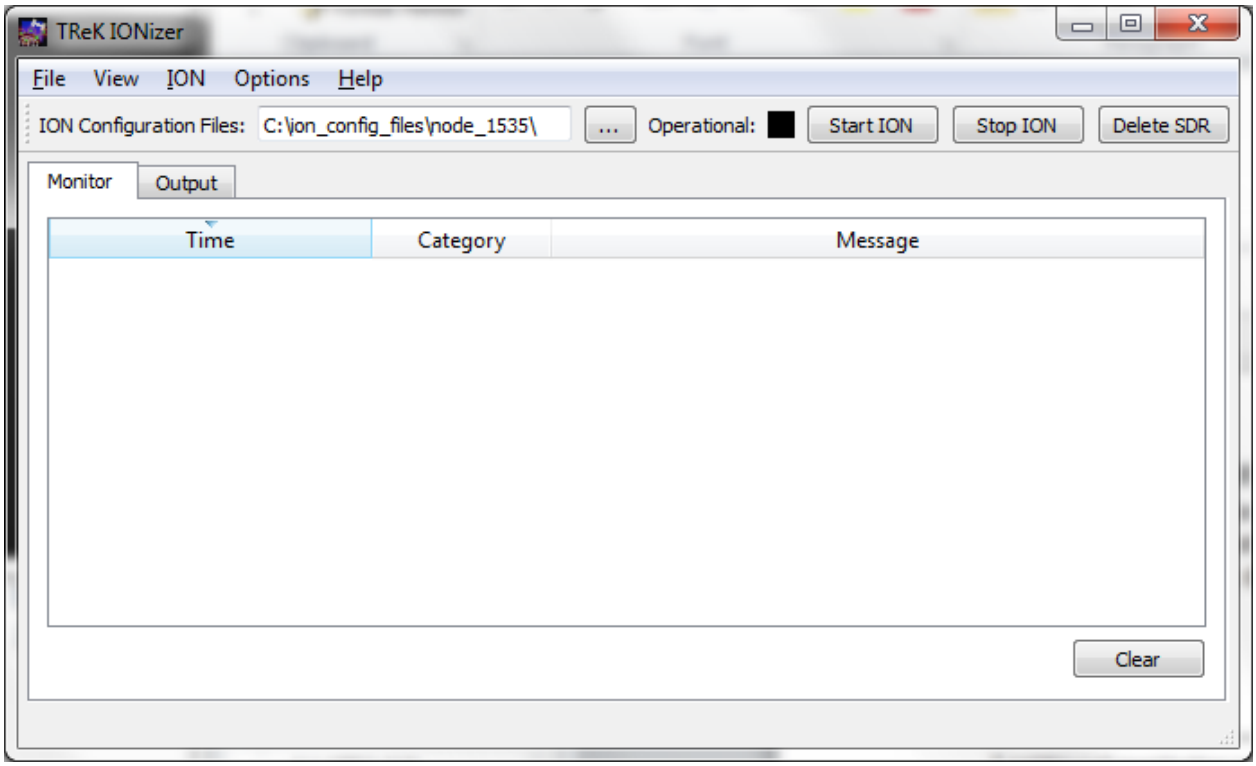

Figure 21 IONizer with Config File Directory

Step 3 – Start ION

Once the configuration file directory is selected, push the *Start ION* button in the toolbar[16]. This will execute the startion.bat file (or startion.sh if you are running on Linux). The ION messages will appear in the main window. If there are errors starting ION this is where you will be notified of the problem and possibly the cause (*note: start errors could also appear on the output tab)*. The *Operational* status will change to green when ION starts. A typical output for starting ION is shown in Figure 22.

[16] ION will make network connections during startup. You may need to configure your firewall to allow the ION processes to access the network.

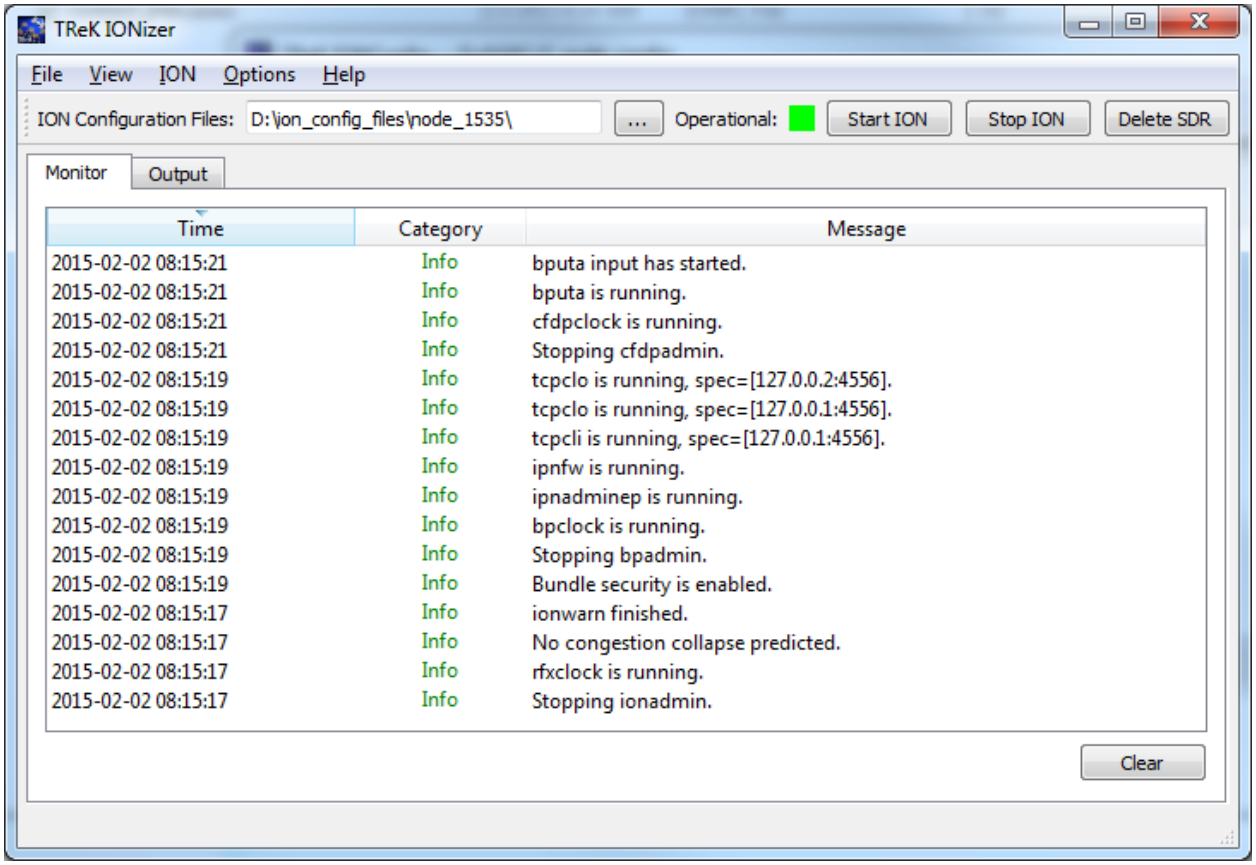

Figure 22 IONizer Main Window After Staring ION

Transfer the files generated by IONconfig for node 9000 to another computer and use the steps in this section to start ION on the other computer. If you want to run ION as a command line interface instead of using IONizer, use the startion.bat (Windows) or startion.sh (Linux) script.

7.1.3 CFDP

This part of the tutorial shows how to transfer a file using CFDP over Bundle Protocol.

Step 1 – Start CFDP

Start the CFDP application from your systems start menu. The CFDP application shown in Figure 23 will start.

