PORT TESTING IN A DUAL-STAR NETWORK: EMBEDDED LANS FOR RF*

Steven Groves (Harris PS&PC, Lynchburg, Virginia, U. S. A.; sgroves@harris.com).

ABSTRACT

In today's RF base stations, it is not unusual for a design to contain several microprocessors and/or DSPs, which need to talk to each other within a closed chassis, probably via an embedded LAN. In Harris's MASTR V public service base station, for example, the chassis is divided into removable modules, each with one or more CPUs and DSPs, whether the module is a receiver, a transmitter, a controller, etc, and they connect with each other on an embedded LAN. In critical applications, like public service, a wise approach to embedded LAN design would be to have two switches in a chassis, with each switch independently connected to all the modules in the chassis. That way, either switch can provide for the LAN in case the other switch is either down or has a connection failure to any module. This effectively creates two switches connected in a parallel "dual star" configuration. This is MASTR V's design (See Figure 1).

The dual star network then requires a software task: have exactly one active switch in operation at a time to avoid packet redundancies, and to change switches immediately if the active switch has issues. This would require all modules and both switches to always test the connection points, and ensure test information is transferred to both switches. Hence, a "packet forwarding" technique is developed for the dual-star embedded LAN as described in this article.

1. INTRODUCTION

Referring to Figure 1, by topological definitions, Ethernet is naturally a star network, which means that all PCs or network-capable components within a "star" send and receive data between each other with use of common devices [1], which in today's basic networks would almost always be switches. For obvious security reasons, embedded LANs are usually expected to be very basic like this, and closed to outside Ethernet traffic, and the network components would all be Ethernet-capable microprocessors and/or DSPs typically needing only one embedded switch to have enough ports to connect to all of them.

In systems needing reliability and redundancy, a good solution is to use a dual-star configuration, as also shown in Figure 1. In the embedded dual star, there are two switches, and the switches do not connect to each other directly. The reason is because the purpose of the dual-star is to have absolute "drop-in" switch redundancy with each switch independently connected to each component in the LAN, and therefore, needing each component to have two separate ports. This precludes having the switches connect to one another directly, as packet confusion may result. The port separation can be accomplished via a variety of ways, but in the MASTR V system, "mini-switches" are installed in the modules, which are less sophisticated than the backplane switches, but can be configured to isolate the ports going to both backplane switches to avoid packet feedbacks.

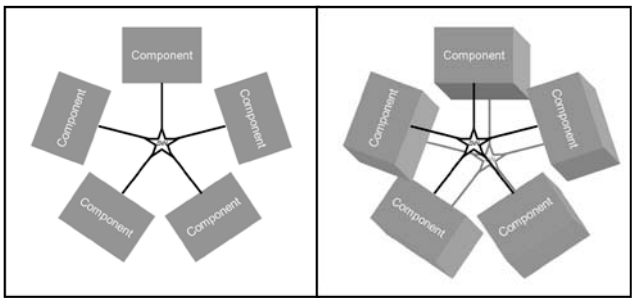

Figure 1. Single and Dual Star Network Topologies

1.1. MASTR V System

The MASTR V base station consists of a chassis with up to 14 hot-swappable slot modules and four external modules, and slots for the two highly-manageable embedded switches with microprocessors. A complete four-channel MASTR V system consists of a chassis containing four transmitters, four receivers, four processor-controlled RF and six controllers (two different types), and the chassis externally connects four RF power amplifiers. The two switches provide the dual-star network for all 18 external and internal modules.

It should be noted that the reason that there are six controllers of two different types is because one of the types of controls can be used for two channels each module. This controller type, called the baseband controller, controls the RF modules' settings and data for the two channels. The other type, called the traffic controller, routes a single

* Information in this paper is not export controlled.

