

**Annual Report: Research Supporting  
Satellite Communications Technology  
by**

**Stephen Horan and Raphael Lyman**

**NMSU-ECE-04-002**

# Annual Report: Research Supporting Satellite Communications Technology

Stephen Horan and Raphael Lyman  
Manuel Lujan Space Tele-Engineering Program  
New Mexico State University  
Las Cruces, NM

---

Prepared for

National Aeronautics and Space Administration  
Goddard Space Flight Center  
Greenbelt, MD

under Grant NAG5-13189

March 31, 2004



Klipsch School of Electrical and Computer Engineering  
New Mexico State University  
Box 30001, MSC 3-O  
Las Cruces, NM 88003-8001

# CONTENTS

List of Figures.....	iii
List of Tables.....	iv
SUMMARY.....	1
1 Faculty and Students Supported.....	2
2 Fault-Tolerant Link Establishment.....	2
2.1 Introduction.....	2
2.2 Link Establishment Requirements and Assumptions.....	3
2.3 Approach.....	5
2.3.1 Matlab-Based Development.....	6
2.3.2 LabVIEW-Based Development.....	6
2.4 Protocol Requirements.....	7
2.5 Protocol State Description.....	12
2.5.1 Cluster Control.....	13
2.5.2 Cluster Head States.....	14
2.5.3 Cluster Slave States.....	16
2.5.4 Cluster Heartbeat Message Processing.....	17
2.5.5 Handshake Message Processing.....	19
2.5.6 Data Message Processing.....	21
2.6 Test Program.....	21
2.6.1 Further Testing.....	28
2.7 Year-Two Program.....	28
2.7.1 Requirements Refinements.....	28
2.7.2 Testing at GSFC.....	30
2.7.3 Expansion of the Testbed Size.....	30
2.8 Dissemination of Results.....	31
2.9 References.....	32
3 AUTO-CONFIGURABLE RECEIVER.....	33
3.1 Introduction.....	33
3.2 Using the Signal Statistics.....	34
3.3 Probability of Error.....	38
3.4 Using the Frame Structure.....	41
3.5 Estimation Algorithm.....	44
3.6 Testing.....	46
3.7 Year-Two Work Plan.....	50
3.8 Dissemination of Results.....	51
3.9 References.....	51

## List of Figures

No.	Title	page
2-1	Cluster control state diagram.	14
2-2	Cluster Head states.	15
2-3	Cluster Slave states.	16
2-4	Checking for the presence of Heartbeat messages for the Cluster Head and Cluster Slave nodes.	18
2-5	States in processing the heartbeat messages.	18
2-6	Checking for the availability of handshake messages for the Cluster Head and Cluster Slave nodes.	19
2-7	States in processing a handshake message for the Cluster Head and Cluster Slave nodes.	20
2-8	Testbed configuration for current use and proposed maximum	31
3-1	Data Formats	35
3-2	Power Spectral Densities for Data Formats	37
3-3	Confirmed NRZ Waveform	38
3-4	Convolutional Encoder	45

## List of Tables

No.	Title	page
2-1	Fault Tolerant Network Test Plan	23
2-2	Test Program Progress	25

## SUMMARY

This report describes the first year of research effort under the grant "Research Supporting Satellite Communications Technology," NAG5-13189. The research program consists of two major projects: Fault Tolerant Link Establishment and the design of an Auto-Configurable Receiver that are being conducted by faculty and students at New Mexico State University (NMSU).

The Fault Tolerant Link Establishment protocol is being developed to assist the designers of satellite clusters to manage the inter-satellite communications. The protocol design is based on token passing to establish channel access permissions and periodic heartbeat messages to probe for link failures between nodes. The protocol management permits the overall cluster of satellites to partition itself into subnetworks to maintain connectivity between subsets of nodes based on mutual connectivity. Within each subnet, there is a cluster head to manage the token passing. Subnetworks can also merge to form larger subnets. Inherent within the design is the recognition that these inter-satellite links will occasionally undergo corruptions that may make a link unreliable. The protocol is being designed so that momentary corruptions of message traffic will not cause link failures. At the end of this first year, the basic design is being realized in software and testing of the software modules has begun. After this testing is completed at NMSU, further testing at Goddard Space Flight Center will be attempted to determine how well this protocol will work with the GSFC testbed.

The need for a receiver autoconfiguration capability arises when a satellite-to-ground transmission is interrupted due to an unexpected event, the satellite transponder may reset to an unknown state and begin transmitting in a new mode. Data will be lost while the ground-station receiver determines the new mode and makes adjustments. To speed the reconfiguration of the receiver, we are developing algorithms that allow the new transmission parameters to be determined automatically by examining the received signal itself. The parameters of interest for the TDRSS Multiple Access Return Service are data rate, data format, and details of the convolutional encoding. We have found that some of these parameters can be determined reliably based on the statistics of the signal alone, while others require the assumption that the data sequence is organized according to some known data-link protocol. Simulation test results show that, under noise-free conditions, our algorithms have a parameter-estimation error rate of less than .001. Additional preliminary testing shows that modeling noise-induced bit errors at a probability of  $10^{-4}$  results in an estimation error rate of about .01. We are continuing our study of how the estimation results are affected by noise. Also, this summer we plan to begin testing some of the algorithms on the Crosslink Simulator at the NASA Goddard Space Flight Center.

# **1 Faculty and Students Supported**

The research program consists of two major projects: Fault Tolerant Link Establishment and the design of an Auto-Configurable Receiver. The following faculty and staff contributed to the program for the first year:

- Dr. Stephen Horan, Professor of Electrical and Computer Engineering
- Dr. Raphael Lyman, Assistant Professor of Electrical and Computer Engineering
- Dr. Phillip DeLeon, Associate Professor of Electrical and Computer Engineering
- Mr. Qingsong Wang
- Mr. Giriprassad Deivasigamani
- Mr. Rahul Vanam

Dr. DeLeon gave assistance in the early development of the Auto-Configurable Receiver. Mr. Wang was with the project for the first few months. He was replaced by Rahul Vanam. It is the expectation that both students, Vanam and Deivasigamani, will be using this work as the basis for their MSEE thesis projects.

## **2 Fault-Tolerant Link Establishment**

### ***2.1 Introduction***

This report describes the work performed under the first year of the algorithm development for the cluster networking algorithm taking channel errors into account. In examining approaches to this problem, we decided to attempt a realization of an algorithm for routing proposed by Chiang et al. [1]. This particular algorithm was designed for use in fading channels and allows the network of nodes to self-organize

into smaller sub-networks. This algorithm was designed for use by 100's of nodes in the network, have a good degree of stability in assigning the roles of cluster head and cluster slave, and allow nodes to move between sub-networks. These are all characteristics of the desired satellite cluster protocol. The routing algorithm is based on a Least Cluster Change method to decide to which sub-network a node belongs. The cluster head controls the transmission of traffic by use of a token to grant permission to each node for channel access. Sequence numbers are used in the cluster management traffic to eliminate stale information and help nodes synchronize. In the development of the protocol, we use this basic philosophy and augment it with persistence metrics to ensure that simple channel errors do not cause links to be marked as broken or nodes unreachable.

## ***2.2 Link Establishment Requirements and Assumptions***

The link establishment algorithm will leverage off techniques designed for mobile applications with the additional requirements necessary for space cluster applications. In this section, we will look at the requirements and environment that we assume will need to be met in space. In subsequent sections, this information will be used to derive requirements for the protocol and test the operations of the protocol.

The cluster initiation and operation algorithm will be based upon the following assumptions about the cluster environment:

1. The cluster members will be homogeneous as to communications capabilities, at least initially.

2. The overall cluster can be broken into sub-networks to allow for communications power limitations.
3. Sub-networks can be joined when there is connectivity between them.
4. Each sub-network will have only one cluster head and the remaining sub-network members are slaves to the cluster head.
5. Only the cluster head communicates with the ground network.

The cluster initiation and operation algorithms will be based upon the following requirements for the cluster environment:

1. The algorithm needs to be scalable for cluster size and cluster architecture.
  - a. The cluster size needs to accommodate clusters of up to 64 members.
  - b. The cluster architecture needs to accommodate networks of networks having up to 64 members.
  - c. The cluster architecture can form new sub-networks whenever elements become unreachable.
2. The cluster must validate a node becoming unreachable to prevent channel errors from forcing false sub-network establishment and subsequent cluster reconfigurations.
3. If a cluster head fails, there must be a procedure to designate a new cluster head.
4. If a cluster head becomes unreachable, there must be a procedure to designate a new cluster head.
5. There must be a procedure for adding new members into any given sub-network and determining which node shall be the new cluster head.

