USING LABVIEW TO DESIGN A FAULT-TOLERANT LINK ESTABLISHMENT PROTOCOL

Stephen Horan and Giriprassad Deivasigamani
Telemetering Center
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM 88003-8001

ABSTRACT

The design of a protocol for a satellite cluster link establishment and management that accounts for link corruption, node failures, and node re-establishment is presented in this paper. This protocol will need to manage the traffic flow between nodes in the satellite cluster, adjust routing tables due to node motion, allow for sub-networks in the cluster, and similar activities. This protocol development is in its initial stages and we will describe how we use the *LabVIEW Sate Diagram* tool kit to generate the code to design a state machine representing the protocol for the establishment of inter-satellite communications links.

KEYWORDS

Satellite cluster communications, ad hoc routing protocols, protocol development, protocol testing, and LabVIEW

INTRODUCTION

This paper describes the initial work performed to develop a link establishment protocol for a network of satellites forming a cluster. This link establishment algorithm will expect that the inter-satellite channel will be unreliable and will, therefore, need to take channel errors into account in the decision making process. The satellite cluster problem is similar to that studied in terrestrial mobile ad hoc networks (MANETS). 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 unnecessarily cause links to be marked as broken or nodes unreachable.

There are competing methods for the routing in ad hoc networks that are being considered by other research groups. The two that are similar, in some respects, to the one considered here are Ad hoc On-Demand Distance Vector (AODV) routing and Dynamic Source Routing (DSR). The AODV protocol is described in [2] and [3] while DSR is described in [3] and [4]. Some of the main differences are between AODV, DSR, and the approach chosen in this study are:

1. AODV and DSR assume no *a priori* node information while our approach assumes that the network may be pre-seeded with participating nodes,
2. AODV and DSR assume that the networks will be open while our approach assumes that network access will be limited to "trusted" nodes,
3. AODV and DSR use routing caches for routing information while our approach uses a routing table to hold the routing information,
4. AODV and DSR obtain routing information in an on-demand manner while our approach keeps the routing information in a Routing Table,
5. AODV and DSR can drop a link due to a single link error while our approach will not declare a link to be down until after several messages have failed,
6. DSR send all of the required routing information in the data package header, while AODV and our approach requires that all intermediate nodes have sufficient path information locally,
7. AODV and DSR assume that the inter-node link range is < 500 m (802.11-type link) while the satellite cluster algorithm needs to cover in excess of 1000 km.

For nodes in a satellite cluster, several of the characteristics of AODV and DSR would seem to pose problems that a different approach could help remove. In particular, the cluster protocol should

1. Acknowledge that the satellite cluster network will have many, if not all, of its participating nodes known before launch so they can be seeded and not need to be discovered,
2. Do not remove routing information due to channel errors unless they pass a threshold,
3. Do not send whole route lists with every packet but only send routing information updates when the link state changes to keep the routing update bandwidth as small as possible,
4. Use low-bandwidth Heartbeat messages to probe for link failures but send them less frequently than is done on mobile networks where they may be sent approximately every second.

For ease of coding, the development of the protocol will not, initially, be directly executed in a high-level language such as C. Rather, we will take advantage one of two widely-used environments that can be used for the protocol development: *Matlab* and *LabVIEW*. 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.

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. The language choice for the initial software development was the National Instruments *LabVIEW*. One major disadvantage of the *State Diagram* toolkit is that the modules cannot be directly translated by the toolkit into a C-type of code representation. This limits the portability to those hosts running the *LabVIEW* environment.

PROTOCOL SPECIFICATION

Before generation of the state diagram for the modules was attempted, a detailed protocol specification was codified. The protocol specification details can be found in [5] and they can be summarized as follows:

1. Use a Routing Table pre-seeded with the expected cluster nodes and allow new, trusted nodes to be added later;
2. Use periodic Heartbeat messages to probe the channel for broken links and let the message interval be user-defined and only send the messages to neighbors within one hop;
3. Use a periodic Token passing mechanism to control access within a subnet under the assumption that the overall cluster may need to be partitioned because not every node may be visible to every other node;
4. Send Routing Table updates from a given node to its one-hop neighbors only when that node detects link connectivity changes or it receives better link information from one of its neighbors;
5. Use Cluster Heads to control Token passing within each sub-net where the Cluster Head is defined as that node in the sub-net with the lowest IP address that is one hop away from all members of the sub-net;
6. When a Cluster Head fails or moves away from a sub-net, the survivors determine the next Cluster Head by the one with the lowest IP address.