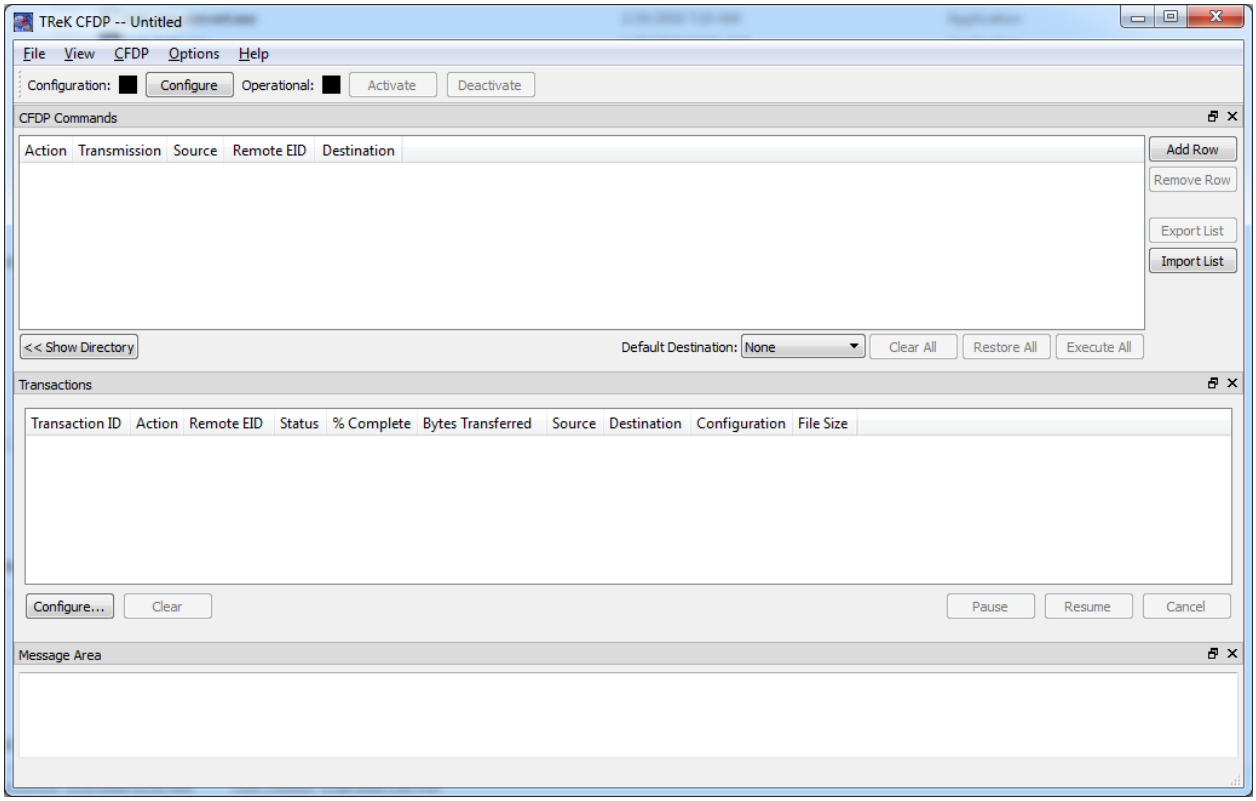

Figure 23 CFDP Application Main Window

Step 2 – Configure the CFDP Application

Push the Configure button in the toolbar. The dialog shown in Figure 24 will be displayed. Change the Configuration from Native CFDP to ION CFDP. The dialog will now look like Figure 25.

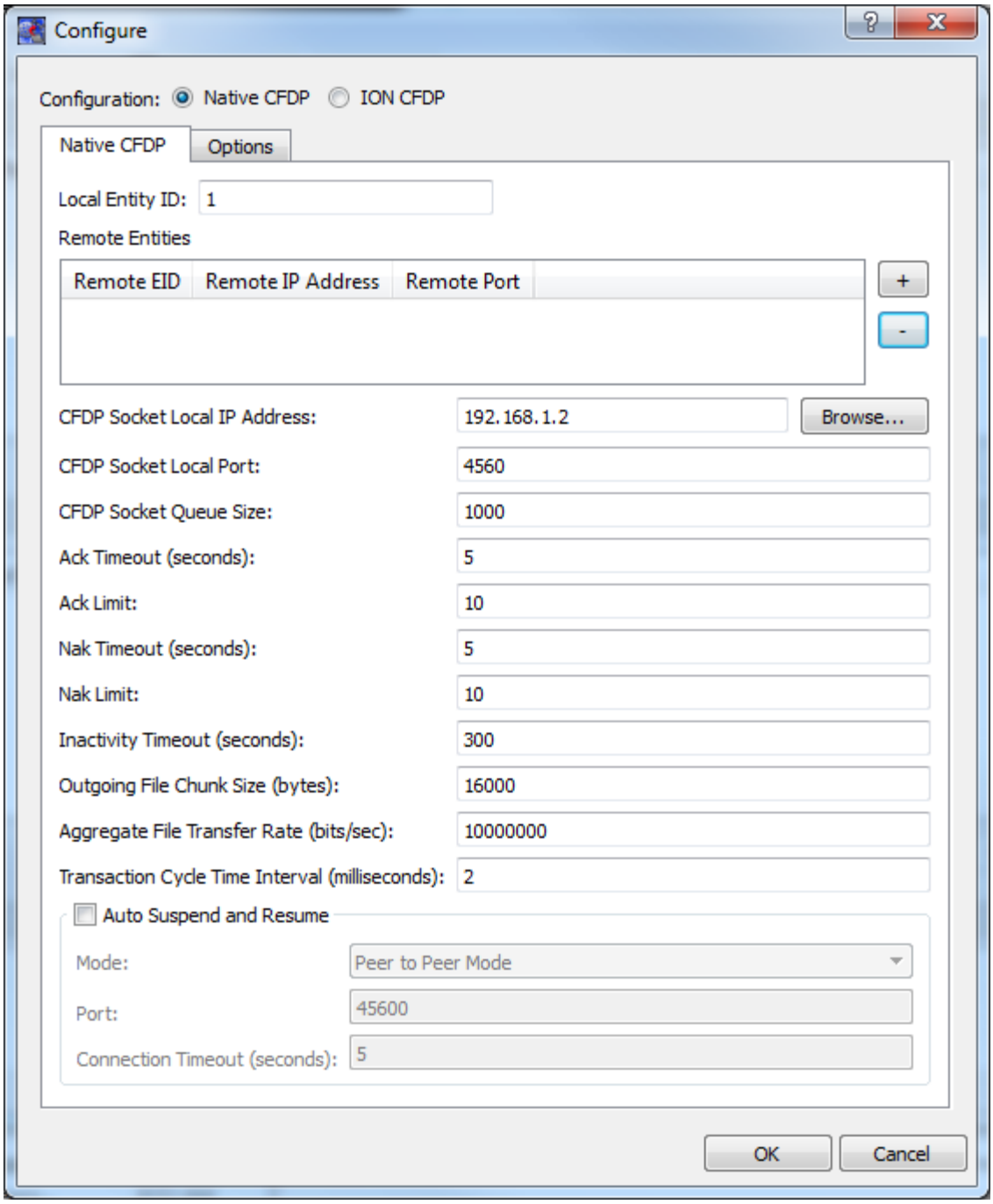

Figure 24 CFDP Configure Dialog

Step 3 – Configure for ION CFDP

In the Configure dialog, select the ION CFDP radio button as shown in Figure 25.

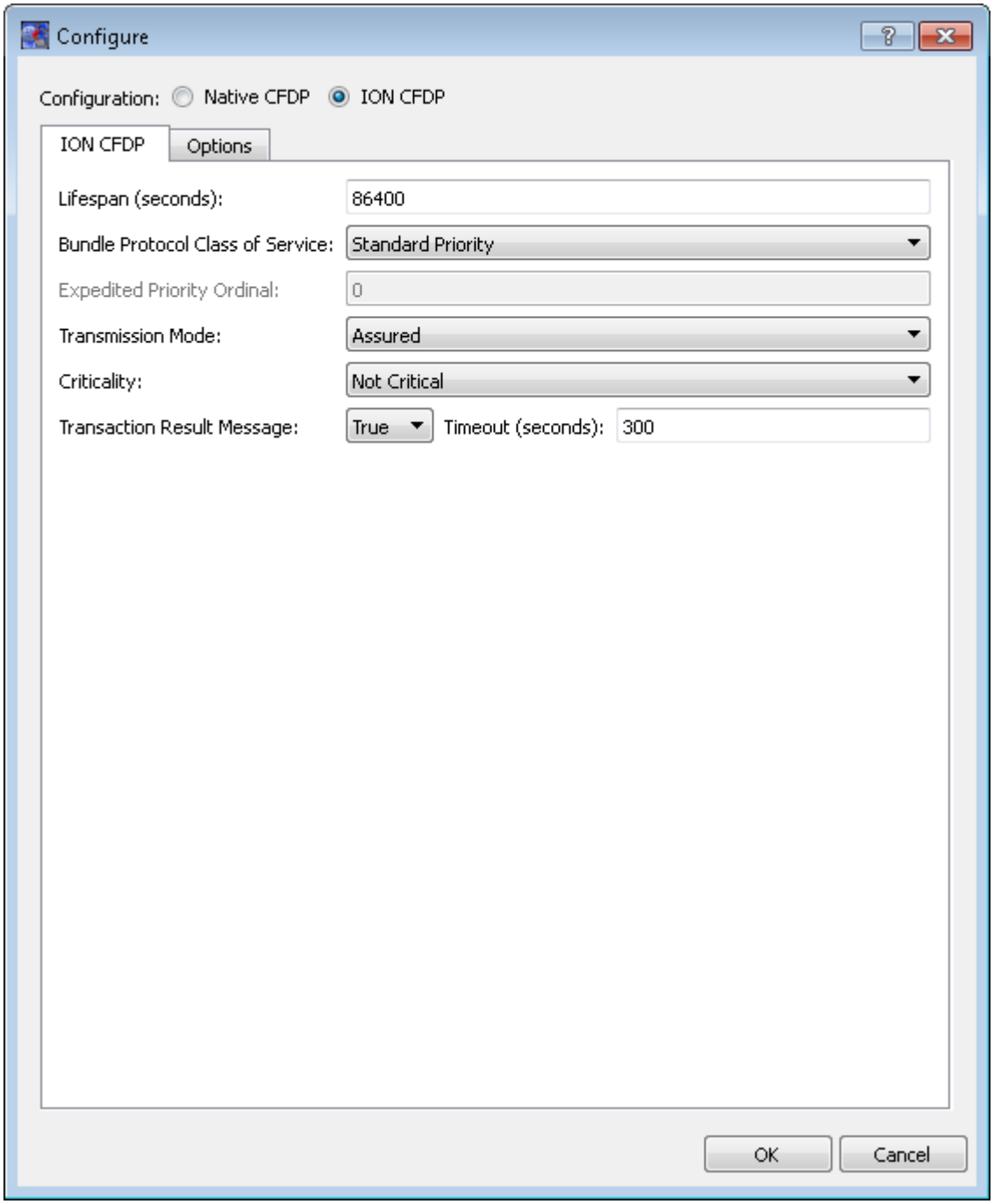

Figure 25 CFDP with ION CFDP Mode Selected

The CFDP over BP Transmission Properties are described below. In most cases, there is no need to change the defaults.

