

Protocol Design of Dual Ring PESNet (DRPESNet)

Jerry Francis, Jinghong Guo, and Stephen H. Edwards

Center for Power Electronics Systems
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061
Phone: (540) 231-5494, Fax: (540) 231-6390
e-mail: jerryf@vt.edu, jguo@vt.edu

Department of Computer Science
McBryde Hall, Virginia Tech, Blacksburg, VA 24061
Phone: (540) 231-5723, Fax: (540) 231-6075
e-mail: edwards@cs.vt.edu

Abstract – A new version of PESNet is presented that uses dual, counter-rotating fiber optic rings to address critical issues in earlier solutions for distributed power electronics control applications. The new protocol supports fault tolerance and network synchronization features. The protocol packet structure, commands and operating modes will be described and analyzed using an example application.

I. INTRODUCTION

The concept of Power Electronics Building Blocks, or PEBBs allows a modular approach to power electronics systems [1,2]. Prior work on these systems suggests the use of a control architecture that separates control algorithms from device specific implementations in order to move toward plug and play power electronics systems [3]. In this architecture, devices are connected in a daisy chained ring using plastic optical fiber, communicating via a protocol called PESNet [4]. The initial PESNet protocol design raised several issues, such as the lack of fault tolerance and dependence on a specific network topology, which prevents its use in some situations. This paper presents a new version of PESNet based on a dual ring solution to address the protocol's critical issues. The physical layer of PESNet remains unchanged, but additional capabilities have been added on top of it. This paper focuses on the protocol design at the network layer.

The basic structure of dual ring PESNet will be presented. The use of two counter-rotating fiber optic rings provides similar fault tolerance properties to Fiber Distributed Data Interface (FDDI), a standardized dual-ring protocol based on the token ring model. PESNet has three working modes: configuration, normal operation, and fault tolerance mode. After the network powers up, it enters the configuration mode first. By negotiating a set of network parameters among nodes, the network is configured to meet specific application requirements. Then, the network can enter normal operation mode. If there is a node failure or link failure, the network will enter fault tolerance mode. Network behavior in each mode will be illustrated. The focus will fall on mechanisms for fault detection, fault reporting, and fault handling. When the failed node or link is repaired or replaced, the network reverts back to normal operation. This "healing" process will be described as well.

Another important issue in the design of PESNet is that of synchronization between nodes in real-time applications. A global net clock is embedded in the protocol to serve as a synchronization reference. The implementation and use of the global net clock will be discussed, and its resolution and jitter will be analyzed.

The PESNet protocol is divided into a set of packet types that support basic network communication, dynamic node address assignment, network configuration, and fault tolerance. The protocol will be outlined, including the functionality of each type of data packet. The performance of PESNet in each working mode will be tested and the results will be shown.

Previous PESNet design is based on single ring daisy chain. If there is one node failure or link failure, the whole network will be out of function. To build fault tolerance capacity to PESNet, dual ring PESNet (DRPESNet) is proposed to tolerate one node failure or one link failure in the network.

II. BASIC STRUCTURE OF DRPESNET

DRPESNet is composed of two rings: a primary ring and a secondary ring. Packets are transmitted in opposite directions in the two rings. From an application point of view, there are two types of nodes connected to DRPESNet: master nodes and slave nodes. Normally, master nodes send out control information to slave nodes. Slave nodes send response back to master nodes. Each node has two pairs of transceivers and receivers. Each node can send data in opposite directions, as shown in Fig. 1.

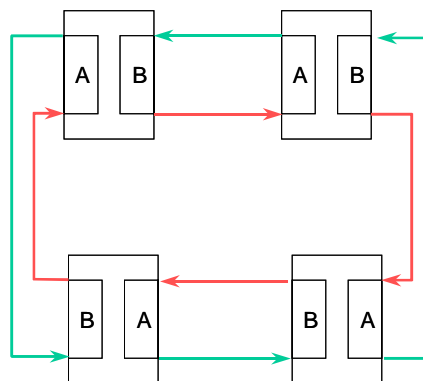


Fig. 1 PESNet topology.

If the fiber optic link is broken, the system should still be able to function. In this architecture, there are two types of failures. There are node failures and link failures. Other types of failures can be treated as one of these two. In a node failure, the node loses power, and ceases to communicate valid packets to at least one side of the failed node. In a link failure, one link in either ring is broken, preventing communication in one direction. In order to accommodate both of these conditions, a bi-directional ring is used. This ring allows the message to backtrack in the event that a node has failed. Figure 2 shows the two possible failures.