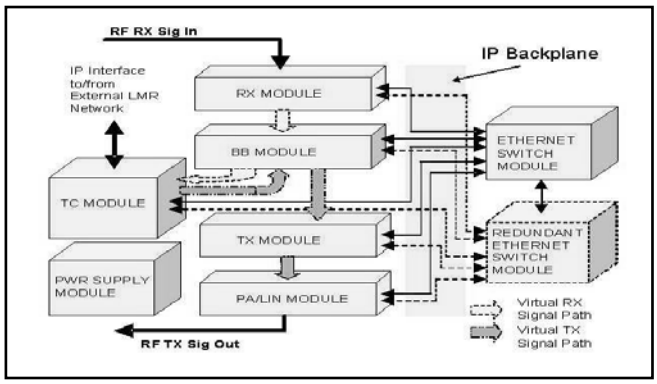

Figure 2. Diagram of One Channel of MASTR V System

channel's traffic (per controller) between other channels, and even other chasses via a WAN network, which is separate from the internal embedded network and has no route to it. For this design, the external WAN, which could span several sites, is not in a dual-star formation, as dual-star is primarily designed for router-less embedded LANs.

Referencing Figure 2, all modules, external and internal, connect two Ethernet connections to the backplane for the internal dual-star backplane switches, with each one of the two connecting to one of the switches. The backplane switches themselves are I. C. boards of a different form factor than the slot-based modules, but are slide-based hot-swap devices just like the other slot-based modules.

To have multiple ports on modules which may only want to dedicate one Ethernet MAC device, each module has its own internal switch, as described above, albeit a much less sophisticated embedded switch than the dual-star switches. This "mini-switch" ties a processor's MAC to the ports going to the backplane switches. However, to keep traffic from routing from either backplane port to the other, special configuration is needed for these "mini-switches." For the purposes of explaining the configuration, these configured "mini-switches" can be thought of nothing more than a "splayed" port, in which traffic from either backplane switch is treated as if it were coming from one single port and is destined for the module, and traffic coming from the module goes out both backplane ports as if both backplane ports were the same port. The "mini" switches are configured by the module's processor.

The microprocessors in the main backplane dual-star switch boards can configure the dual-star switches to do advanced traffic filtering as needed. This way one switch can block all traffic except for traffic to and from the processor so that it can be placed in standby, while the other switch stays active, to avoid packet duplication.

1.2. Software Requirements

Now we get to the software requirements. The primary requirement is to have exactly one switch providing the

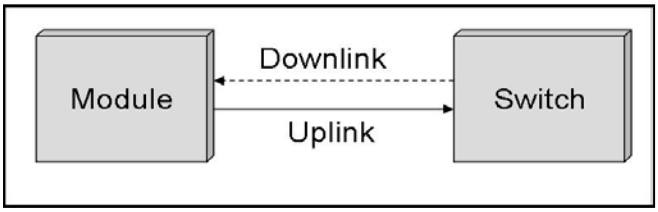

Figure 3. Definition of Uplink and Downlink

network and one switch configured to block traffic so as to avoid continuous packet duplications. There needs to be a way to detect if a port on the active switch has a connection issue with one of the backplane switches, or if the active switch has gone down, so that the other switch can take over. Both switches need to know about this port connection issue close to the same time so that they can switch roles quickly. They also need to know the difference between a bad port on one module, and a vacant module. Finally, there must be a small overlap when the switches change roles so that there is absolutely no packet loss, if avoidable, and to at the same time, minimize packet duplication.

2. WHY NOT SPANNING TREE

The first idea that probably comes to mind to most network experts is why not use spanning tree (RSTP)? Spanning Tree is a Layer 2 protocol used for preventing packet storming [2], and is used in just about every external off-the-shelf switch and router. The dual-star format used by the MASTR V uses mini switches to splay traffic to and from both switches, so someone might argue that it may be worth a try to manipulate RSTP protocol among both types of switches so as to accomplish the goals of the design.