6. The integration procedure must validate the candidate new members for inclusion in a sub-network are truly within proximity and that statistical variations in communications quality are not being interpreted as proximity.
7. The link algorithm shall support at least three grades of service.
  - a. The first grade of service is for high priority, low bandwidth, and minimal latency traffic needed for cluster operations, e.g. instrument pointing.
  - b. The second grade of service is for medium priority, high bandwidth, and medium latency traffic needed for cluster operations, e.g. instrument data sets.
  - c. The third grade of service is for low priority, low bandwidth, and long latency traffic needed for cluster operations, e.g. health and welfare data.
8. If a sub-network has only one member, it must be capable of becoming a cluster head.
9. The communications distance between cluster members is assumed not to exceed 100 km.
10. The communications distance between the cluster head and a ground access point is assumed to be that of a Low Earth Orbit satellite and will not exceed 1000 km.

### **2.3 Approach**

For ease of coding, the development of the protocol will not be directly executed in a high-level language such as C. Instead, a state generation tool based on a high-level

language will be used. In particular, we have looked at two approaches: one based on *Matlab* and one based on *LabVIEW*.

### **2.3.1 Matlab-Based Development**

The *Matlab* environment is successfully used in many analysis environments for communications, signal processing, and controls. One addition to the Matlab environment is the *Stateflow* toolkit. *Stateflow* is designed for tasks such as protocol development that can be expressed in terms of states with well-defined transitions. During the fall 2003 semester, the effort was directed towards developing the protocol state diagram in *Stateflow*. While this product does have a large learning curve associated with it, the main deficiency found with *Stateflow* is that it does not directly support networking protocols such as Transmission Control Protocol (TCP) and Unconnected Datagram Protocol (UDP). These need to be developed in other Matlab environments and then run with the Stateflow modules. After a number of unsuccessful experiments with *Stateflow* and *Matlab*, there was not any successful configuration to make this work that we were able to devise. Therefore, an alternative approach was sought.

### **2.3.2 LabVIEW-Based Development**

In later 2003, the National Instruments released a *State Diagram* toolkit for use with the LabVIEW programming environment. This toolkit is very similar to the *Simulink Stateflow* toolkit. However, it has one major advantage: the LabVIEW environment fully

supports TCP and UDP communications modes without special modification or non-standard modules.

## **2.4 Protocol Requirements**

The cluster protocol development will be designed to meet the following preliminary requirements. These requirements will need to be further developed during year two of the project as we learn more about the operations of the cluster in the various testbeds.

### 1) Cluster Head and Slave State Table Configuration

A) The cluster members shall maintain a State Table for the cluster. This state

Table shall contain at least the following data items:

- i) IP address for the individual cluster member
- ii) The Routing Table
- iii) Designator of current node status as either a Cluster Head or a Slave
- iv) Number of nodes in the cluster
- v) Port numbers for transmitting
  - (a) Tokens
  - (b) Data
  - (c) Handshake messages
  - (d) Heartbeat messages
- vi) Token re-issue period
- vii) Heartbeat reissue period
- viii) Node timeout value

B) The Routing Table shall consist of the following data items at a minimum

- i) IP address for each node in the cluster
  - ii) Next hop from each node to the other nodes
  - iii) Number of hops between nodes
  - iv) Current node sequence number
  - v) Sequence Number Roll Over Flag
  - vi) Time since last heartbeat message received
- C) The State Table shall be checkpointed to a disk file whenever
- i) The cluster is initialized
  - ii) An updated Routing Table is received
  - iii) A forced checkpoint message is received
- D) The ordering of the nodes in the Routing Table is to be
- i) First entry is the IP address of the Cluster Head
  - ii) Subsequent entries are listed in the failover order for determining a new Cluster Head
- E) Each node shall be able to determine its order in the failover sequence from the Routing Table
- F) Each node shall determine its current status as either a Cluster head or Slave from the State Table
- G) Messages between nodes shall use UDP as the message passing protocol
- 2) Generic Message Identifiers and Structures
- A) Message Types for handshake messages are as follows. The message type shall be a single ASCII character at the start of the data field for the message. All acknowledgement messages will use the same transmit port identifier.

Listeners shall pick up the acknowledgement on the port numbered one higher than the transmission port.

- i) A – Acknowledgement message to complete a handshake
- ii) B – Reset message to indicate that the node should fall back to the “factory” set values
- iii) C – Last Good message to indicate that the node should drop back to the “last good” checkpointed state
- iv) D – Routing Table message to update Routing Table contents

B) Sequence Number shall be generated as follows

- i) Initial value is 0
- ii) Unsigned 32-bit integer
- iii) Increments by 2 for every heartbeat or Routing Table update message transmitted
- iv) Rollover Flag TBD

C) Total Routing Table update messages shall be generated as follows:

- i) Number of rows concatenated with number of columns as 32-bit integers in hex format
- ii) Routing table matrix of indicated rows and columns as 32-bit unsigned integers in hex format

D) Subsection Routing Table update message shall be generated as follows:

- i) Number of rows in the message as a 32-bit integer in hex format
- ii) For each new/updated node entry, the columns for that node in 32-bit unsigned integer hex format; Roll Over Flag as a Boolean variable.

### 3) Heartbeat Message Specifications

- A) The heartbeat messages will be transmitted using a UDP message protocol
- B) The heartbeat messages shall be transmitted to all nodes one hop away from the originating node using the Routing Table information.
- C) The heartbeat messages shall be transmitted every “heartbeat reissue period” as specified in the state table and transmitted from the heartbeat message transmit port as specified in the state table. The heartbeat listening port shall be 1 greater than the number assigned to the transmit port.
- D) Each node shall listen for heartbeat messages on the heartbeat listen port. If the heartbeat message is from a node in the receiving node’s routing table, then the receiving node shall
  - i) Update the sequence number in the routing table
  - ii) Update time last heartbeat message received in the Routing Table
  - iii) Check to see if the received heartbeat message represents a better link (smaller hop count) in the routing table or an odd sequence number.
    - (a) If this represents an improved metric or odd sequence number,
      - send the revised metric to the one-hop neighbors
      - checkpoint the new Routing Table
    - (b) If this represents no change or a worse metric, do nothing with the Routing Table.
- E) The structure of the heartbeat message shall be
  - i) Sequence number and Roll Over Flag of the originating node
  - ii) Time stamp

- F) Each node will monitor the arrival interval between receptions of heartbeat messages. If the time since last message received exceeds three times the timeout value, that node with the missing heartbeat messages will be considered to be unreachable by the monitoring node. This will cause
- i) Updating the Routing Table for the missing node entry to change the sequence number to an odd value
  - ii) Checkpointing the revised Routing Table
  - iii) Transmitting the revised Routing Table entry for the missing node to all one-hop neighbors of the monitoring node.
- G) If a heartbeat message is received from a node not in the current Routing Table, then
- i) Updating the Routing Table for the new node entry by appending it to the end of the current Routing Table
  - ii) Checkpointing the revised Routing Table
  - iii) Transmitting the revised Routing Table entry for the new node to all one-hop neighbors of the monitoring node.
- 4) Token Message Specifications
- A) The token messages will be transmitted using a UDP message protocol and initiate from the Cluster Head.
  - B) The token messages shall be transmitted to all nodes using the Routing Table information. Tokens will be transmitted in sequence following the order in the Routing Table.

- C) The token messages shall be transmitted every “token reissue period” as specified in the state table and transmitted from the token message transmit port as specified in the state table. The token listening port shall be 1 greater than the number assigned to the transmit port.
- D) The structure of the token message shall be
  - i) Time stamp
- E) Each node will monitor the arrival interval between receptions of token messages. If the time since last message received exceeds three times the timeout value, that node will assume that it has separated from the rest of the cluster. This will cause
  - i) The node missing the token will begin to reconfigure itself as a Cluster Head
  - ii) The existing Cluster Head from the original cluster will diagnose the missing token and reconfigure the cluster Routing Table as necessary.

## ***2.5 Protocol State Description***

In this section, we will look at the current development of the protocol for the cluster link establishment. This is to be considered to be the initial attempt at the overall link establishment protocol. This protocol is currently under test to ensure that the basic functions are properly configured. In year-two additional functions will be added and more extensive testing will be conducted of all states and features.