Property	Description
Lifespan	The lifespan is the bundle's "time to live" (TTL) in seconds. The bundle is destroyed if it's TTL has expired and it has not reached its destination.
Bundle Protocol Class of Service	The Bundle Protocol Class of Service defines the transmission priority of outbound bundles from three ION priority queues corresponding to bulk, standard, and expedited priorities. The expedited priority queue must be empty before bundles in the standard or bulk queues are serviced by ION. Therefore, bundles with expedited priority should only be sent in critical/emergency situations.
Expedited Priority Ordinal	The expedited priority ordinal is only associated with the expedited priority class of service.
Transmission Mode	The transmission mode defines the reliability of bundle delivery to a destination. The three transmission modes supported are best effort, assured, and assured with custody transfer.
Criticality	A critical bundle is one that has to reach its destination as soon as is physically possible. For this reason, bundles flagged as critical may not include custody transfer and require an ION configuration with contact graph routing. In some cases, a critical bundle may be sent over multiple routes to ensure delivery to its final destination. Critical bundles are placed in the expedited priority queue and should only be used in emergency situations.
Transaction Result Message	Provides support for feedback to the sending node of the CFDP transaction. This feature requires that the TReK CFDP library/application be running on both the sending and receiving nodes.

Table 9 Configuration Parameters for ION CFDP

Step 4 – Configure Optional CFDP Parameters

In the Configure dialog, select the Options tab as shown in Figure 26.

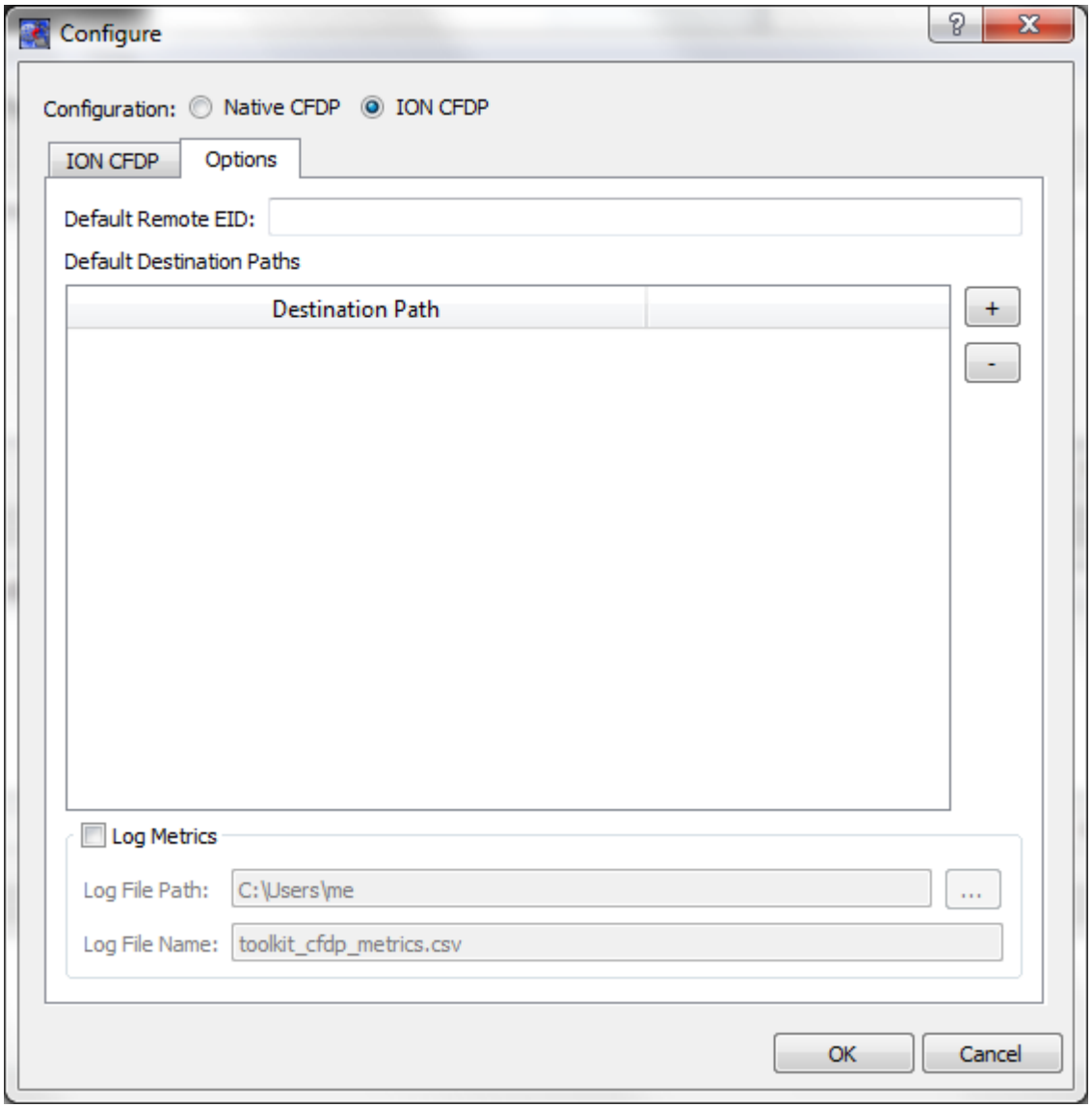

Figure 26 CFDP Configure Dialog Options Tab

The options tab allows you to create defaults that can be used with the CFDP application to make sending files easier. The *Default Remote EID* is used to determine which destination to use when transferring files. For ION CFDP the EID, or entity ID, is the same as the node number. Type in 9000 for the *Default Entity ID*. You can add one or more directories on the remote computer as well. Use the "+" button to the right of the list to add a row. Just type a directory path in the field. An example of a completed dialog is shown in Figure 27.