The answer is the mini switches are controlled in the MASTR V by various types of processors with different commercial Ethernet controlling software, some of which are not readily able to use custom types of Layer 2 protocols. The second reason is that it the MASTR V system needs to signal the user to replace a bad switch or module, and RSTP is limited in how it detects and reports network issues.

3. PORT FORWARDING

A Layer 3 alternative is for the modules and the backplane switch processors to use a single, custom UDP packet, which acts as both the test packet and the information packet to relay to all modules and both switches. That approach would require that the backplane switch processors have active Layer 3 connections to the network and therefore be network components themselves. One should keep in mind that the backplane switches have microprocessors, which themselves are connected to their

respective switch and are therefore network components. Therefore, the switches themselves can receive and transmit data to and from the modules like they were one of them. Each switch even has a separate IP and MAC address. What should be clarified is that the switches cannot directly send packets to each other and need the modules to relay information to each other.

The custom UDP packet used in the port forwarding procedure is a packet containing the connection information for the LAN, which is a bitmap of passing and failing connections between each module and switch. It should be noted that today's networks use duplex connections and a connection fault may be direction specific (Figure 3), and therefore the packet contains one bit for each direction to and from each switch resulting in four bits per module (Figure 4). The connection directions from module to switch are called uplinks, and connection directions from switches to module are called downlinks.

Each module has four bits within a field of the UDP packet to represent the status of the uplink to one switch, downlink from the one switch, uplink to the other switch, and downlink from the other switch. A port failure between a module and a switch may either be an uplink, a downlink, or both. If the module is showing negative in both directions on both switches, the network can assume that the module is vacant or has been removed. If all modules show port failures in both directions for a switch, then it can be assumed that the switch is removed or down.

This UDP packet is constantly being transmitted by all modules to both switches, and to all modules from both switches, and is identical in structure for all transmitting members, but each bit within the packet is either manipulated by the source to indicate an issue, or forwarded by the source from information it received from a previous source, depending on if it is a switch (microprocessor) or a module. Both switches and all modules transmit this packet at an identical rate, say for arguments sakes, period P, and transmit them every period P regardless of when each switch or module may or may not receive a packet from another member. Each module independently unicasts a port forward packet to both switches (each switch individually), every time period P and, each switch independently unicasts a packet of this kind at the period to all modules (one-at-a-time), regardless of whether or not the module is vacant.

With the timing of the transmitted packets independent of each other, there will most likely be skews between the times when one switch receives packets from each module, when a module receives packets from each switch. With this design, skewing is irrelevant because all packets which each switch or module expects to receive will be arriving within one cycle of its own transmit period P. However, for good margin, the timeout for overdue packets from any sender is set to 2P. As each module and switch is transmitting a packet, it is testing the port connections. Uplinks to