The detailed specifications were designed as a state machine for realizing the protocol. The top-level states are illustrated in Figure 1. In *LabVIEW* notation, this is the initial Virtual Instrument (VI) defining the protocol. In the INIT state, the user-defined parameters and initial Routing Table and State Table are built to define the protocol variables. Then each node determines if it is a Cluster Head or Cluster Slave based upon the node's IP address and location in the Routing Table.

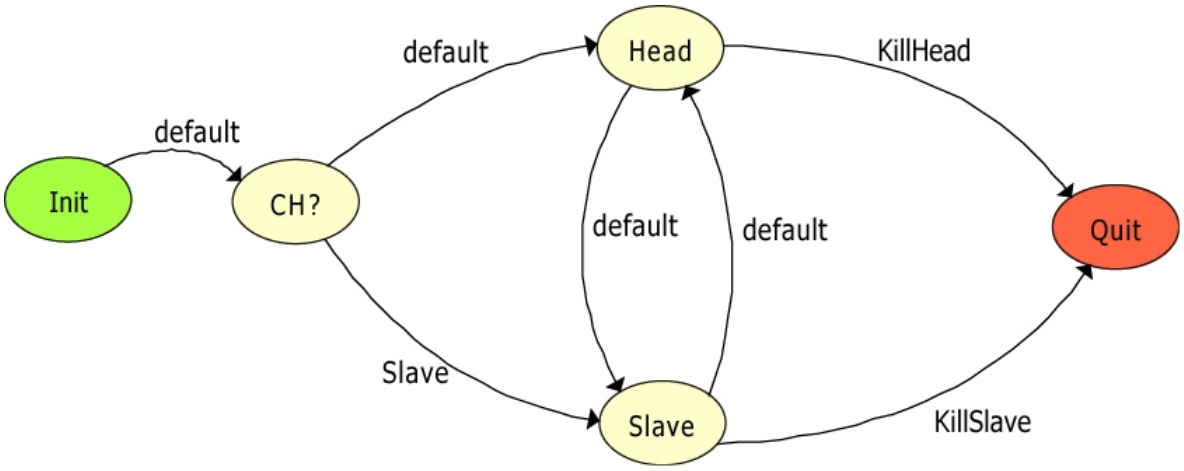

Figure 1 – Top-level states in the cluster link establishment protocol.

Each node then enters either the HEAD or SLAVE state and executes appropriate processing there. These states may be exited if a state change is detected, for example detecting the failure of the existing Cluster Head, or if the protocol received a management message to stop the protocol.

The Cluster Head and Cluster Slave have similar state diagrams that are realized as state machines as well and are called as sub-VIs from the main VI. The state diagram for the Cluster Head is given in Figure 2. After initialization, the Head and Slave enter a continuous loop. The basic structure is to

1. check for the presence of Heartbeat (HB) messages on the input port and process them if available;
2. check for the presence of Handshake (HS) messages, e.g. a Routing Table update and process them if available;
3. check for the presence of a Token message and process it if available;
4. check for time to issue a Heartbeat message and do so to the one-hop neighbors if it is time;
5. check for the time to issue a Token message and do so to the next entry in the Routing Table for the sub-net.

A Cluster Slave does not issue Token messages so the Token timing check is not part of the Slave VI states.

The individual states in the Cluster Head and Cluster Slave VIs can be made into VIs of their own with a finite number of states. This is illustrated in Figure 3 for the Process Heartbeat message state. In this VI, the Heartbeat message of processed and the Routing Table is updated. The Routing Table may also be transmitted to the one-hop away nodes if significant changes are detected as part of the Heartbeat message processing.