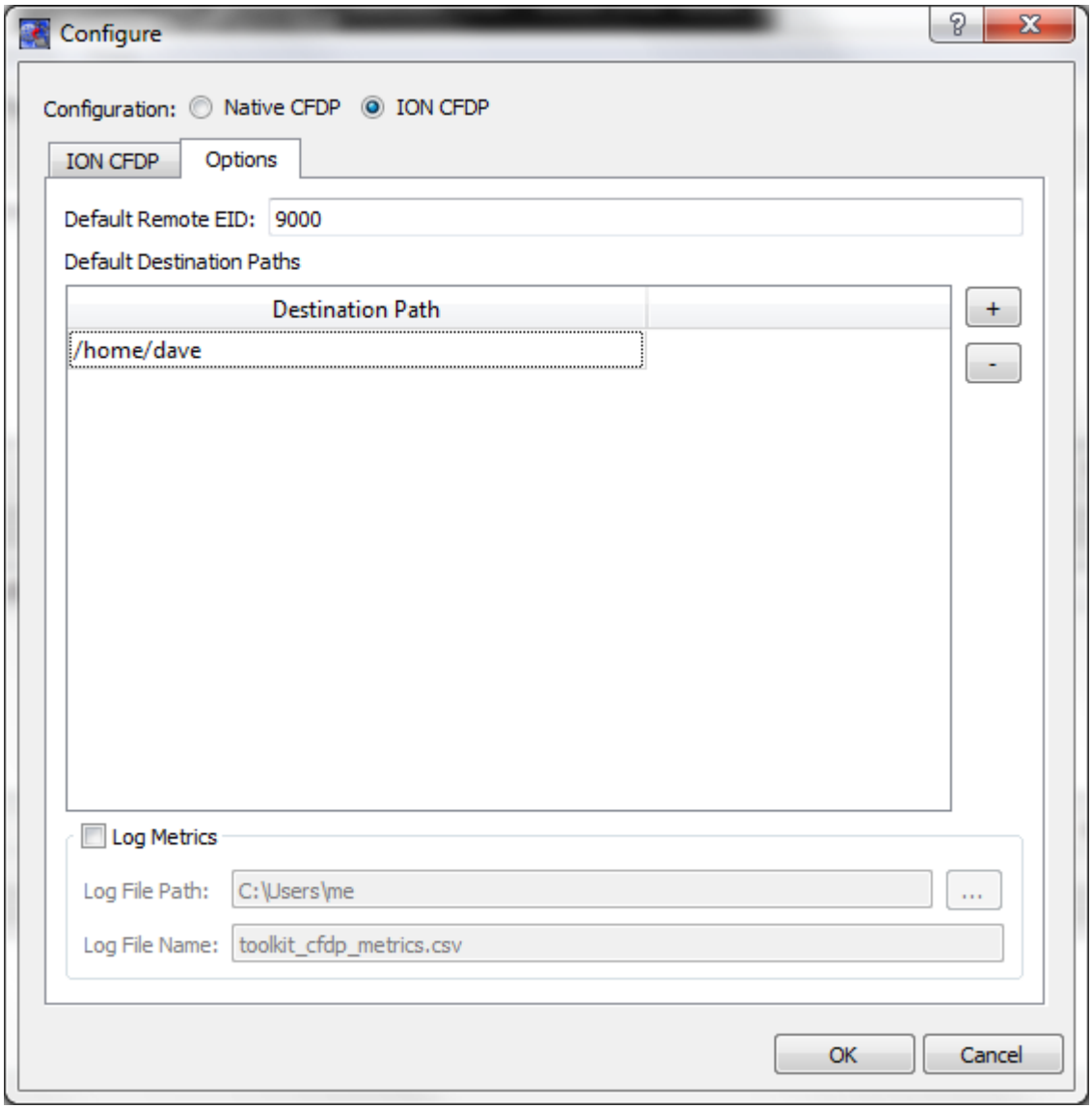

Figure 27 CFDP Configure Dialog Options Tab with Data

Push the OK button to save the configuration.

Step 5 – Activate the CFDP service.
Before you perform this step, ION needs to be running. During activation, the CFDP application will attemp to connect to ION. If ION is not running, activation will fail and you will see messages similar to the following:

> Failed to activate CFDP service.
> Error: Failed bp_attach. (bp_attach_device.cpp:116)
> Error: Failed to create a library device. Pathname: trek_bp_device_api.dll (device_manager.cpp:1158)

If ION is running, push the Activate button in the CFDP Main Window. Your main window should look like the one shown in Figure 28.

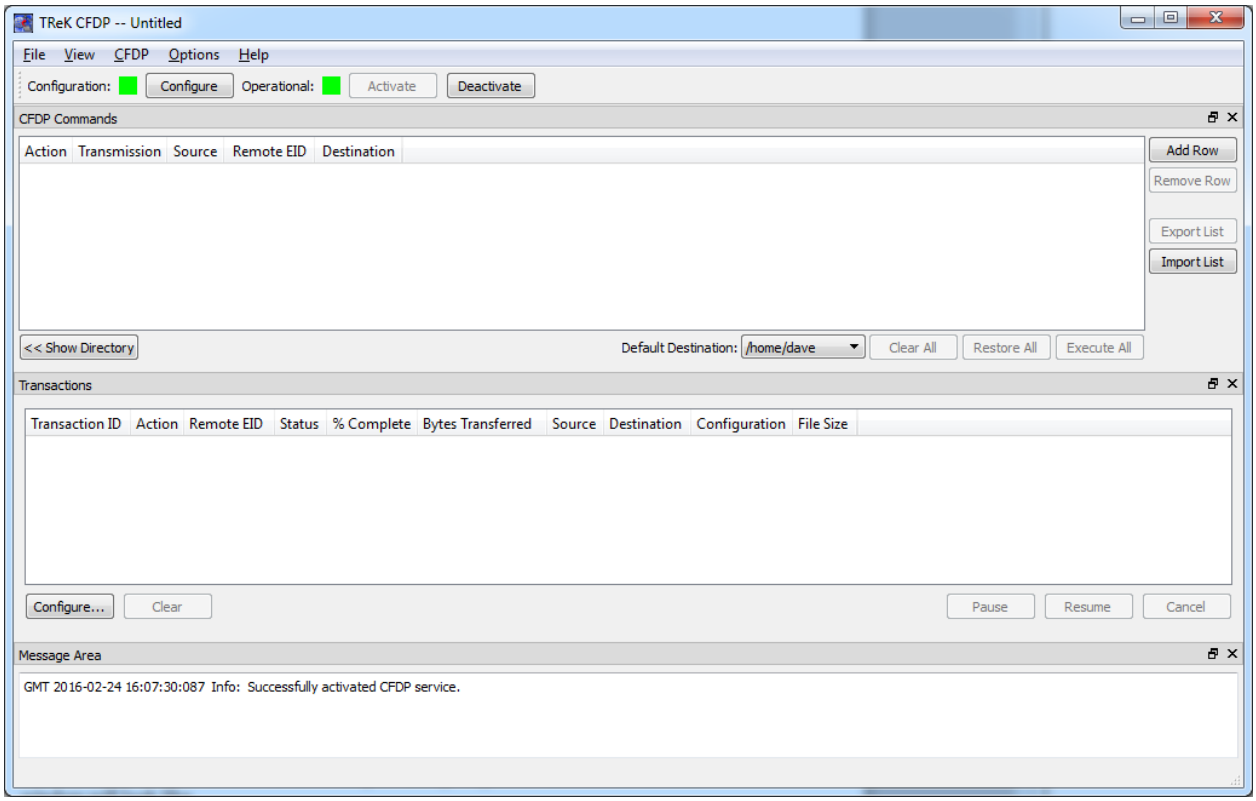

Figure 28 CFDP Active in ION CFDP Mode.

Step 6 – Transfer a File

At this point ION should be running on both computers. Using an explorer window, select a file and drag and drop it into the CFDP Main Window. The CFDP Main Window should look similar to the one shown in Figure 29.

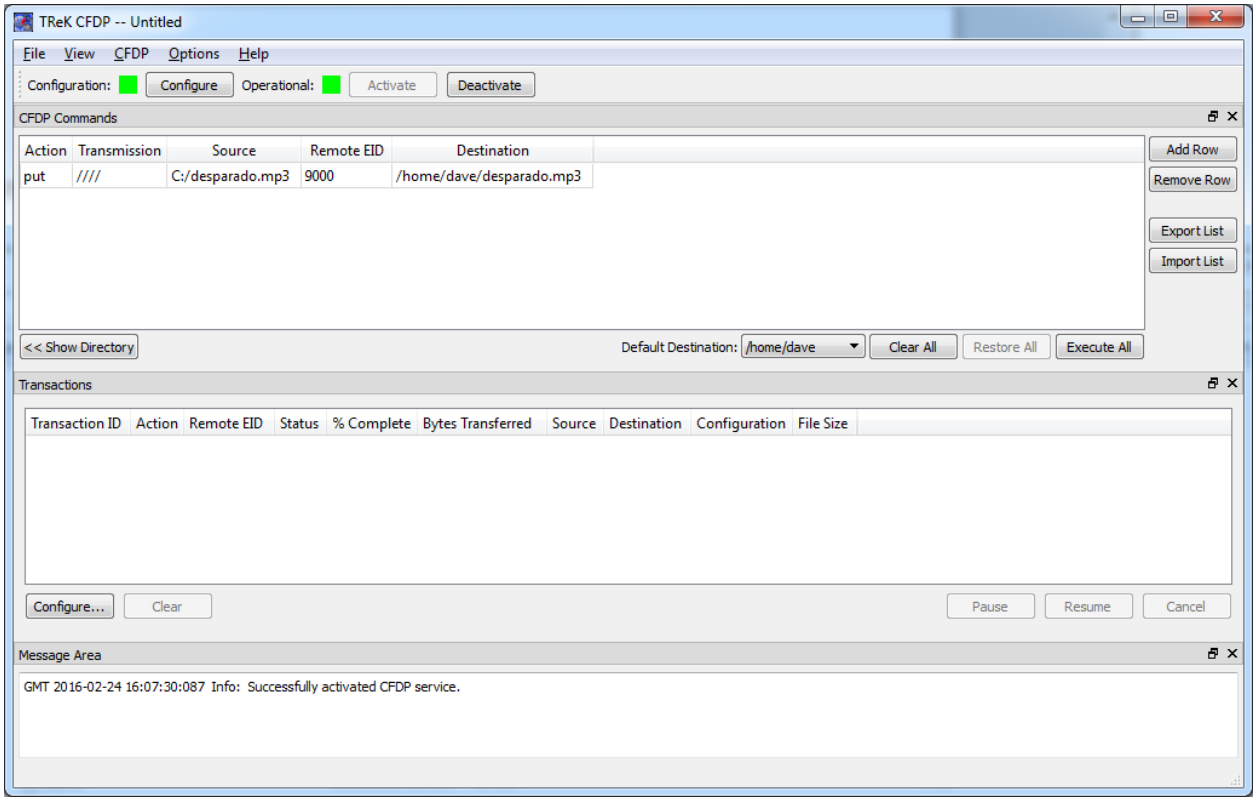

Figure 29 CFDP Main Window with Complete File Transfer Information

Push the Execute all button to transfer the file. The Transactions area will display a message showing the status of the transaction. If the transfer is successful, the transaction will be green and you will know that the file was successfully transferred to your second computer.