## 2.5.1 Cluster Control

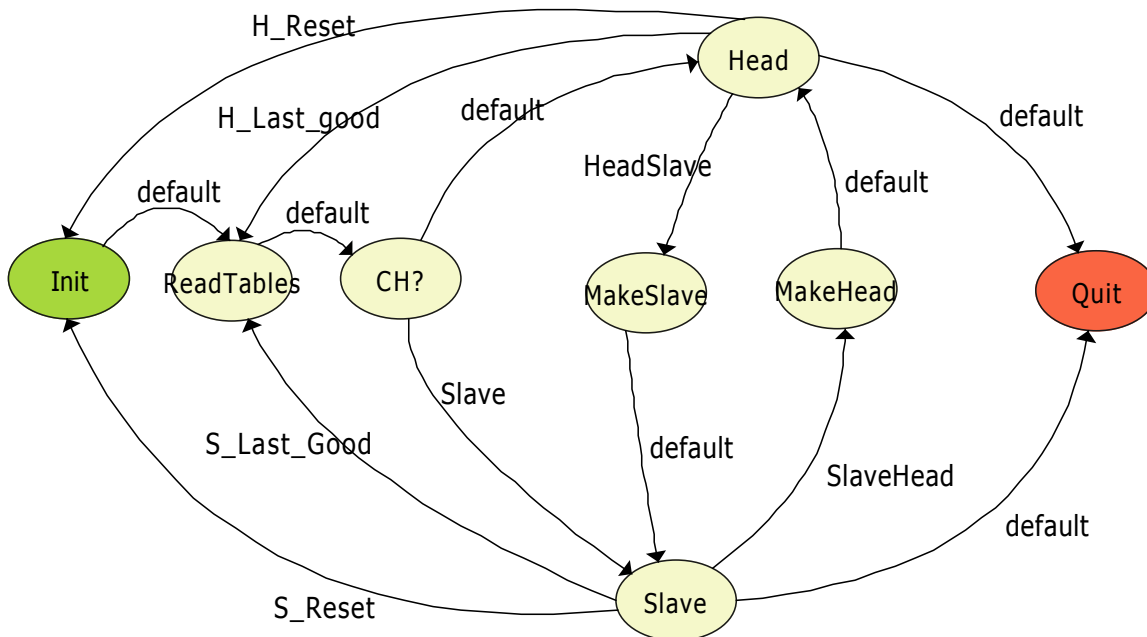
The cluster members will self-initialize upon power-up. The initialization will include the following capabilities:

- Determining if the node is a cluster head or a cluster slave based on IP address and location in pre-loaded routing table.
- Establish the node's state table
- Checkpoint the state table for backup.

After initialization, the node will enter either the "Cluster Head" or "Cluster Slave" states. If the head or slave detects faults, those nodes can exchange roles. The state diagram for the control process is illustrated in Figure 2-1. The control can also cause changes of state if certain messages are received:

- A Reset message causes the cluster nodes to go back to the re-initialization state.
- A Last Good message causes the cluster nodes to revert back to the last checkpointed state.
- A Quit message causes the protocol to terminate.

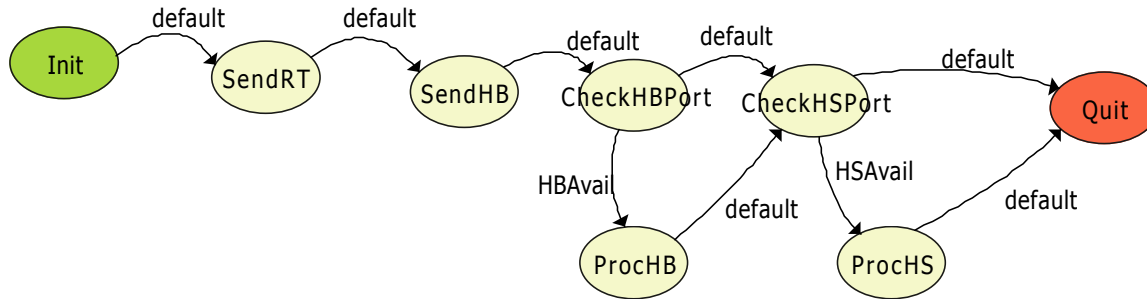
For now, we are anticipating that the Reset and Last Good messages would be sent from an operations control and not developed within state in the protocol itself.



**Figure 2-1** – Cluster control state diagram.

### 2.5.2 Cluster Head States

The Cluster Head has the responsibility of initiating token passing and acting as the initial node through which all messages are routed until the actual hop metrics are sorted out in the nodes. The initial state diagram for the Cluster Head is given in Figure 2-2. In this case, the Cluster Head state diagram is just the version being used for testing protocol modules. The final version will not merely exist after receiving a handshake message.



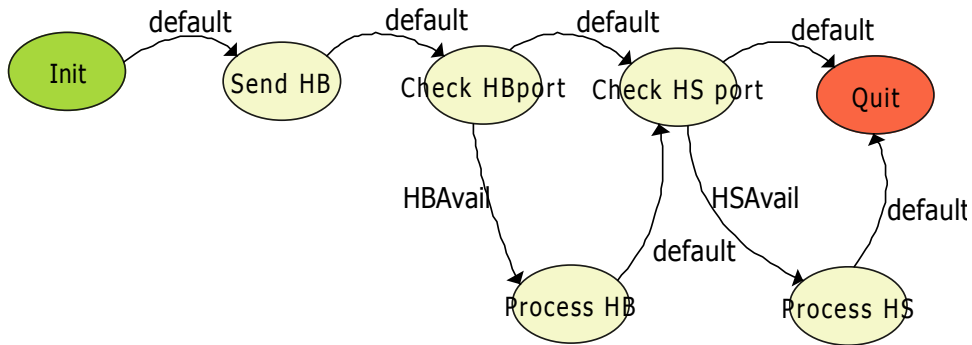
**Figure 2-2 – Cluster Head states.**

The states in the Cluster Head are

- Send initial Routing Table to all cluster members to ensure that correct values are stored in each node at power up.
- Send a heartbeat message to all nodes to ensure link connectivity.
- Check for heartbeat messages from other nodes.
- If a heartbeat message is found, process it.
- Check for a handshake message.
- If a handshake message is present, process it.

After checkout of the protocol processing is finished, the Cluster Head state diagram will go into a loop starting at the send heartbeat message stage.

A more detailed look at some of the states is given below since it will be the same for both the Cluster Head and the Cluster Slaves.



**Figure 2-3** – Cluster Slave states.

### 2.5.3 Cluster Slave States

Each node in the cluster that is not a Cluster Head is termed a Cluster Slave. The Slave has the responsibility of subsequent token passing. The initial state diagram for the Cluster Slave is given in Figure 2-3. In this case, the Cluster Slave state diagram is just the version being used for testing protocol modules. The final version will not merely exist after receiving a handshake message.

The states in the Cluster Slave are

- Set initial Routing Table to all cluster members based on pre-established rules; in this case, all nodes communicate through the Cluster Head so it is one hop away and all Cluster Slaves are two hops away from any given node.
- Send a heartbeat message to all nodes within one link hop to ensure link connectivity.
- Check for heartbeat messages from other nodes.
- If a heartbeat message is found, process it.

- Check for a handshake message.
- If a handshake message is present, process it.

After checkout of the protocol processing is finished, the Cluster Slave state diagram will go into a loop starting at the send heartbeat message stage.

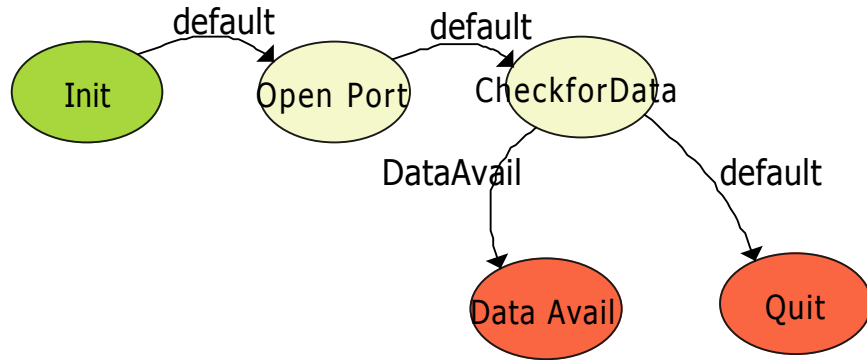
A more detailed look at some of the states is given below since it will be the same for both the Cluster Head and the Cluster Slaves.