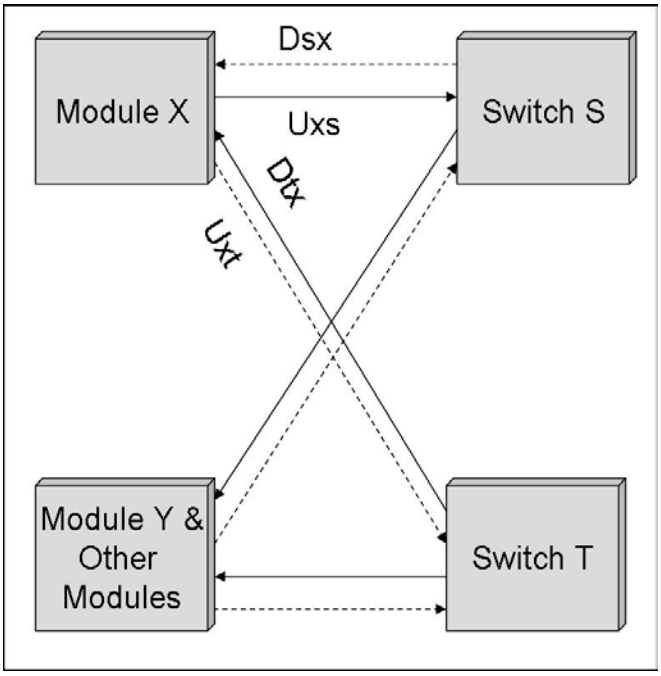

Figure 4. Diagram of Module Uplinks and Downlinks

switches are tested by the modules and downlinks to modules are tested by the switches. Information not known directly by a switch or module is forwarded from the packet(s) the switch or module received before it transmits. Even though there is no timing codependence in packet transmission between components, packet information effectively "flows" from module to switch, and then back to the modules, and back to the other switch. The best way to describe the process is to follow a pathway of information for a single module, step-by-step, as other bits of information are added in each step. The module whose information we want to follow is module X (See Figure 4).

3.1 Step 1: Switch S to Modules

If a switch has received a packet from a specific module, it marks the time stamp of its reception and flags that the "uplink from module X is passing." The approach used for this design is for the switch's processor to also carry an internal copy of the uplink/downlink fields in global memory of the packet to be transmitted and change this bitmap as necessary for the next time period P expires and it is time to transmit this data to all of the modules. That makes the fields to be transmitted most readily available at the time to transmit.

After period P expires, a switch checks the time stamps of the last time each module has expired, including module X. If a switch has counted two transmit periods and has not received a packet from a specific module within that 2P period, the switch marks the packet as overdue and sets the

correct bit within its internal bitmap to say "uplink from module X is overdue or is failing" implying that the port connection may have failed. For reference purposes, this uplink information bit is "Uxs." The switch is responsible for changing its packet for transmit to reflect the passing or failing uplink from each module. It is responsible also for forwarding other modules' information as will be explained later in the process, but to simplify for the next step, only module X's Uxs is referenced in this step. The summary of the step is: this switch is switch S and is sending information to all modules, including module Y, that S's uplink from X is condition Uxs. Module Y's role will be explained later.

Summary of Step 1
• Switch S provides uplink condition of X (Uxs) to module Y.

3.2 Step 2: Modules to Switch T

The modules receive packets from a switch S with uplink information on all modules. Each module is responsible for forwarding to the other switch T, uplink status between switch S and every module in the chassis except for itself. The information on a module's own uplink status to the switch S is moot because the process does not rely on a switch using a bad port to tell a module its port is bad, as a port failure may either be directional or bidirectional. Therefore, module X's information needs module Y, which represents a confirmed good module, to forward the information to the other switch. It should be noted that there needs to be at least two confirmed good modules in the system for this design to work, but a MASTR V chassis has a functional minimum of 5 modules, so there is sufficient single-failure redundancy. Module Y (and other modules) in the system receive Dsx from switch S. If module X received a packet from S, then Module X marks "downlink from S to X is good," or Dsx is good.

After transmit time period P has expired within each module, the module checks the time stamp of each switch's last packet and if it is greater than 2P outstanding, the switch's downlink to itself is marked as bad, in that "downlink to module X from switch S is bad," or Dsx is bad. If X's downlink from S is bad, the last uplink information it has from all other modules to S is moot because it is outdated. However, module Y is assumed good, and uplink condition Uxs is forwarding to switch T.

Summary of Step 2
• Module X provides downlink condition of S (Dsx) to switch T. • Module Y forwards uplink condition Uxs to switch T.

3.3 Step 3: Switch T to Modules

The other switch, switch T, now receives packet information from every module with good uplink connections to it. It then marks its internal memory with each good uplink. Module Y provides Uxs (along with the other good modules, of course), and module X provides Dsx. Uplink condition of module X to switch T (Uxt) is known directly by this switch, depending of whether or not it received a packet from X within the last 2P. Keep in mind that both switches have identical software and for the purposes of the test protocol, behave identically. The only difference is which fields they manipulate in the packet, and they are compliments of one another.