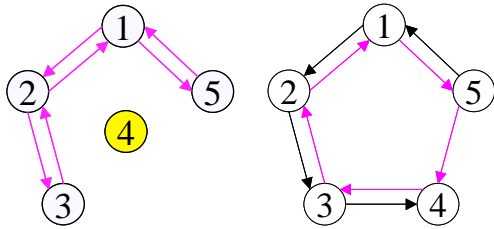


Fig. 2 Node Failure (left) and Link Failure (right)

DPRPESNet works in 3 modes: configuration mode, normal operation mode and failed mode. Only the primary ring will be used in configuration mode.

To simplify communication management at each node, each node sends out data in one direction during configuration.

During normal operating mode, the number of hops that a message must take to travel around the ring is N , where N is the number of nodes in the network. During a node failure, the number of hops that a message must take in order to completely travel around the ring will become $2N$. This includes two slots for buffered messages. Each message will be passed once on the outer ring, into buffers at the two end nodes, and then through the inner ring. The inner ring will be used solely for routing and receiving requests. All transmission is done on the outer ring. This allows the network to heal. When the complete double ring is restored, all commands are processed on the outer ring, and the inner ring is emptied of remaining commands, and the empty communications slots are replaced by null packets.

There is also fault tolerance for components of the packet header built into the protocol. Before these can be discussed, the concept of the network time and the net clock must be introduced.

Another issue that is important to this protocol is network topology independence. It is desirable to be able to insert and remove nodes from the network while the converter is running. Without the double ring, that is not possible. The double ring allows the packet to be routed around a link break during an insertion or removal, where the ring will be broken for some time, and then reconnected. When the number of nodes changes or the path that the information travels on changes, the

communication protocol must be able to account for the extra hops in the network. The previous version of PESNet relies on the network nodes being in a specific order for network synchronization. The new version of PESNet that is described here is based on scheduled events, and thus it allows for the network synchronization to be independent of the location and the number of hops between two specific nodes.

III. DATA PACKET FORMAT

A network packet begins with a TAXI command. This command, when encoded, is the same size as a byte of data. The next piece of information is encoded as data, and is the sender address. The sender address size is specified as one of the three network parameters, P_a . Following the sender address is the receiver address. The receiver address is followed by the network time. The network time is used for synchronization, and is described in more detail later. The size of the network time is a parameter for the network, P_n . If there is a faulted module, and the network is in the faulted state, the next field, the faulted address field contains the address of the node that has faulted. It is as big as one of the other two addresses. Following the faulted address is the data fields. These fields always come in pairs. There can be 2,4,6,8, 10, or 12 bytes in the data fields. The number of data bytes is specified as the third parameter, P_d .

The data packet is in the following format:



Each field is further explained in the following table.

Field	# of bits	Description
CMD	4	Command
SrcAddr	$N \cdot P_a$	Source node address
DestAddr	$N \cdot P_a$	Destination node address
NetTime	$N \cdot P_t$	Network global time
FltAddr	$N \cdot P_a$	Fault node address
D[...]	$N \cdot P_d$	Data
CRC	8	Cyclic redundancy check

The size of node address, network time and data are negotiable. Different applications require different amounts of information and different synchronization requirements. The size of the network packet places constraints on how fast the network can operate and how much data can be transmitted each switching cycle. For some converters that have high switching frequencies and lots of nodes, the packet must be small, whereas for converters that are switching slower, larger amounts of information can be passed between nodes in one packet.

To accommodate all applications, three network variables are defined to allow for different operating modes. For example, a converter switching at 40kHz with two controllers and 12 phase legs, all the new duty cycles must be transmitted within 25 microseconds to all the phase legs. This means that a packet must be transmitted in 2 microseconds. If it takes 80 nanoseconds to transmit one byte due to 4B/5B encoding, then a maximum of 22 bytes can be transmitted per packet during normal network operation to meet the timing constraints. However, during a fault, the number of hops will increase. Therefore, it will take double the number of bytes. During a fault, the message length can be only 11. This means that the maximum packet size is only 10 bytes due to constraints on the data field size.