#### **2.5.4 Cluster Heartbeat Message Processing**

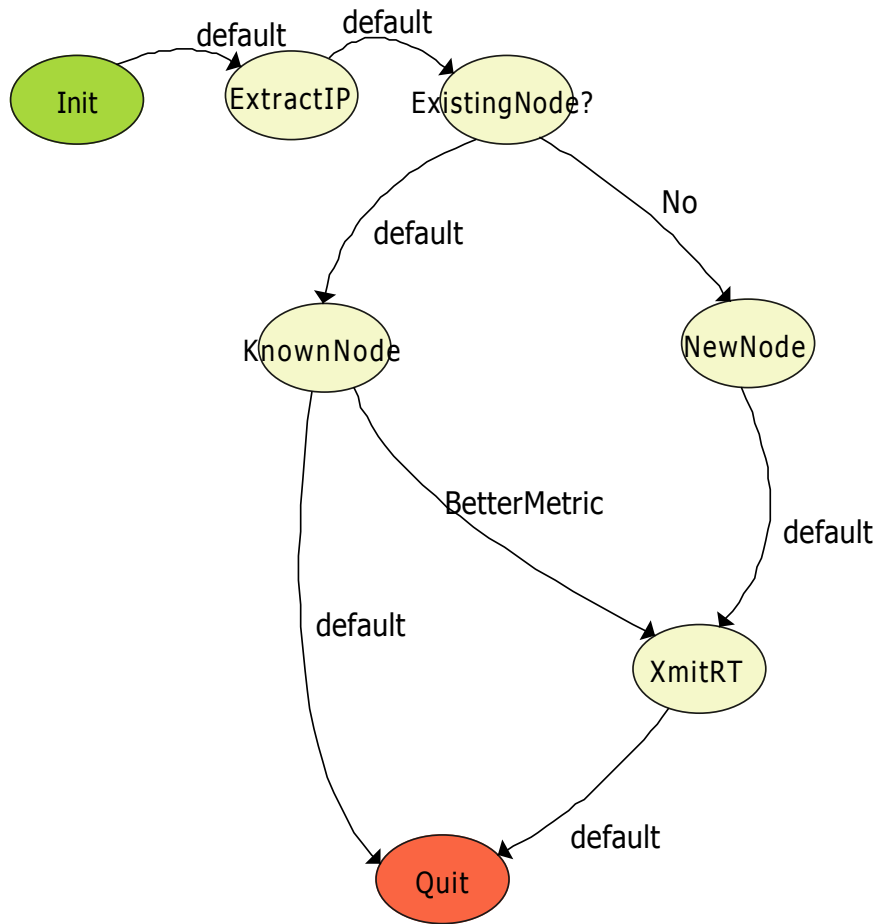
Both Cluster Slaves and the Cluster Head will use heartbeat messages to test the link connectivity between all one-hop neighbors. These messages will be sent at a rate determined by the cluster configuration management. The states involved with checking for the presence of a heartbeat message is given in Figure 2-4. Here, the protocol opens the UDP port, if data is not available (within 2 seconds), the state exists. If data is available, then the message is processed. The message processing is given in Figure 2-5.

When processing the heartbeat messages, the protocol goes through the following steps:

- Extract the originating node's IP address
- Determine if this is a node already in the Routing Table
- If the node exists in the routing table, then a check is made to see if the Routing Table has the same metric indicating that the node is one hop away.
- If the message indicates a better metric, then the Routing Table is updated and sent to all one-hop nodes.



**Figure 2-5** – Checking for the presence of Heartbeat messages for the Cluster Head and Cluster Slave nodes.

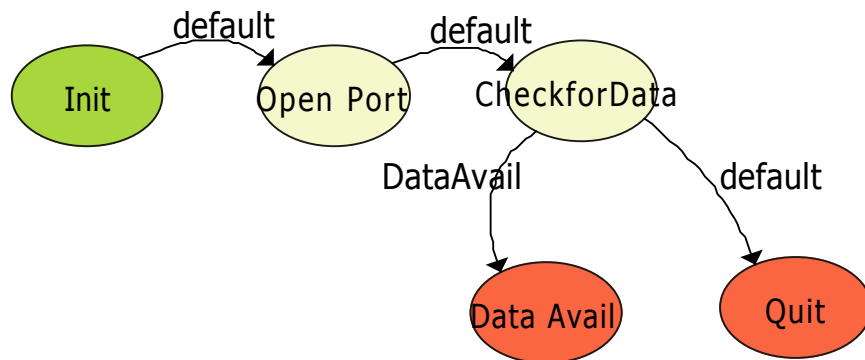


**Figure 2-4** – States in processing the heartbeat messages.

- If the originating node is not in the Routing Table, then it is added and the new Routing Table is sent to all one-hop nodes.

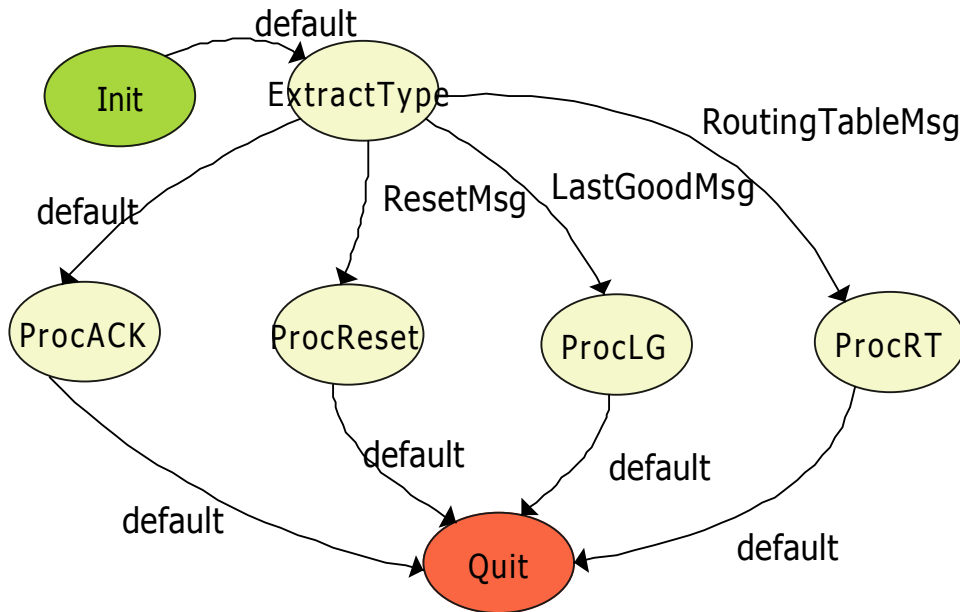
### 2.5.5 Handshake Message Processing

The states involved in checking for handshake messages are illustrated in Figure 2-6. This is used in both the Cluster Head and Cluster Slave nodes. As can be seen, the states mirror the heartbeat processing. The major functional difference is the different UDP port used for heartbeat messages and the handshake messages.



**Figure 2-6** – Checking for the availability of handshake messages for the Cluster Head and Cluster Slave nodes.

The handshake message processing is illustrated in Figure 2-7. In the protocol design, there are four types of handshake messages:



**Figure 2-7** – States in processing a handshake message for the Cluster Head and Cluster Slave nodes.

- Acknowledgements
- Reset
- Last Good
- Routing Table

The Acknowledgement message is designed for use with general handshakes. The Reset message is to send the node back to the initial “factory built” configuration, the Last Good message takes the node back to the last checkpointed state, and the Routing Table message updates the node Routing Table.

The reception of the Reset message and the Last Good message cause the overall control to revert to an earlier state as was illustrated in Figure 2-1. The reception of a Routing Table message will cause the following processes to occur:

- Parse the Routing Table entries from the sender
- Update the receiving node's Routing Table if a better hop metric is found
- Transmit a revised Routing Table, if it occurs, to the one-hop neighbors of the receiving node.

### **2.5.6 Data Message Processing**

Data messages will interact with the protocol's token passing mechanism. Each node will only be able to transmit data when it has the cluster token. At this time, the state diagram for Token passing is still being developed. We envision that this token passing mechanism may need to be able to have message priorities attached to allow different grades of service to be established.

## **2.6 Test Program**

The test program for the protocol is designed around building capabilities. The overall test plan for the initial development is given in Table 2-1. The details of the test program, to date, are given in Table 2-2. From these tests we expect to (1) validate the basic approach for the protocol and (2) determine what enchantments are needed and if there are better approaches to the protocol development. One alternative approach for consideration that has come from the tests so far is the use of UDP Multicast

Dissemination Blocks for the message transmission. We are also seeing potential ways to abstract some of the protocol processing to make it more general and have the state machine have fewer potential states.