Note: For the transaction to be marked as successful, a TReK CFDP application will need to be running on the receiving node.

7.2 Four Nodes ISS

For ISS operations a four node DTN configuration will be used. The payload teams will be responsible for each of the endpoint nodes. The other two nodes are provided by the ISS program. Figure 30 shows the four nodes used for ISS payloads. The payload team is responsible for configuring the two endpoints: Payload Ground and Payload Flight. The other two nodes are provided by the ISS program.[17] The interface for the payload to each gateway is TCP.[18] The gateways will communicate with each other with LTP over UDP.

[17] Note that the HOSC node (ground gateway) is implemented using dtn2 instead of ION. You can still use the IONconfig application to generate the files for the ION nodes.
[18] The ground interface is currently defined as TCP in SSP 50305 Volume 2. It is anticipated that the onboard gateway will use TCP as well, but it is not defined in an official document at this time.

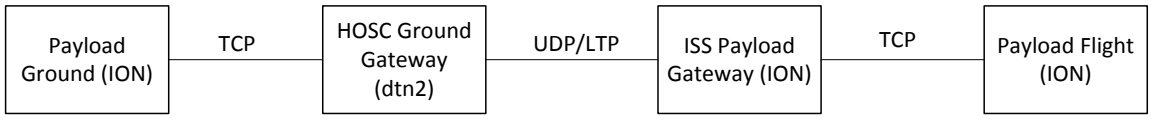

Figure 30 Four Node ISS Example

Step 1 – Copy the 4-node ISS template file

A template file is available in the TReK installation that can be used as a starting point for configuring the payload ground and flight ION nodes. Copy the file to a directory where you have write access. The file is named 4_node_iss.xml and is located in the template/ionconfig directory in the TReK installation.

Step 2 – Start IONconfig and open the template file

Start the IONconfig application from your systems start menu. From the File menu, select Open and browse to where you put the template file in the previous step. Select the file and press the Open button on the dialog. Your IONconfig application should look similar to Figure 31.

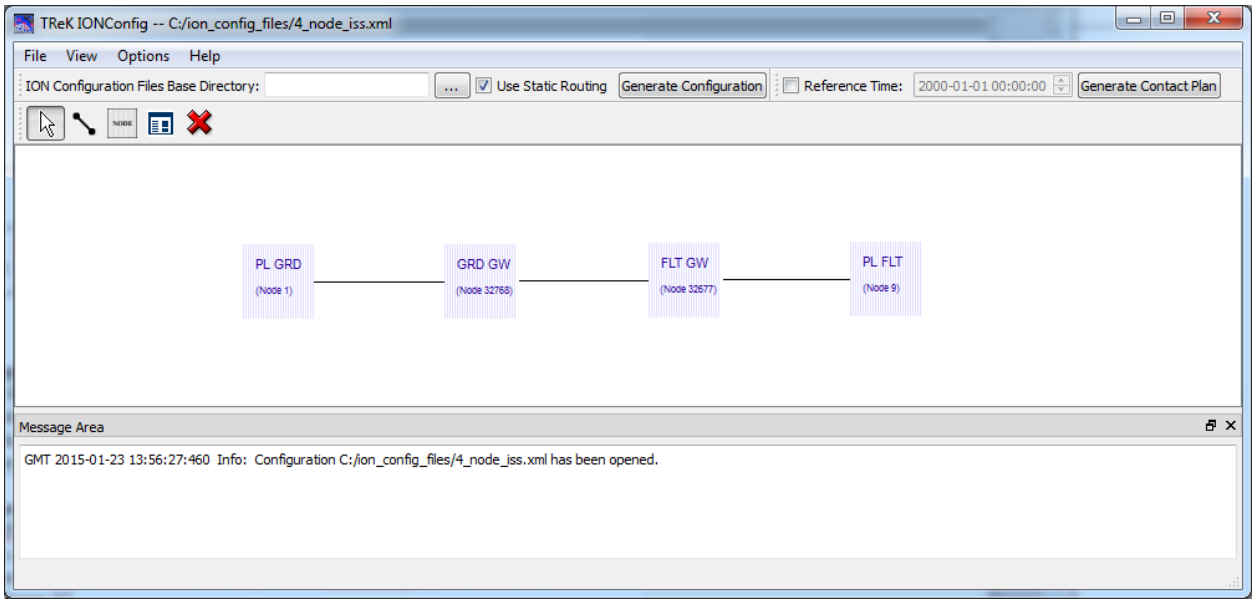

Figure 31 IONconfig 4-node ISS

Step 3 – Edit the payload ground node

The payload ground node (PL GRD in the diagram) has defaults for as much information as possible, but you will still need to edit and update some fields. The information that should be edited appears in Table 10. The first column lists the tab to select and attribute to edit. If you are writing BP enabled applications, you will also need to add the service numbers for those applications to the General tab.

Tab/Attribute	Value
General/Node Number	The node number should be set to the value assigned to your ground node. It is possible to have multiple ground node numbers assigned.[19]
General/Node IP Address	You should select the IP address of the computer that will be used for this node. During operations the TReK Proxy will be used to configure this IP address as needed.[20]
Configuration/SDR File Path	This value should be a directory on the computer that will be used for this node where you have write access.
Configuration/Heap Words	The default value of 250,000 will generate a data store (SDR) of 2 million bytes on a 64-bit computer. It is suggested that you increase this size to at least TBD.

Table 10 Payload Ground Changes

Step 4 – Edit the payload flight node

The payload flight node (PL FLT in the diagram) has defaults for as much information as possible, but you will still need to edit and update some fields. The information that should be edited appears in Table 11. The first column lists the tab to select and attribute to edit.

[19] When logging into the HOSC with the HPEG application you may be presented with a list of ground node numbers. You must choose the one that is used in the configuration files generated by IONconfig or the ION node will not start.

[20] At this time the IP address of your computer when connecting to the HOSC may not be the same as the one assigned to your network card. The TReK Proxy will hide this problem from you.

Tab/Attribute	Value
General/Node Number	The node number should be set to the value assigned to your flight node.
General/Node IP Address	You should select the assigned onboard IP address of the computer that will be used for this node.[21]
Configuration/SDR File Path	This value should be a directory on the computer that will be used for this node where you have write access.
Configuration/Heap Words	The default value of 250,000 will generate a data store (SDR) of 2 million bytes on a 64-bit computer. It is suggested that you increase this size to at least TBD.
Bundle Protocol/TCP Nominal Data Rate:	The default value (zero) will send the data as fast as the TCP convergence layer will allow. It may be necessary to throttle the rate at which the data is generated to prevent available data storage from being exceeded on the payload gateway node.[22]

Table 11 Payload Flight Changes

Step 5 – Generate the configuration files
Fill in the *ION Files Base Directory* text field in the toolbar with a directory to save the configuration files to or select a directory by pushing the button to the right of the text field.

Press the *Generate Configuration* push button from the toolbar. This will generate all of the files and scripts needed to run ION on each node in the diagram. These files will be placed in the directory specified by *ION Files Base Directory*. An explanation of all of the files generated by IONconfig can be found in Section 4.2.

Step 6 – Copy the configuration files to the ground and flight computers
Copy the files generated to the appropriate computer.

The generated configuration files can be used with IONizer as shown in the previous example. You will need to start the HPEG application to log into the POIC prior to starting ION. The HPEG User Guide provides a step-by-step tutorial for this and includes documentation on running the TReK provided ERIS Simulator if the POIC is not available.

7.3 Two Nodes with LTP

Figure 32 shows the same two node DTN configuration used in the first example except that LTP over UDP will be used instead of TCP as the convergence layer. This configuration can be used to connect a flight and ground computer directly over an IP

[21] The flight node number and flight computer IP address are assigned when the Payload to ISS ICD is written.
[22] The need to throttle data rates is anticipated in the future. The rate for each payload will likely vary with agreements reached between the payload and ISS program.

enabled space link. LTP can also run directly over a link layer protocol (e.g., AOS[23]), but ION provides a prebuilt UDP implementation for LTP so we will use that.

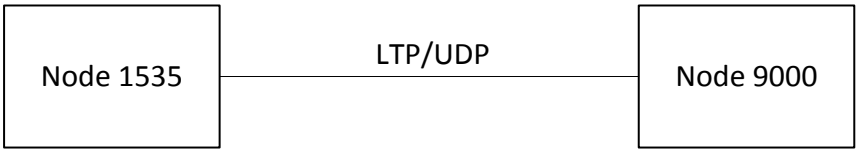

Figure 32 Two Node Example with LTP

This configuration will be very similar to the first example. If you saved your configuration from the first example, the easiest way to create this example is to make a copy and edit it. This section will only explain the differences between the configurations and not have step-by-step instructions for what has already been covered. This example will assume that you have saved your configuration file from the two-node configuration in Section 7.1.