In order to perform feedback control, the sensor and data values must also be transmitted back to the controller so that it can close the loop. The minimum size of the data is limited to the number of bytes that must be transmitted to the controller from the hardware.

During startup, the controller will initialize the hardware so that the network packet size is defined by parameters.

The four-bit CMD field gives only 16 possible commands. Eleven commands are defined so far.

NS_NULL (0x0): The network operates in lock-step mode. This means that there always must be a packet sent, whether it is meaningful or not. When a node does not have any data to transmit, it will transmit a NS_NULL packet. The data field in this packet is ignored, but the header still contains valid information such as the network time. A node may replace an NS_NULL packet with a valid packet containing other information. These are just placeholders that are used to maintain synchronicity.

NS_NORMAL (0x1): A node interacts with other nodes on PESNet by using data slots called attributes. Some attributes are read only and some are read-write. There are 255 attributes. The data fields in a packet of type NS_NORMAL correspond to attribute numbers 0x10 through 0x1F, depending on how large the data field is as defined by the network parameter for data size. Typically, this information is the new value of the duty cycle for a half-bridge module power module and the value of the network time to actually implement the new duty cycle. The first byte of this packet represents a key and a direction. The direction is 1 if the command is sent from the master to the slave and 0 if vice versa. When a master sends data to a slave, the slave responds with attribute information in range 0x20 to 0x2F or less if the data length of the packet data field is smaller. The key value is the same as when the node received the packet from the master.

NS_POLL (0x2): This is a command that obtains specific attribute values from a module. Each data value in the packet contains an attribute number. The value of the attribute is returned in a reply packet.

NS_CONFIGURE (0x3): This is a command that is used to modify values of another node. The data for this command comes in pairs. The first byte of the pair represents the attribute to change. The second byte of the pair is the new value to update the attribute with. If two fields have the same address, the first one is used, and the second one is ignored.

NS_INCREMENTAL (0x4): This command starts a block transfer of data. It is used when a lot of data has to be sent to a node, and that data is in sequence. The first byte contains the starting attribute number, and the second byte contains the data length. The remaining bytes are the data to transfer. The larger the data array is, the more effective this command will be.

NS_STATE (0x5): This command manages the state of multiple hardware modules. The first byte of the state command defines the network time at which the state transfer will happen. The second byte contains the type of state transfer. Typical state transfers are listed below.

ID	Name	Description
0x01	RESET	Perform hardware reset
0x02	SHUTDOWN	Shutdown module
0x03	STOP	Stop module operation
0x04	RUN	Set to run (from stop)
0x05	SET_MASTER	Set the module's master
0x06	PROGRAM	Set the module to program mode so that it can be updated.
0x07	SAFE	Enable module safe state
0x08	HOT_SWAP	Ready for hot swap-out

NS_COMMAND (0x6): This is an application specific command sent to the device. The contents of the data packet are specific to the application.

NS_EVENT (0x7): This packet originates from a slave node and is sent to other slave nodes or master nodes. These packets may be sent when abnormal conditions exist. One example is an invalid configuration. Another example is a new device on the network. A new node will use this packet with address zero to notify other devices that a new node exists on the network without an address.

NS_SET_PARAM (0x8): This packet is always a fixed size of 18 bytes. It is used to set the network type and assign node address after the network powers up. The first data field contains the first parameter, Pa, which is the number of bits in the address field. The second data byte, Pt, is the number of bits in the net clock. The third data byte, Pd, is the number of bytes in the data field. The fourth data byte contains information on the network encoded into a byte. This byte is broken up into bits as described below.

Bits	Meaning
7	Network fault tolerance 1-enabled. 0 – Disabled
6..4	Network Topology 000-Daisy Chain, Others reserved
3..2	Extended info length (in byte pairs)
1..0	Reserved

Extended info is the bytes following the network packet that are used for application configuration. For the present, these bits are reserved until further studies are done on how to best use them.

NS_NETCMD (0x9): This command has as the first byte more options. Eight bytes are defined for now. They are PING, SENDADDR, TIMEOFS, FORCECLK, CLRBUF, HWRESET, LINKFAULT, and REQADDR, NEWNODE. These are used only rarely, and are not used during normal operation.

NS_REQ_PARAM (0x9): This command is very different from the others. It has no data trailing it. The purpose of this command is to allow new nodes to request the network parameters Pa, Pt, and Pd from an adjacent configured node when it is just inserted or during network startup. The neighboring node will respond with a SET_PARAM command that will configure the new node.