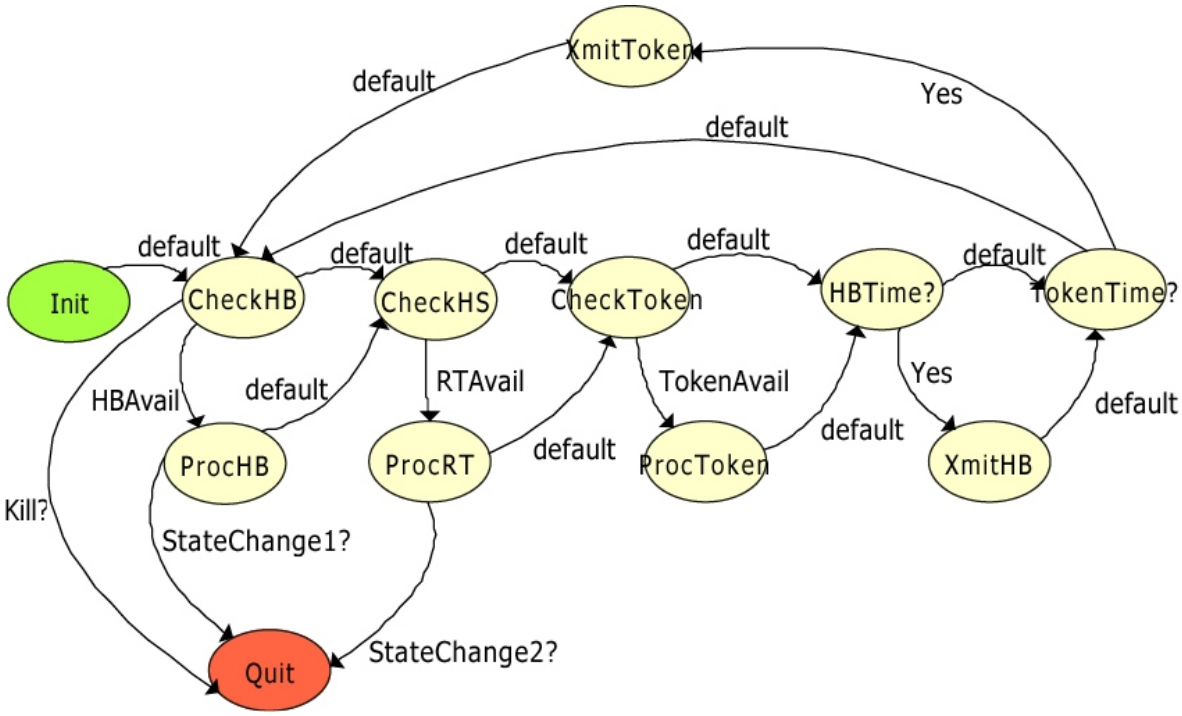

Figure 2 – State diagram for the Cluster Head state. The Cluster Slave state is identical except for the Token transmission timing.

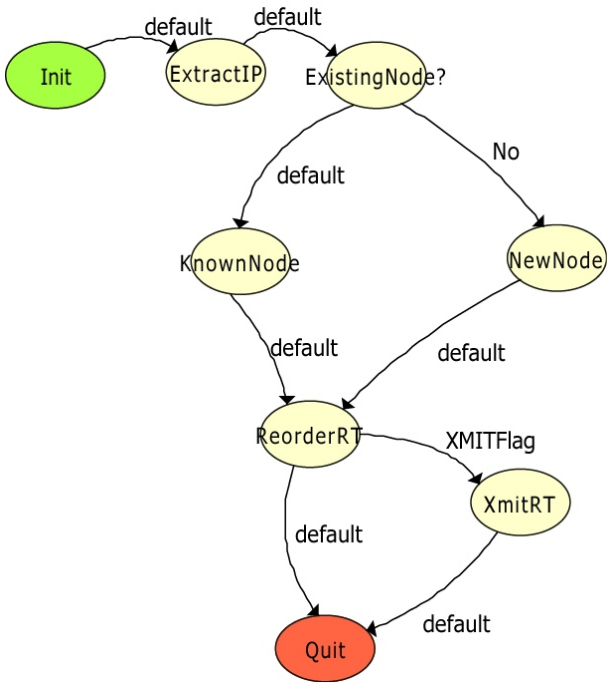

Figure 3 – The Process Heartbeat states within the Cluster Head or Cluster Slave VIs.