**Table 2-1 -- Fault Tolerant Network Test Plan**

No.	Test Area	Sub Tests	Target Completion Date	Status
1	Cluster Head (CH) and Slave (S) Configuration		12 March 2004	
		State Table Set up	12 March 2004	Complete
		Can find "MyIP" in tables	12 March 2004	Complete
		Properly Determine CH or S status	12 March 2004	Complete
		Open/Close Data Files	12 March 2004	Complete
		UDP Transfer of Messages	12 March 2004	Complete
2	Heartbeat (HB) Tests		24 March 2004	
		Normal HB one hop to CH	22 March 2004	Complete
		Loss of HB from Slave (single hop)		In progress
		New Member HB to CH directly		In progress
		New Member HB through Slave		In progress
3	Token Passing Tests		2 April 2004	

No.	Test Area	Sub Tests	Target Completion Date	Status
		Normal Token passing around cluster		
		Single dropped Token		
		Broken link		
4	Configuration Management Tests		16 April 2004	
		Slave move out of cluster		
		Slave move into cluster		
		Failover CH		
		Merge clusters		
5	Data Transfers			
		Single File	30 April 2004	
		Segmented Files		
		Multiple Files		
		Data Priority		

**Table 2-2 -- Test Program Progress**

Test No.	Test Name	Sections	Specifications Covered	Test Date	Status
1.1	State Table Initialization				
		<ul style="list-style-type: none"><li>• Read data from front panel</li><li>• Write data to checkpoint file</li><li>• Validate checkpoint file contents</li></ul>	1.A, 1.B, 1.C.1	March 12, 2004	Completed successfully
1.2	Update Routing Table				
		<ul style="list-style-type: none"><li>• Modify Routing Table</li><li>• Transmit to all nodes</li><li>• Verify all nodes checkpoint Routing Table and update their State Tables</li></ul>	1.G, I.C.2	March 12, 2004	Completed successfully
1.3	Find "My IP"				
		<ul style="list-style-type: none"><li>• Determine current sequence number in failover table</li></ul>	1.D, 1.E	March 12, 2004	Completed successfully
1.4	Determine current node status				
		<ul style="list-style-type: none"><li>• Display current status (CH or S)</li></ul>	1.F	March 12, 2004	Completed successfully

Test No.	Test Name	Sections	Specifications Covered	Test Date	Status
2.1	Heartbeat Message Generation				
		<ul style="list-style-type: none"> <li>• Message generated according to specified reissue period</li> <li>• Message has correct format</li> <li>• Message goes to all assigned nodes</li> <li>• Messages received by each node</li> </ul>	3.A, 3.B, 3.C, 3.E	March 22, 2004	Completed successfully
2.2	Heartbeat Message Processing				
		<ul style="list-style-type: none"> <li>• Single-hop message processing</li> <li>• New heartbeat destination based on revised Routing Table</li> </ul>	3.D	March 22, 2004	Completed successfully
2.3	Single message failure				
		<ul style="list-style-type: none"> <li>• Only lose a single message</li> <li>• Do not change Routing Table contents</li> </ul>	3.F	March 22, 2004	Completed successfully
2.4	Total Link Failure				
		<ul style="list-style-type: none"> <li>• Lose messages for extended period</li> <li>• Verify updated Routing Table is sent</li> </ul>	3.F	Currently in test	Incomplete
2.5	New Cluster member				

Test No.	Test Name	Sections	Specifications Covered	Test Date	Status
		<ul style="list-style-type: none"> <li>• Nearest member receives message</li> <li>• Propagate new Routing Table</li> <li>• New member included in heartbeat message distribution</li> </ul>	3.G	Currently in test	Incomplete

### **2.6.1 Further Testing**

As noted in Table 2-2, detailed testing of some of the features has not happened as of this report date. This testing will continue to validate the functions.

After the main functional testing is complete, the protocol will be converted to a Matlab-based equivalent using one of the LabVIEW toolkits for this purpose. Once this conversion is completed, regression testing of the Matlab-based configuration will commence prior to bringing the code to GSFC over the summer.

## **2.7 Year-Two Program**

The second year of the program will concentrate on developing capabilities to make the protocol more useful. In particular, the requirements developed at NMSU will need comments by interested observers at NASA. Further work will need to be done on integrating the data transfer with the token-based message passing. Also, as new concepts evolve from the testing, further enhancements to the protocol can be made. The following paragraphs address these issues in more detail.

### **2.7.1 Requirements Refinements**

The requirements developed to date need to be reviewed and tested against satellite cluster design concepts. It is expected that once this is done at the start of Year 2, further requirements refinements will occur and the protocol state diagram will need to be changed to reflect these refinements. NMSU personnel will need assistance from

NASA in arranging for contacts within NASA who will be able to guide the requirements refinement.

### **2.7.1.1 Data Message Processing**

The routing and coordination protocol does not explicitly involve message passing at this time. It is anticipated that an existing message-passing protocol such as the CCSDS File Data Protocol (CFDP) can be used for this function. It is not believed that CFDP is currently a gated protocol that will only transmit when a token is in hand at the sending node. Work will need to be done to see how to coordinate the data messaging protocol with the actual token passing mechanism.

### **2.7.1.2 UDP Multicast Block**

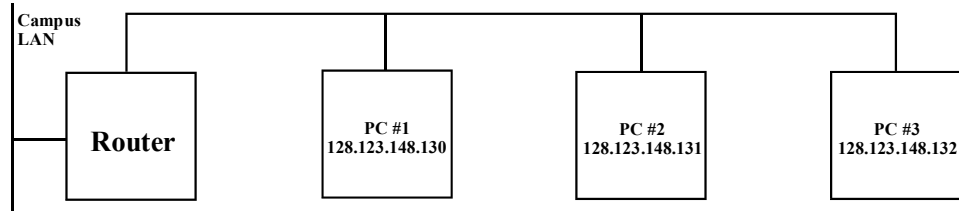
An interesting feature supported by LabVIEW is the ability to use Multicast UDP blocks. The multicast blocks use the address range of 224.0.0.0 to 239.255.255.255. We see the biggest potential for this address block in configuring the heartbeat messages for configuration checks within the cluster and allowing a means for merging partitions in reconfiguring the cluster. This addressing mode is not supported by all routers and switches. The NMSU networking personnel have configured the testbed to permit this in the testbed router. This may be a problem for other testbed locations when testing this method.

### **2.7.2 Testing at GSFC**

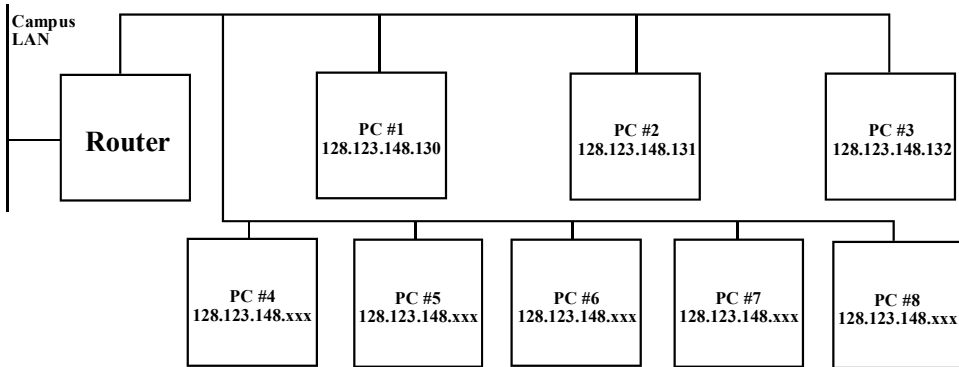
The first external test for this protocol development is the use of the communications testbed at Goddard Space Flight Center. We anticipate working with GSFC personnel to attempt this type of testing in June 2004. If this set of tests is successful, then further testing at that site should be scheduled. The largest unknown at this time is the conversion of the LabVIEW code to Matlab-based code and how this code will work on the GSFC testbed. One alternative to be considered if the Matlab conversion is problematic is to make the LabVIEW representation into a stand-alone executable package. The disadvantage of this is that the stand-alone LabVIEW is frequently a very large executable program.

### **2.7.3 Expansion of the Testbed Size**

The current testbed is composed of three PCs as illustrated in Figure 2-8a. This current configuration allows for a single cluster head and two slaves. While this is adequate for initial development, it is not adequate for more extensive testing dealing with cluster partitioning. For the next year, we would plan to expand the testbed configuration to a minimum of eight computers. This is illustrated in Figure 2-8b. In this case, a single router would still be used to service the testbed members. Based upon preliminary discussions with the NMSU Network Operations Center personnel, this is probably required, in our case, to properly configure the UDP Multicast block testing.



(a) Current testbed configuration



(b) Proposed testbed configuration for year two

**Figure 2-8** – Testbed configuration for current use and proposed maximum expansion capabilities for the next year.

## **2.8 Dissemination of Results**

The dissemination of the results of this first phase of the program will be in two immediate venues: the Space Internet Workshop IV to be held in June 2004 and the International Telemetry Conference to be held in October 2004. By the end of the summer, a paper should be ready for submission to the IEEE Aerospace Conference or a similar venue.

## **2.9 References**

- [1] C-C Chiang, H-K Wu, W. Liu, M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," IEEE Singapore International Conference on Networks, 1997, p. 197 – 211.

## 3 AUTO-CONFIGURABLE RECEIVER

### 3.1 Introduction

In order to demodulate a satellite-to-ground transmission correctly, a ground-station receiver must have knowledge of certain transmission parameters; e.g., data rate, data format, details of error-correction coding, etc. Normally, these are known ahead of time, but sometimes, when an unexpected event occurs, the satellite transponder may reset to an unknown state and begin transmitting data using a different set of parameters. This can cause data to be lost until the receiver operator determines that an event has occurred and makes appropriate adjustments.