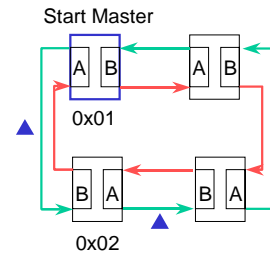
IV. CONFIGURATION MODE

The purpose of configuration is to assign a unique address to each node on the ring, negotiate systems parameters among all nodes according to application requirements and node capacity, after the network starts up.

Node address assignment

Before the network powers up, a master node is predefined as a start-master by setting a specific dip switch (“start-master” bit) on the Universal Controller board to “1”. If there is more than one master node on the ring, only one master node can be set as a “start-master”.

After the network powers up, all slave nodes and non-start-master nodes send REQ_PARAM packets. Each master node checks its “start-master” switch. The start master sends out a broadcast packet to assign address to each node on the primary ring. Three addresses are reserved for special purpose. Address “0x00” is reserved as the node address for “no network failure” in the FLTADDR field and is used as a broadcast address; address “0x01” is typically the address of the start-master. The data format is shown in Fig.3.



NS_NET_PARA	0x03	0x01	0x02	0x08	CRC
Command	Node address assigned next	Address size (byte)	Net time size (byte)	Data size (2-byte)	

Fig. 3 Node address assignment.

In order to assign the addresses during configure mode, the NS_NETCMD::REQADDR packet is used. The command field is set to 0x9, which indicates this is a NS_NETCMD packet. The second field contains the broadcast address, 0x00. Each node at this point will not have an address. The new node will take the address in the second data field, increment this value, rewrite it back to the second field, and send this packet to the next node.

Once the start-master gets the NS_NETCMD packet back, every node on the ring has been assigned a node address. The address size, net time size and data size fields contain the net parameters after negotiation among all nodes.

Start communication

After the system configuration is completed, the start master sends out a NS_NULL packet to both rings to start the network. Every node receives the NULL packet will change its transmission mode from uni-direction to both directions. Nodes will forward the NULL packet to both directions and listen to data arrival in both directions. At this point, the network is ready for use.

Configuration

There is a predefined configure-master in the DPESNet. The configure-master is responsible for configuring the network and each node. This configure master may be the same as the start master. Configuration packets are transmitted through the primary ring if possible. The NS_CONFIGURE packet will be sent to each node to set the application specific parameters and initial conditions.

V. NORMAL OPERATION MODE

During normal operation mode, each node sends out and listens to data on the primary ring, as shown in Fig. 4.

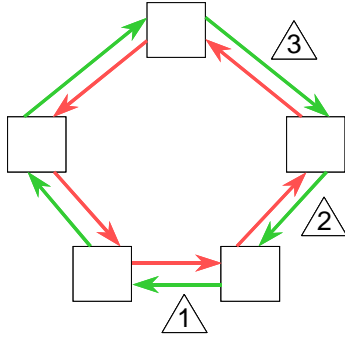


Fig. 4 DPESNet in normal operation mode.

If a node is the destination of a packet, this node will take the packet off the ring and replace it with a NS_NULL packet. If the packet is NS_NORMAL, this node swaps the SrcAddr and DestAddr field, and forms a corresponding NS_NORMAL packet to the sender to indicate its status.

During normal, fault-free operation, the worst-case delay from the master to a slave is:

$$\text{Delay} = (N-1) * (\text{Delay at each node})$$

The worst case delay from a slave to the master is:

$$\text{Delay} = (N-1) * (\text{Delay at each node})$$

During the normal operation mode, the FltAddr field should be "0x00".

VI. FAILED MODE

When there is a node failure on the ring, data will be rerouted through secondary ring. Except the failed node, the rest nodes will form a bigger single ring through the primary ring and secondary ring, as shown in Fig.5.

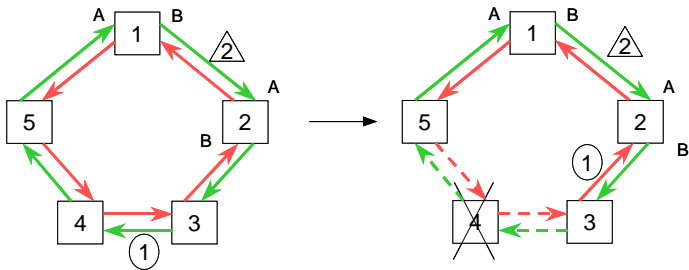


Fig. 5 One node failed in DPESNet.