Step 1 – Copy Configuration

Start the IONconfig application from your systems start menu. Open the configuration file from the first example and save it with a different name.

[23] The Advanced Orbiting Systems (AOS) Space Data Link Protocol is another CCSDS protocol and beyond the scope of discussion of this document. You can find more information on AOS at http://www.cccsd.org

Step 2 – Edit Node 1535

Double click on Node 1535 and go to the LTP tab. The dialog will look like Figure 33.

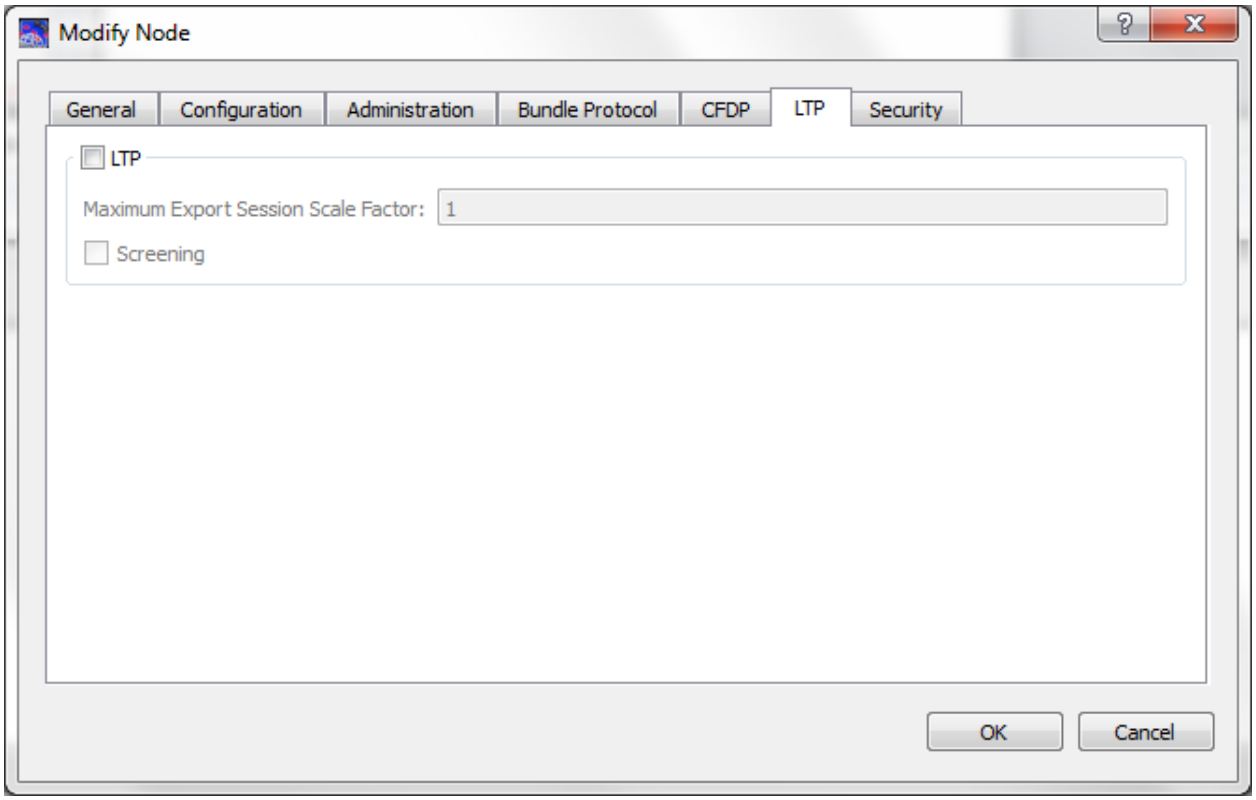

Figure 33 LTP Properties

Check the box next to LTP to enable LTP for this node. LTP uses "sessions" of data. Data generated on this node is referred to as "export sessions". Export Sessions are defined in the Duct dialog and stored in a map. ION uses a hash table to access sessions quickly during processing. The *Maximum Export Session Scale Factor* can be used to increase the size of the hash table to help speed up the access in looking up sessions. Unless you have reasons to believe that this has become a problem, just leave the default.

The *Screening* checkbox allows you to require ION to throw away unexpected LTP data. ION knows when data should arrive based on the contact plan. If screening is turned on, then ION will drop any data received outside of a known contact. This requires computer clocks at each node to be synced. It is recommended that you leave this option off.

After you have made the changes to the dialog it should look like Figure 34.

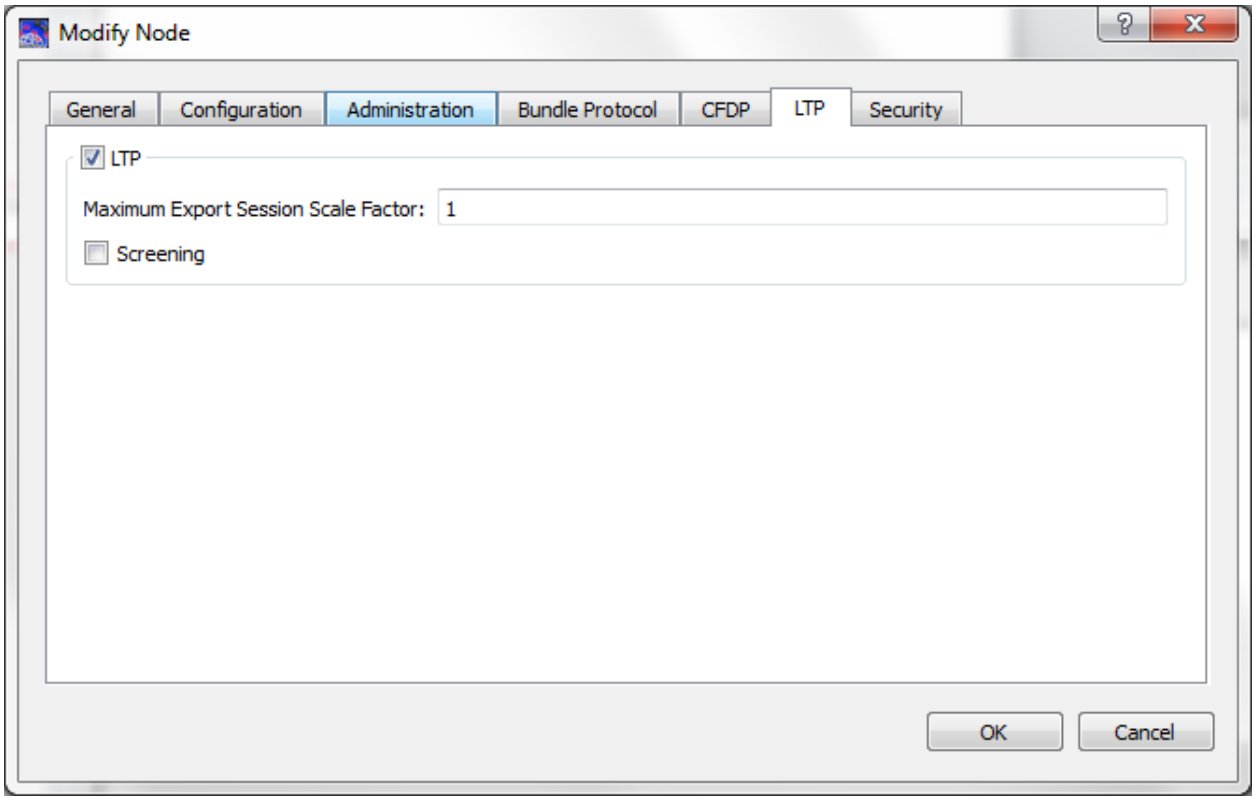

Figure 34 LTP Properties (Node 1535)

Click on OK to save the changes.

Step 3 – Edit Node 9000

Double click on Node 9000 and make the similar changes that were made for node 1535: Turn LTP on (LTP tab).

Step 4 – Modify the Duct

Double click line between the two nodes to modify the duct. The dialog in Figure 35 will appear.