The task of reconfiguring the receiver for the new parameters could be eased if the parameters could be determined automatically from the received signal itself. For example, data formats such as NRZ and BiΦ have distinctive statistical signatures [1]. Thus the data format, as well as the data rate, could be determined by estimating the power spectral density of the received signal. As we shall see, there are also time-domain approaches to these problems that offer simplicity and a method for calculating an error bound.

Some other parameters present more of a challenge. For example, there is no statistical distinction between random data formatted as NRZ-L and data formatted as NRZ-M. Such a distinction can be made, though, if we make use of *a priori* information regarding the frame structure of the transmitted data. Even if the link-layer protocol is not known at the ground-station receiver, if the number of possible protocols is small, we can distinguish between NRZ-L and NRZ-M by assuming that a particular one of

these data formats was used in the transmission. We then demodulate accordingly and determine if the resulting data “makes sense.”

The goal of this project is to develop algorithms that may be used by a ground-station receiver to examine a received signal and determine the transmission parameters that were used in generating it. To do this, we will use both statistical and frame-structure based techniques, as described above. We will see that estimation of some parameters, such as data rate, requires analog modeling, with the incoming signal waveform sampled at a high rate, while other parameters, such as details of convolutional encoding, require only the demodulated baseband bit stream.

The reference model for the study will be a satellite-to-ground link through the TDRSS Multiple-Access service [2]. In the sections that follow we will describe algorithms that examine the sampled analog waveform to determine the data rate and the general class of data format for a received signal, as well as a technique for computing a bound on the probability that the decision is in error. Then we will describe algorithms that operate on the demodulated baseband digital stream to determine the details of the convolutional coding, if any, as well as provide a more complete determination of the data format.

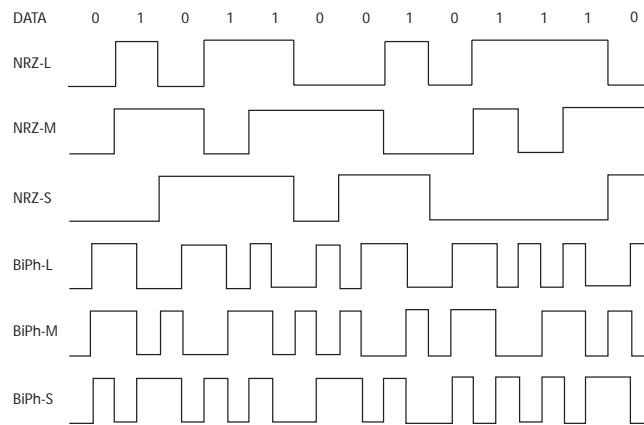
Finally, we will provide the details of our test plan and the current status of our testing, as well as a discussion of what our research goals and plans are for the following year.

### ***3.2 Using the Signal Statistics***

When examining a signal to determine the transmission parameters, keeping the algorithm simple is very important. For this reason, we wish to minimize the number of

assumptions we make about the received signal. Fortunately, some parameters may be determined by examining the signal statistics alone. Examples of this include the data format and the data rate.

In the TDRSS MA service [2], allowable data formats include *non return to zero* and *biphase* types (NRZ and Bi $\Phi$ , see Figure 3-1) [1]. NRZ-L formatted data, for example, responds to the logic “level” of the data, assigning one signal level to a logic 1 and another signal level to a logic 0. By contrast, Bi $\Phi$ -L assigns a signal-level *transition* to each logic level. The transition for logic 1 is in the opposite direction of the transition for logic 0.



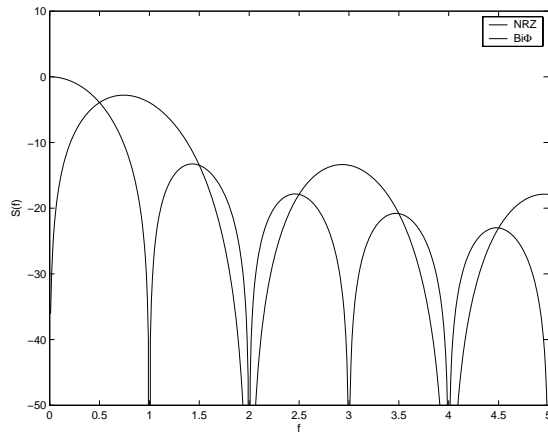
**Figure 3-9 - Data Formats**

NRZ-M and Bi $\Phi$ -M respond to a logic 1 or “mark.” For NRZ-M, a logic 1 is indicated by a change in signal level from the previous symbol. A logic 0 is indicated by no change in signal level. For Bi $\Phi$ -M, a logic 1 is indicated by a change in the direction of the signal-level transition with respects to the previous symbol. NRZ-S and Bi $\Phi$ -S are

similar to NRZ-M and Bi $\Phi$ -M, except that they respond to a logic 0 or “space,” instead of a logic 1.

In the Multiple Access service, the formatted symbols are further modulated using direct-sequence spread spectrum, but the chip rate and details of modulation are the same for all data rates. Thus, we assume that the signal has been despread using well-known code acquisition and tracking procedures [3].

Note that, in contrast to NRZ, all Bi $\Phi$  formats guarantee a signal-level transition during each symbol. For this reason, the power spectral densities of the two format types are distinct (see Figure 3-2). This suggests a frequency-domain approach for determining the data format of the received signal. We simply compute the periodogram of the signal and compare it to the spectra in Figure 3-2. For example, we note that the PSD of NRZ-formatted data has a negative slope at  $f = 0$ , whereas Bi $\Phi$ -formatted data has a positive slope there. If  $S[n]$  is the periodogram of the received signal, we may approximate the slope of the PSD using the forward difference  $S[1] - S[0]$  [4]. If this difference is negative, we conclude that the data are NRZ formatted. A positive difference indicates Bi $\Phi$ . The data rate could also then be estimated by determining the frequencies of the spectral nulls.

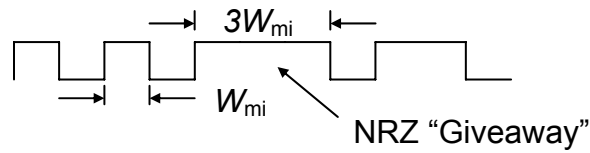


**Figure 3-2** – Power Spectral Densities for Data Formats

Unfortunately, in order to obtain an accurate estimate of the PSD using the periodogram, it is usually necessary to buffer a long data record, and there is no convenient way to estimate the probability that the data-format determination procedure described above will yield an incorrect result.

To address these problems, we consider a time-domain approach. For example, we may determine the data rate by sampling the received signal at a rate much higher than the greatest expected data rate. We then search the sampled signal for transitions in level. The shortest distance between adjacent transitions we call  $W_{\min}$ . By examining Figure 3-1, we see that if the data format is NRZ, then  $W_{\min}$  is almost certainly the bit period. If the data format is BiΦ, then  $W_{\min}$  will be one-half the bit period.

To determine whether the signal is NRZ or BiΦ, we note from Figure 1 that for BiΦ, the longest time between signal-level transitions is  $2W_{\min}$ . Thus, if our sampled signal contains a segment of length  $3W_{\min}$  without a transition, then it is certainly NRZ (see Figure 3-3).



**Figure 3-3** – Confirmed NRZ Waveform

If we examine our sampled signal and find that there is no segment of length greater than  $2W_{\min}$  without a transition, then we conclude that it is formatted as Bi $\Phi$ . This conclusion may be in error for certain transmitted data sequences. For example, if the transmitted sequence is “11001011” formatted as NRZ,  $W_{\min}$  will be equal to one bit interval, and the time between signal-level transitions will always be less than  $3W_{\min}$ . This is because the data we are transmitting happens to contain no sequence of three consecutive like bits; i.e., “111” or “000”.

### 3.3 Probability of Error

To compute the probability that the above procedure results in an incorrect decision, we note that the error event may be written as

$$E = (\text{Bi}\Phi \cap \text{dNRZ}) \cup (\text{NRZ} \cap \text{dBi}\Phi), \quad (1)$$

where, e.g., “Bi $\Phi \cap \text{dNRZ}$ ” means the event that Bi $\Phi$  data were transmitted, but we decided that the signal was formatted as NRZ. The two error types in (1) are mutually exclusive so from elementary probability we have [5]

$$P(E) = P(\text{Bi}\Phi \cap \text{dNRZ}) + P(\text{NRZ} \cap \text{dB}\Phi). \quad (2)$$

If BiΦ data are transmitted, then from Figure 3-1 we can see that, neglecting noise-induced errors, the signal will never contain a segment of length  $3W_{\min}$  without a transition in level. Thus, we will never decide that NRZ was transmitted when the data are in fact formatted as BiΦ,

$$P(\text{Bi}\Phi \cap \text{dNRZ}) = 0. \quad (3)$$

Also,

$$P(\text{NRZ} \cap \text{dB}\Phi) = P(\text{dB}\Phi | \text{NRZ})P(\text{NRZ}). \quad (4)$$

We see from (4) and (2) that the error probability depends on the *a priori* probability that the transmitted sequence was formatted as NRZ. Since this probability may not be known, we can bound the error probability by noting that  $P(\text{NRZ}) \leq 1$ . Making use of (4), (3) and (2) we have

$$P(E) \leq P(\text{dB}\Phi | \text{NRZ}). \quad (5)$$

If the received data are formatted as NRZ, the probability that we will decide in favor of BiΦ is just the probability that the transmitted data contain no sequence of three consecutive like symbols; i.e., no “111” and no “000”. The reason for this is that if such a sequence did occur, it would cause NRZ-L formatted data to hold a signal level for  $3W_{\min}$  or longer.