During the failed mode, a node forwards any data not addressed to it on the secondary ring. A node will recognize any data with the source address of its own on the secondary ring as data that has traversed every node on the network and has not been received by the destination. A node will accept data addressed to it from both the primary ring and secondary ring. A NS_NULL packet will replace absorbed data.

Masters are responsible for determining which node on the ring is failed, and no data will be send to that node address after that.

During normal operation, the worst-case delay from the master to a slave is:

$$\text{Delay} = 2*(N-1) * (\text{Delay at each node})$$

And the worst-case delay from a slave to the master is:

$$\text{Delay} = 2*(N-1) * (\text{Delay at each node})$$

VII. FAILURE RECOVERED

When a failed node recovered, the network should be taken back to it operation mode. That means that eventually all traffic will eventually be on the primary ring.

Every node forwards packets not addressed to that node on the same ring it gets it. Every node discards data sent by itself from both rings and recognizes that the destination node is absent from the network. Every node can accept data addressed to it from both rings.

VIII. NETWORK SYNCHRONIZATION

In a power electronics converter, all phases must update the duty cycle calculated in the controller at the same time. If the duty cycles are changed asynchronously, then the output waveform will have harmonics in it that will cause noticeable distortion [3]. One benefit of this network topology is that the packet transmission occurs in lock-step mode. The packet transmissions are therefore synchronized. By maintaining a counter in the packet header, a clock can be created that allows for each node in the network to know some value that can be used for network synchronization. This counter is incremented every time the message is transmitted. In steady state, if all nodes have the same value of the counter, then every message will also have the same value in its header. Since the messages are coming from two directions, the two values of the network clock, as well as the previous value used by that node can be used to provide a fault tolerant mechanism to control the value.

If the network clock value is the same from both adjacent nodes, then the next value is one more than that value. If there is a difference between the two, and one value is one more than the previous value transmitted, then that is the next value.

Initially, the network clock value is zero. If the network clock is un-initialized, or the node is completely disconnected, then the clock will go to zero. Zero is the only invalid state of the network clock. The clock value can be forced from the NS_NETCMD::FORCECLK command. When this packet is sent, the network clock is updated in the node that received this command, and the command data is incremented and sent. The new clock value is used in communicating in both directions.

IX. IMPLEMENTATION

In order to implement the protocol in the past, AMD TAXI transmitters and receivers were used. Cypress has released the CY7C9689-AC chip, which combines the functionality of the two previous AMD chips into one chip. Two of these new chips have been placed on the Universal Controller board to allow for the dual ring.

The protocol will be implemented onboard the Universal Controller [5] using a pipelined architecture. VHDL code is being written to implement the PESNet protocol. A Xilinx Virtex XCV400 series FPGA has been selected to implement the protocol.

X. CONCLUSIONS AND FUTURE WORK

A flexible communications protocol tolerant of node and link failures has been presented. The protocol can manage high switching frequencies with many nodes. Future plans exist for this communications protocol as a place in a Plug and Play architecture for power electronics systems.

Future work involves the establishment of a high power test bed using the counter-rotating ring and hot-swapping power devices during converter operation. Additional information is available about the application or protocol.

ACKNOWLEDGEMENTS

This work is funded by the Office of Naval Research. This work made use of ERC Shared Facilities supported by the National Science Foundation under Award Number EEC-9731677.

REFERENCES

- [1] T. Ericson, "Power Electronics Building Blocks – a Systematic Approach to Power Electronics"
- [2] T. Ericson and A. Tucker, "Power electronics building blocks and potential power modulator applications," *IEEE Conference Record of the Twenty-Third International Power Modulator Symposium*, New York, NY, pp. p.12-15; 1998.
- [3] I. Milosavljevic, D. Borojevic., I. Celanovic, "Modularized communication and control structure for power converters," *8th European Conference on Power Electronics and Applications, EPE*, September 1999
- [4] Celanovic, Ivan, "A Distributed Digital Control Architecture for Power Electronics Systems," Thesis, Virginia Polytechnic Institute and State University, May 2000
- [5] J. Francis and D. Boroyevich, "Design of a Universal Controller for Distributed Control and Power Electronics Applications," CPES Seminar Proceedings, 2000