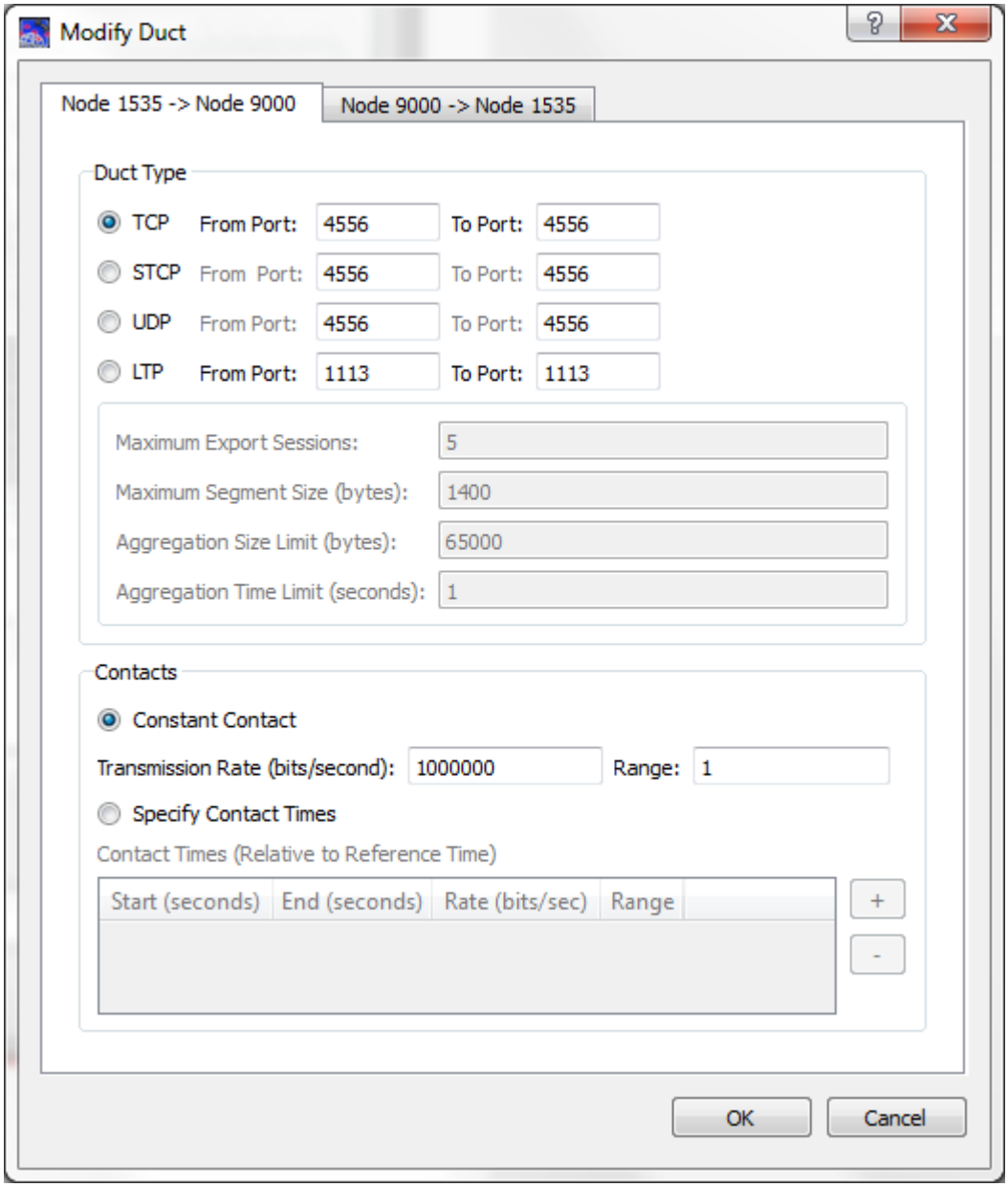

Figure 35 Duct Properties

Change the Duct type to LTP. The port numbers for LTP are the defaults for the protocol and can remain unchanged. The four options available when you select LTP are related and should be set based on your operational needs. The ION installation includes a spreadsheet and document that can help you determine the values to use. The following paragraphs also contain guidance. It is suggested that you run tests with these parameters to ensure that they work well.

The *Maximum Export Sessions* impose flow control on LTP and helps prevent the space in the data store from being filled up by LTP transmission sessions. The value you use for *Maximum Export Sessions* should take into consideration the expected size of bundles, the data rate for your transmission, any delay in the network (the time it takes to get to the next node), and disruptions in the link (data that may not make it to the next node). If you underestimate the number of export sessions needed, you will impose an additional throttle on your transmission rate. If you overestimate the number of export sessions needed, you will use more system resources (memory) than necessary.

You can use the following formula to help determine the value to set for *Maximum Export Sessions* (Note: The result is a minimum required value to prevent unwanted throttling of data and may need to be increased after testing in an operational environment.):

maximum export sessions = df * xr * rtt / (asl * 8)

Where:

df: disruption factor to cover undelivered data (suggested value 1.2)
xr: transmission rate defined for contact in this dialog
rtt: round trip time for packets on this network (generally twice range value)
asl: aggregation size limit as set in this dialog (times 8 to convert to bits)

The *Maximum Segment Size* defines the block size for outbound data. Typically this will match the value for the “payload bytes per frame” set on the Bundle Protocol tab of a node. The default value of 1400 should be sufficient.

The *Aggregation Size Limit* is used when smaller bundles are used with LTP. LTP will aggregate these small bundles into a single “session”. The *Aggregation Size Limit* defines how much data to collect before creating a new LTP session to transmit it. The *Aggregation Time Limit* defines how long LTP will wait while aggregating small bundles prior to sending them. Aggregating small bundles allows LTP to require fewer sessions.

For this example we will use the defaults for all of these parameters. For actual operations, you should test your LTP parameter values to ensure that they will provide optimal performance. LTP can still work with bad inputs for these parameters, but you will not efficiently use the bandwidth.

LTP requires knowledge of the contacts. For actual space operations this is typically accomplished by ingesting formatted information about satellite and ground contacts into a program to generate the correct ION formatted files. You can also manually add contacts on the Duct dialog. The *Transmission Rate* is used to throttle the LTP data. The *Range* defines how long it takes for data to travel between the two nodes. For this example, we will change the transmission rate for data from node 1535 to node 9000 to 400,000. The dialog should look like Figure 36.

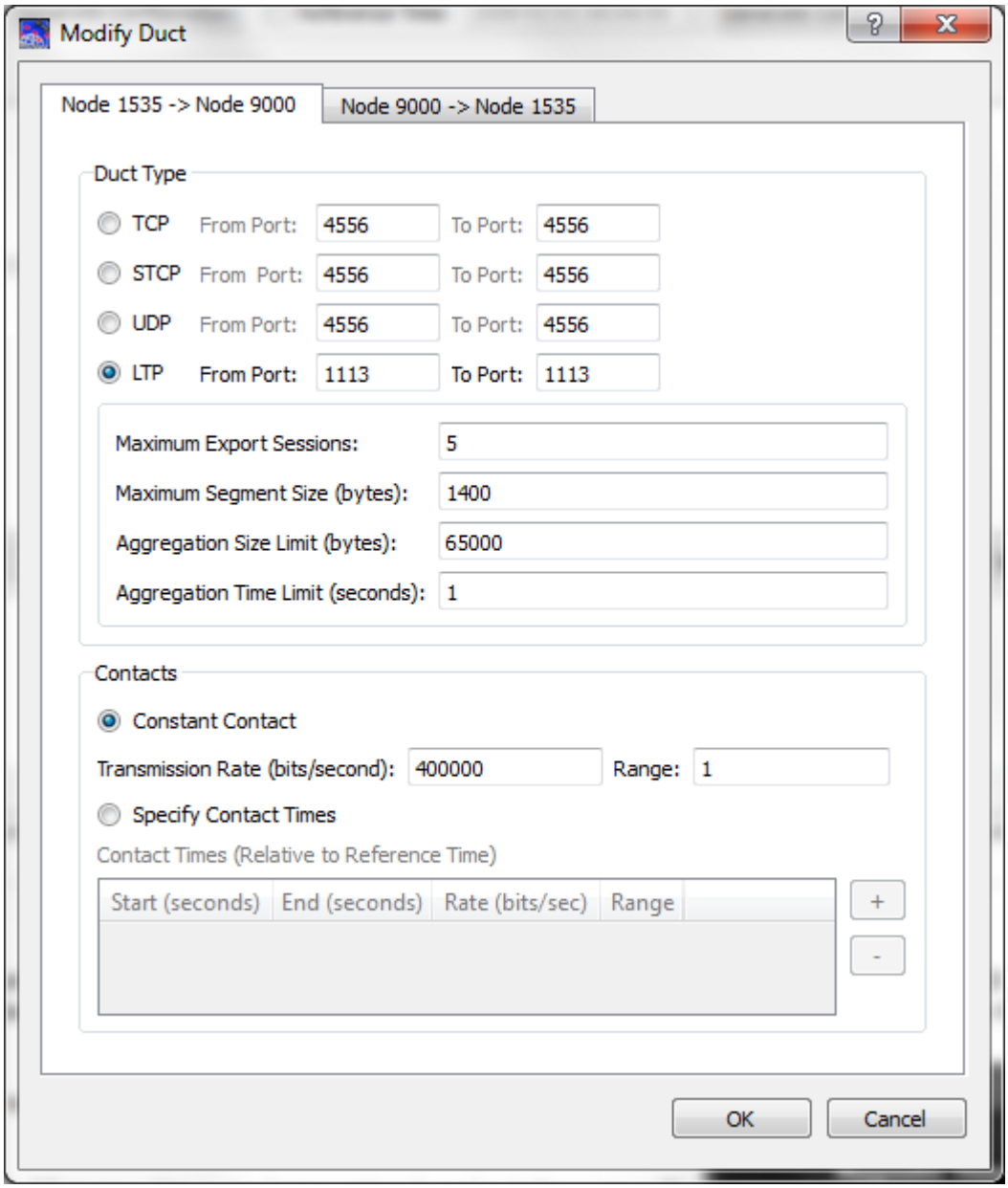

Figure 36 Duct Properties (LTP Part 1)

Click on the tab labeled "Node 9000 -> Node 1535" and make the same changes. This time change the transmission rate to 8,000,000. The dialog should look like Figure 37.