From this point on, it is frequently possible to encode all of the state processing within a single-state sub-VI rather than making further refinements to the state machine. This is a design decision for the protocol designer. The advantage the *State Diagram* toolkit brings is that the state diagrams can be developed quickly and then more time can be spent on the detailed processing modules. In the protocol software development, 26 modules were developed to program the protocol. Some of these modules perform the flow control between states within the VIs while others perform actual computations or state variable manipulations.

PROTOCOL TESTING

The software developed for the link establishment protocol was tested and the full description is given in [6]. The testing philosophy was to build the basic state variable structure, verify that it could be checkpointed to a disk file and recovered, and then add well-defined modules that built incrementally upon the successful development and testing of previous modules. This is where process reverses flow from the design stage. During the design, we tried to keep let the protocol logic dictate the state flow within the VIs and defer the detailed processing until as late as possible. Once the detailed processing modules are completed, they are tested at the unit level to ensure proper functionality and then integrated with other modules. A total of 11 test sequences were run to verify that the modules and VI control logic functioned properly.

RESULTS

The test program described in [6] was intended to validate the initial phase of the satellite cluster link establishment protocol. This initial development is intended to provide a basic functionality that can be further tested and refined. The capabilities demonstrated in this testing included:

1. The ability to use IP addresses as the means to control node designations as either a Cluster Head or Cluster Slave and have these designations based upon the current contents of the Routing Table.
2. The ability to use the contents of the Routing Table to
   a. determine the path of the Token through the Cluster members,
   b. determine which cluster members are to receive a Heartbeat message from each node,
   c. determine which cluster members have become inactive or unreachable.
3. The ability to transmit Heartbeat messages with a predetermined re-issue period to probe the cluster for unreachable members.
4. The ability to transmit Token messages with a predetermined re-issue period to allow nodes to control data traffic.
5. The ability to exchange Routing Table messages between the cluster members and update this Table based upon changing conditions.
6. Persistence in the transmission of Heartbeat messages until the time-out period is exceeded.
7. Persistence in issuing Token messages and nodes are not removed from the Token path until the time-out period has expired.
8. The ability of the Cluster Head to control the issuing of Token messages.
9. The ability of the cluster nodes to select a new Cluster Head if the original Cluster Head fails

or becomes unreachable.

With these capabilities, the initial software is ready to proceed to additional testing with channel errors added to the scenario as well. These types of tests will allow us to quantitatively compare the results obtained with this protocol against those with different routing approaches or with different cluster management techniques. Based on these subsequent tests, additional features can be added to augment performance.

CONCLUSIONS

The *LabVIEW State Diagram* toolkit can be used to generate VIs that are embodiments of state diagrams as well as having VIs that perform more traditional computational tasks. The largest advantage seen in this process was the ability of these types of state variable toolkits to be good vehicles for organizing the logic flow between states in the protocol and their abilities to be easily edited to modify the logic if flaws are found or if different approaches are desired. While this type of development could be performed in a high-level programming language such as C, the use of a graphical toolkit made the development process much easier. Because the toolkits also perform syntax checking as the code is developed, they are expected to have a quicker development cycle by eliminating those types of errors. For a practitioner well versed in the basic environment, albeit *LabVIEW* or *Simulink*/*Matlab*, these types of toolkits are to be considered for these "nontraditional" applications of the product.

ACKNOWLEDGMENT

This material is based upon work supported by the National Aeronautics and Space Administration under Award No. NAG5-13189. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

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.

[2] C. E. Perkins, E. M. Royer, I. D. Chakeres, "Ad hoc On-Demand Distance Vector (AODV) Routing," Internet Draft draft-perkins-manet-aodvbis-00.txt, October 2003.

[3] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," IEEE Personal Communications, Feb. 2001, 16 – 28.

[4] J. Broch, D. B. Johnson, and D. A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad

Hoc Networks," Internet Draft draft-ietf-manet-dsr-01.txt, December 1998.

[5] S. Horan and G. Deivasigamani, "Link Establishment Algorithm Development – Phase I," NMSU-ECE-04-004, Las Cruces, NM, May 2004.

[6] S. Horan and G. Deivasigamani, "Link Establishment Algorithm Development – Test Report," NMSU-ECE-04-005, Las Cruces, NM, May 2004.