The same probability holds for NRZ-M and NRZ-S as well. To see this, note from Figure 3-1 that an NRZ-M waveform may be obtained by first differentially encoding the data, and then formatting the result as NRZ-L. If a random sequence is differentially

encoded, then the result is also random, and the probability of three consecutive like symbols is unchanged. In the case of NRZ-S, the original data must be inverted before differential encoding, but if a logic 1 is as likely to occur as a logic 0, the probabilities will again not be changed. We will make further use of these observations in the following section.

Suppose that the transmitted data consist of  $N$  binary symbols,  $\{b_1, b_2, \dots, b_N\}$ . Let us use the symbol  $NA_n$  to mean the event that  $b_n, b_{n-1}$ , and  $b_{n-2}$  are not all alike. Then

$$P(\text{dBi}\Phi \mid \text{NRZ}) = P\left(\bigcap_{n=3}^N NA_n\right). \quad (6)$$

The events  $\{NA_n\}$  are not independent. For example, if  $4 \leq n \leq N$ ,  $NA_n$  is not independent from  $NA_{n-1}$ , because the sequences  $\{b_{n-2}, b_{n-1}, b_n\}$  and  $\{b_{n-3}, b_{n-2}, b_{n-1}\}$  have the bits  $b_{n-2}$  and  $b_{n-1}$  in common. Thus, we evaluate the joint probability in (6) in terms of conditional probabilities [5]. For  $N \geq 5$ , this yields

$$P\left(\bigcap_{n=3}^N NA_n\right) = P(NA_4 \cap NA_3) \cdot \left(\prod_{n=5}^N P(NA_n \mid NA_{n-1} \cap NA_{n-2})\right). \quad (7)$$

It is easy to show that  $P(NA_4 \cap NA_3) = 5/8$ . The conditional probabilities are given by

$$P(NA_n \mid NA_{n-1} \cap NA_{n-2}) = \frac{4}{5}, \quad 5 \leq n \leq N. \quad (8)$$

Substituting these values into (7), and making use of (6) and (5), we have

$$P(E) \leq \left(\frac{5}{8}\right)\left(\frac{4}{5}\right)^{N-4}. \quad (9)$$

The bound in (9) may be used to show, for example, that an error probability  $P(E) \leq .0001$  may be achieved if the transmitted data are  $N \geq 44$  bits long.

Recall that at the receiver, we sample a segment of the received signal in order to determine the data format and the data rate. Suppose that, after doing the  $P(E)$  computations as above, we determine that we must sample enough of the signal to include at least  $N$  bits of transmitted data. How many seconds of the received signal does this represent? Since we do not yet know the data format or rate, we consider a minimum allowable data rate. For the Multiple Access service, the minimum data rate is 100 bps. If data transmitted at this rate are formatted as NRZ, then we expect  $W_{min} = 1/100$ , since the minimum pulse width for NRZ is just the bit interval. If we must have  $N \geq 44$  for  $P(E) \leq .0001$ , then we should sample a segment about .5 sec long. In practice, the low data rate 100 bps is seldom used. If it is reasonable to assume a higher minimum data rate, the length of the sampled segment could be reduced.

### **3.4 Using the Frame Structure**

We mentioned previously that an NRZ-M waveform may be obtained by first differentially encoding the data, and then formatting the result as NRZ-L. Data may be formatted as Bi $\Phi$ -M in a completely analogous manner. For NRZ-S or Bi $\Phi$ -S, the original data sequence is inverted before differential encoding. From this we see that once the format type, NRZ or Bi $\Phi$ , is determined, the signal may be demodulated as if it were NRZ-L or Bi $\Phi$ -L. Then the format suffix of the signal may be determined by whether the resulting sequence is differentially encoded.

As discussed previously, NRZ and Bi $\Phi$  may be distinguished using a statistical approach. But there is no statistical difference between a binary sequence before and after differential encoding. A distinction can be made, though, if we know something

beforehand about the data being encoded. For example, all of the data-link protocols most commonly used for space-to-ground transmission employ known binary sequences or *sync words* for proper frame synchronization [1]. If we differentially decode a received sequence, and if the decoded sequence has an occurrence of one of these sync words, then it is highly likely that the transmitted sequence was differentially encoded.

A similar approach may be used to determine the details of the convolutional encoding that was applied. For example, if Viterbi decoding of a rate- $\frac{1}{2}$  convolutional code is applied to a received signal, and if a recognized sync word is found in the resulting sequence, we conclude that the transmitted signal was coded in this way.

The line code and convolutional encoding may be determined jointly by trying all the possibilities. For example, suppose a received signal is known to be either L- or M-form modulation, and either uncoded or rate- $\frac{1}{2}$  encoded. We form the following four sequences: 1) the original sequence unchanged, 2) the original sequence after differential decoding, 3) the original sequence after Viterbi decoding of the convolutional code and 4) the original sequence after differential decoding and Viterbi decoding. Note that because convolutional encoding is applied before modulation at the transmitter, the operations are decoded in opposite order at the receiver to obtain sequence 4. The line code and convolutional encoding may be determined by searching each of the four sequences for sync word occurrences.

This approach provides a simple means of determining the type of modulation and convolutional encoding that have been applied to a received signal. A problem occurs, though, if one of the possible sync words is very short. For example, the HDLC *flag*

byte, “01111110”, is only eight bits long [6]. On average, we would expect this sequence to appear about once every 32 bytes in a random data stream. Thus, if a sequence is sufficiently long, it may contain a spurious occurrence of the flag byte even if it has not been properly decoded.

The problem may be addressed by handling HDLC separately. After forming the various decoded sequences from our received signal we test each to determine if it contains a valid HDLC frame. If one of them does, then we conclude that this is the correctly decoded sequence. Otherwise, we proceed to search for the occurrence of a sync word *other than* the flag byte.

It is possible to determine whether a data sequence contains a valid HDLC frame by using more detailed knowledge of the HDLC frame format. Specifically, error detection in HDLC is accomplished by appending a 16-bit frame-check sequence to the body of the frame [6]. Then a bit-stuffing operation is performed before the flag byte is attached to the beginning and end of the frame.

Bit stuffing prevents the spurious occurrence of the flag byte within the body of the HDLC frame. The flag byte contains a sequence of six consecutive ones. Thus, each time a sequence of five consecutive ones occurs in the frame body, a zero is stuffed. For example, “0111111” becomes “01111101” and “01111110” becomes “01111100”. This operation is easily reversed at the receiver. Note that bit stuffing affects the frame-check sequence as well as the user data.

To determine if a data sequence contains a valid HDLC frame, it is searched for two consecutive flag bytes. The data between these bytes is then destuffed and checked for the presence of a properly computed frame-check sequence. If the FCS is correct,

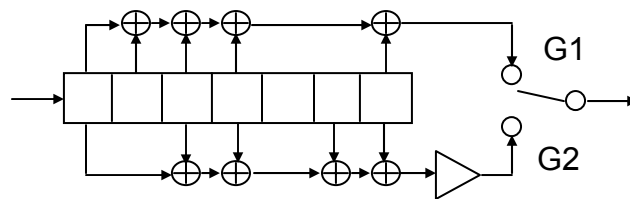
then the sequence is declared to be valid HDLC data. Also, note that during idle periods, when there are no user data to transmit, a continuous sequence of flag bytes is transmitted instead. Thus, a sequence is also declared as valid HDLC data if it contains several consecutive flag bytes with no user data.

### **3.5 Estimation Algorithm**

To study the approach discussed in the previous section, we have developed an algorithm that uses frame-structure information to recover transmission parameters for a data sequence that might be transmitted on a space-to-ground link over the Multiple Access service. The transmission parameters of interest are the data-format suffix (-L, -M or -S), the convolutional code rate, the node synchronization, and the inversion pattern (see discussion below). These are some of the parameters required by the ground-station receiver in order to correctly demodulate and decode the received signal. The data rate and the format type (NRZ or Bi $\Phi$ ), which were discussed in Section 3.2, are assumed to have been estimated already.