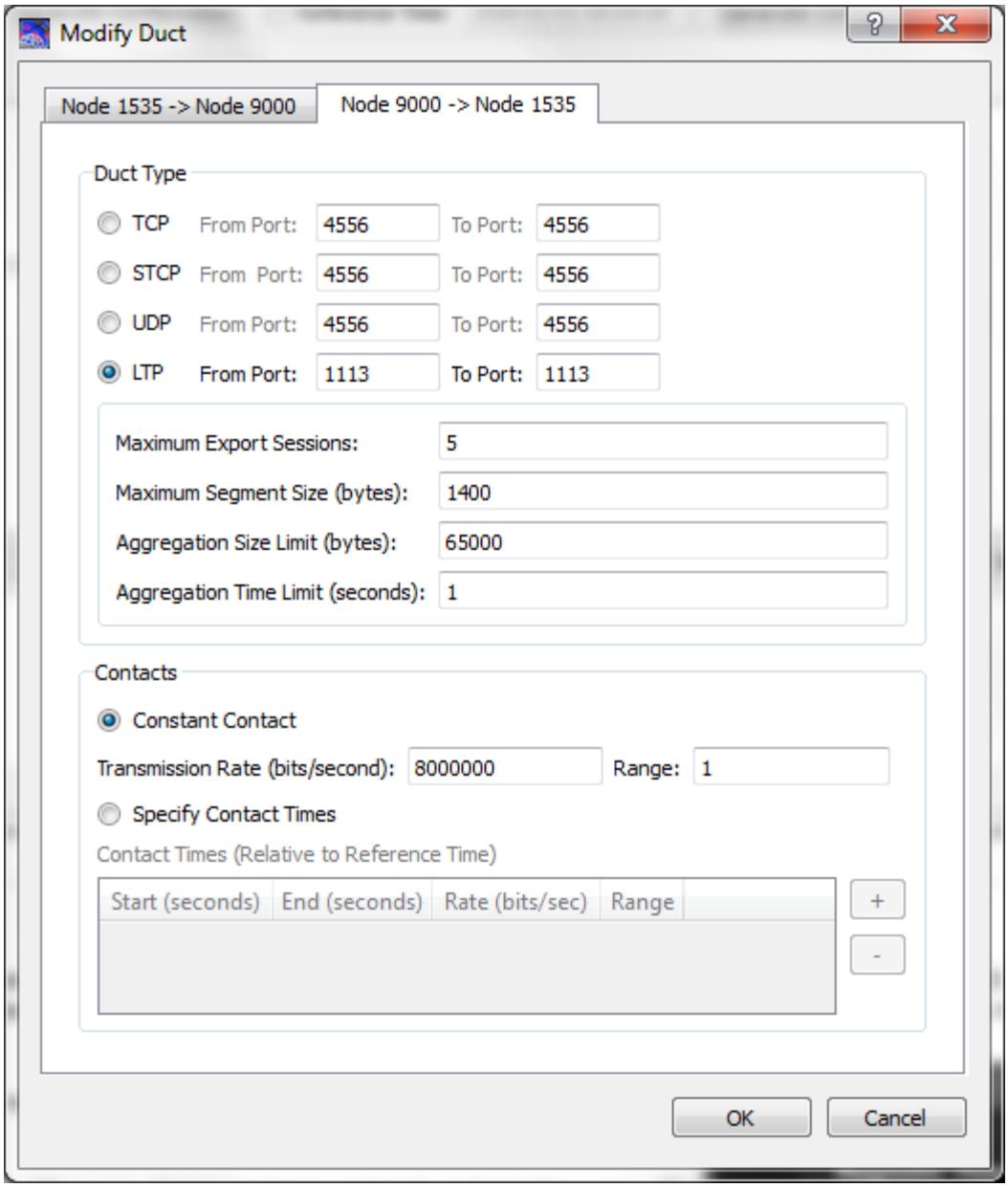

Figure 37 Duct Properties (LTP Part 2)

Click on OK to save the changes.

Step 4 – Generate the Configuration

You may wish to change the *ION Configuration Files Base Directory* to prevent overwriting the configuration files generated from the first example. After making the directory change, push the Generate Configuration button. The files can now be used with IONizer as described in the first example.

8 Terminology

This section contains a summary of the terminology related to DTN, ION, and TReK. The items are listed in alphabetical order.

***Bundle* –** The bundle is the primary transfer block of data over a delay tolerant network. Bundles flow between each node on the network. RFC-5050 defines the Bundle Protocol specification.

***Contact Graph Routing (CGR)* –** Contact graph routing is a technique used to create efficient routes for forwarding of DTN bundles when the contacts among nodes are not continuous. CGR also provides efficient means of bundle forwarding when multiple routes are available.

***Contact Plan* –** A contact plan defines when nodes within a delay tolerant network can communicate. In some cases the communication between two nodes may be continuous. For nodes that are part of a space link these communications times are almost never continuous. The contact plan allows each node to know when it may send data to another node or receive data from another node.

***Convergence Layer* –** The convergence layer is responsible for sending of bundles between nodes. For ISS communications the convergence layer is either TCP/IP or UDP/IP.

***Custody Transfer*[24] –** Custody transfer is a request made from the node sending the data to the node receiving the data. If the receiving node accepts custody of the data, the sending node can delete its copy of the data. By accepting custody of the data the receiving node accepts responsibility for ensuring the data is transferred to the final destination or until it can pass custody to another node. By passing custody in this manner a single copy of the data can be maintained in the DTN.

***Data Store* –** See Simple Data Recorder

***Data Throttling* –** Data throttling is limiting the rate at which a convergence layer outputs data. When using UDP or TCP as the convergence layer ION will default to sending data as fast as the underlying protocol can handle. There are cases where the receiving node may need data at a slower rate to prevent problems such as exceeding the capacity of the data store.

***Duct* –** A duct is an ION specific term referring to the communication mechanism used to send data between nodes.

[24] ION 3.3.0 does not fully implement custody transfer. Ensuring data delivery between nodes is done by the underlying convergence layer protocol. For ISS, TCP is used for nodes that have continuous contact. For nodes with disruptive links, LTP provides retransmission of the data.

Echo Service – The echo service is provided by the bpecho executable in ION. A node running the echo service can respond to bundle pings initiated by the bping[25] executable. This allows for a simple bundle protocol check to ensure proper network configuration.

Endpoint – An endpoint determines how to route the bundle data to the appropriate application.

Gateway – Any node that routes data for other nodes is considered a DTN gateway. The DTN gateway looks at the node number in the bundle and routes the data to the appropriate node in the network.

ion.log – The ion.log file contains all messages generated by the ION processes. When using the TReK IONizer application, the ion.log file is displayed in the main window. Details on the messages found in the ion.log file can be found in the ION documentation. An abbreviated version is available in Section 4.3.1.

Licklider Transmission Protocol (LTP) – The Licklider Transmission Protocol (LTP) is one of the DTN protocols used to provide reliable retransmission of data over disruptive and long delay links.[26] LTP provides an efficient mechanism for the retransmission of missing data over disruptive links by providing checkpoints which trigger the node receiving the data to report on what data has been received and what is missing.

Node – Each member of a DTN network is referred to as a node. In most cases a node can be considered a single computer. Each node has a unique number that identifies it. This number is known as the node number. In most cases there will only be a single node on any computer. There are cases where multiple nodes can be found on a single computer, but that isn't usually the case for ISS operations.

The node number for a computer is analogous to a computer's IP address. When multiple nodes reside on a single computer, you can think of that as a computer with multiple network cards each with its own IP address.

Service Number – Each bundle has a service number which helps the DTN nodes understand what application is associated with the bundle. Just as the node number is analogous to an IP address, the service number is similar to a port number.

Simple Data Recorder (SDR) – The simple data recorder (also referred to as a data store) stores bundles that need to be transmitted as well as bundles that are received. The size needed for the SDR is dependent on the amount of data transferred to/from the node and the length of time the data must reside on the node prior to the transfer occurring.

[25] The bping executable is not supported on Windows. The executables bping and bpecho are ION specific and will not necessarily work with other implementations of DTN.
[26] The LTP specification is defined in RFC 5326.

Static Routing – Static routing refers to the routing of bundles through a known path. With static routing the sending node often relies on other nodes to forward bundles and the sending node has no knowledge of the route or link availability along the route.

TReK Proxy – The TReK Proxy hides HOSC interface complexities from the ION node on the payload ground computer.

Watch Characters – A feature in ION that writes a single character to standard output for events that are considered significant for the underlying DTN protocols. These watch characters can be used to help determine and identify problems with the DTN node. You can see a complete list of ION characters in the ION documentation. A list of the ones available with TReK can be found in the IONizer application.