As discussed in the previous section, the data-format suffix may be determined by demodulating the received signal as if it were NRZ-L or Bi $\Phi$ -L, and then determining if the resulting sequence is differentially encoded. For the Multiple Access service, data may be transmitted without error-correction coding, or with convolutional encoding at rate  $\frac{1}{2}$  or rate  $\frac{1}{3}$  [2]. To decode a data sequence properly, it is also necessary to know the generator polynomials that were used by the convolutional encoder, but since only one set of polynomials is specified for each coding rate used in the Multiple Access service, it was not necessary to make a separate determination of these.

If a sequence has been convolutionally encoded, though, the Viterbi decoder must know which bits were produced by each polynomial. Figure 3-4 shows the operation of the convolutional encoder. Unencoded bits are clocked into a seven-bit shift register from the left. These seven bits are then combined using modulo-2 adders. The adders shown above the shift register represent the operation of polynomial “G1”, and the adders below represent the polynomial “G2”. Bits produced by G1 and G2 are clocked out alternately onto the output stream. For a rate- $\frac{1}{2}$  encoder, two bits are clocked out every time one bit is clocked in. Note that often the output from G2 is inverted, as shown in the figure.



**Figure 3-4 – Convolutional Encoder**

It is clear from the above that the data presented to the Viterbi decoder consist of a sequence which alternates between G1 bits and G2 bits. When we begin to demodulate the data, we will not know if the first bit we receive is from G1 or G2. The process of determining this is called *node synchronization*.

As was mentioned previously, Figure 3-4 shows that the G2 bits are inverted before they are clocked out. This is done to avoid long strings of 1's or 0's in the output stream. It is typical in convolutional encoding to invert the bits from one or more

polynomials. The specification for which polynomials are to be inverted we call the *inversion pattern*. It is of course necessary to know this in order to decode the signal properly.

The algorithm that we have developed estimates all of these parameters using frame-structure information, as discussed in the previous section. We assume that the satellite transponder transmits data using either HDLC [6] or one of the CCSDS data-link protocols [7]. We decode the received data sequence under all the possible sets of assumptions regarding the values of the parameters we are trying to estimate. The various output sequences are stored in a table, and then each sequence is checked to see if it conforms to one of the possible frame formats. A conforming sequence is assumed to have been decoded correctly. Thus, we conclude that the parameter values used in decoding it are correct, and these become our estimates for reconfiguring the receiver.

### **3.6 Testing**

We have developed a Matlab simulation that exercises the algorithm described in the previous section. Under each of the possible frame structures, the simulator generates 1000 random data sequences encoded under each of the possible assumptions about the transmission parameters. Each time a data sequence is generated, it is fed to the parameter-estimation algorithm. The estimates returned by the algorithm are compared with the assumptions used in encoding the sequence. If one or more parameters were estimated incorrectly, an error is recorded, and the simulator state giving rise to the error is logged.

The random data sequences are generated using simple models of the assumed protocols. When the CCSDS protocols are assumed, for example, the sequences consist of a fixed number of frames, each with 200 bits of random user data prefixed by an appropriate sync word. The first frame of each sequence is truncated at a random point, so that the estimation algorithm does not know the starting position of the data. The HDLC frames are generated similarly, except that a frame-check sequence is appended to the user data, and then a bit-stuffing operation is applied, as described in Section 3.4, before the flag byte is attached.

The main goal of our test program is to determine the performance of our algorithms under realistic conditions, with noise-induced errors in the demodulated bit stream. Our initial testing was done in a noise-free environment, though, in order to obtain an initial indication as to the viability of our approach, as well as to aid in tracking down programming errors. The transmission parameters we are estimating are as follows:

1. Data rate
2. Data format: NRZ-L, NRZ-M, NRZ-S, Bi $\Phi$ -L, Bi $\Phi$ -M, Bi $\Phi$ -S
3. Convolutional coding rate
4. Inversion pattern (e.g., is the G2 bit inverted?)
5. Node synchronization

To test the performance of the estimation algorithm, we will execute Matlab-based simulations that assume the following parameter values:

1. **Data-link protocols:** HDLC, CCSDS-TC, CCSDS Proximity-1, CCSDS-TM
2. **Number of frames:** 5-10 (approximately 1000 to 2000 bits)
3. **Bit error rates:** 0, and  $10^{-4}$  to  $10^{-2}$
4. **Data rates:** 1-10 kbps
5. **Data formats:** NRZ-L, NRZ-M, NRZ-S, BiΦ-L, BiΦ-M, BiΦ-S
6. **Convolutional coding rates:** initially uncoded and rate  $\frac{1}{2}$ ; later, rate  $\frac{1}{3}$  will be added.
7. **Inversion pattern:** with and without inversion of every other bit
8. **Node synchronization:** Nodes tested equals the inverse of the coding rate.

Our figure of merit is estimation error rate; that is, the percentage of received signals for which one or more transmission parameters were estimated in error. This error rate may depend upon which data-link protocol was used. For each set of parameters simulated, we shall choose the protocol with the worst performance as the basis for the measured error rate. This testing will be **completed by May 14**.

**As of March 28**, we have completed simulations assuming noise-free conditions.

These involved 1000 repetitions each of the following parameter values:

1. **Data-link protocols:** HDLC, CCSDS-TC, CCSDS Proximity-1, CCSDS-TM
2. **Number of frames:** 5
3. **Bit error rate:** 0
4. **Data rates:** 1-10 kbps
5. **Data formats:** NRZ-L, NRZ-M, NRZ-S, BiΦ-L, BiΦ-M, BiΦ-S
6. **Convolutional coding rates:** uncoded and rate  $\frac{1}{2}$
7. **Inversion pattern:** with and without inversion of every other bit
8. **Node synchronization:** Nodes tested equals the inverse of the coding rate.

Under these noise-free conditions, the data rate was always estimated to within the resolving power based on the sampling rate of the analog waveform, and the other parameters were determined without error. This indicates that the algorithms are working as expected. The noise-free testing helped us to isolate and eliminate causes of errors that we had observed during late summer and early spring. These included, e.g., cases in which invalid data sequences were mistakenly declared as valid. This problem was corrected with minor changes to the estimation algorithm.

In addition, we have preliminary results from simulations in which noise-induced bit errors were modeled while estimating the convolutional encoding parameters and the data-modulation suffix (L, M or S). These parameters may all be estimated from the demodulated digital bit stream, and do not need to examine the sampled analog waveform. The preliminary indication of these simulations is that, with about 1000 bits of received data, a noise-induced bit-error rate of  $10^{-4}$  produces an estimation error rate of about 1%. This is reasonable because, at this error rate, we expect a bit error in

about one out of every ten data sequences. Note that a single bit error will not always result in an estimation error.

### **3.7 Year-Two Work Plan**

Our work plan for the second year of this project includes the following items:

1. Installation and testing of at least one parameter estimation module in the Crosslink Simulator at Goddard Space Flight Center (July 31).
2. Use results of GSFC to refine algorithm and user interface (September 30).
3. Adapt algorithms that examine the sampled analog waveform to perform well in the presence of Gaussian noise (December 31).

A good choice for an initial module to test on the Crosslink Simulator would be the algorithm for determining the data-format suffix (e.g., NRZ-L vs. NRZ-M). It is a simple module that would provide useful information. It would also help the NMSU team to gain familiarity in adapting Matlab modules to the simulator hardware. Testing for items 1 and 3 above would involve iterations similar those described in the previous section.

### **3.8 Dissemination of Results**

Most of the results described in this report have already been published in the following conference papers:

P. De Leon, Q. Wang, S. Horan, and R. Lyman, "A Design for Satellite Ground Station Receiver Autoconfiguration," *International Telemetry Conference*, Las Vegas, October 2003.

R. Lyman, Q. Wang, P. De Leon, and S. Horan "Transmission Parameter Estimation for an Autoconfigurable Receiver," *IEEE Aerospace Conference*, March 2004.

### **3.9 References**

- [1] S. Horan, *Introduction to PCM Telemetry Systems*, 2<sup>nd</sup> ed., Boca Raton, FL: CRC Press, 2002.
- [2] *Space Network Users' Guide*, Rev. 8, Mission Services Program Office, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2002.
- [3] R. L. Peterson, R. E. Ziemer, and D. E. Borth, *Introduction to Spread-Spectrum Communications*, Englewood Cliffs, NJ: Prentice-Hall, 1995
- [4] P. DeLeon, Q. Wang, S. Horan, and R. Lyman, "A Design for Satellite Ground Station Receiver Autoconfiguration," *International Telemetry Conference*, Las Vegas, October 2003.
- [5] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3<sup>rd</sup> ed., New York: McGraw-Hill, 1991.
- [6] U. D. Black, *Data Link Protocols*, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] *TM Synchronization and Channel Coding*, CCSDS 130.0-R-1, Consultative Committee for Space Data Systems, 2002.