Summary of Step 3
• Switch T provides Uxt to module Y. • Switch T forwards Dsx to module Y from module X.

3.4 Step 4: Modules to Switch S

Modules Y and X are to now monitor switch T for packets. Module X knows first hand if it received a downlink packet from T within 2P and gives that information, Dtx, to switch S upon its time to transmit. Module Y forwards uplink information from X to T (Uxt) and downlink information from S to X (Dsx) upon its time to transmit.

Summary of Step 4
• Module X provides downlink info Dtx to Switch S. • Module Y forwards uplink info Uxt and downlink information Dsx to Switch S.

3.5 Step 5: Switch S to Modules (Second Loop)

At this point, S has the full network picture, but there needs to be two more steps to make switch T know the picture. The final situation is that it needs to pass Dtx back to Y, so that it can flow it forward to T. This process is in addition to the processes it performed before, and since we are assuming steady state, there is no change in the data from Step 1.

Summary of Step 5
• Switch S provides uplink condition of X (Uxs) to Module Y. This is same data as Step 1, assuming steady state. • Switch S forwards downlink Dtx to Module Y.

3.6 Step 6: Modules to Switch T (second loop)

Now Y must forward all it knows back to T, including Uxs, along with all its other requirements mentioned in Step 2. Module X continues what it's doing.

Summary of Step 6
• Module X provides downlink info Dsx to Switch T. This is the same data as step 2, assuming steady state. • Module Y forwards uplink info Uxs to Switch T, identical to step 2 assuming steady state. • Module Y forwards Dtx to Switch T.

3.7 Questions

Now both switches know the network status. This begs the question: what about the other modules besides Y and X? The answer is that all good modules will act as module Y, and, in steady state, will forward identical uplink information on X to both switches. Provisionally, a "majority-rules" for this design takes place in the switches, even though the data will be identical in steady state. The switches also wait for steady state before performing any changes. If there is a second bad module, the switches can detect "another bad X" the same way as before and can nullify its uplink data on other modules. The second bad module would be a compromise in the redundant switch design of the dual-star as it is expected that a user would replace a defective switch or module when there is a first failure. The goal is to simply nullify the first fault, and also recognize the second fault, should it occur, and make sure that the switches still provide for the modules that have no connection faults.

If someone concentrates on the process a little bit, they may ask if there is still another flaw in the design: A module X with a bad uplink cannot report that its downlink to the other switch is good or bad. The answer is that the switches assume a downlink from the opposite switch is good if it is unknown. A second fault on the same module is a compromise in redundancy and the pick up on this special "tunnel-through," fault where a module has only one good uplink to one switch and only one good downlink on the second switch, so as to simply make sure the network stays up for the sake of the uncompromised modules.

One more question is what to do with the fault information which the switches have. The switch processors which participate in the port forwarding procedure above also configure the switch hardware. The hardware uses advanced Layer 3 filtering and the switch deemed active, configures the hardware to allow normal traffic between all the modules. The switch deemed the standby switch only allows traffic between itself (its processor) and the other modules so that it does not duplicate packets while on standby. If the switch process discovers that the active switch has a bad connection, then the switches hand off their active/standby status to each other. The active switch becomes standby and the standby switch becomes active, while the network host gets notified of this fault so that the user can replace the switch or module which is defective. There is a small amount of overlap in active switches to absolutely prevent LAN downtime and at the same time minimize packet duplication.

Finally, what about software overhead, particularly in embedded software? It is definitely a drawback for software to need extra code to handle application-level network traffic. Having said that, the only concern is on the switch processor since it has to provide the filters, set up the switch, process test packets, and make complex decisions on what packet to send to all modules. The module simply needs to apply the latest information from switch A and switch B to the packet it will transmit, and update the connection status fields for which it is responsible.

4. TEST CASES

To verify the design, included are some test cases. They refer to the procedure in Section 3.

4.1 Module Removed

Create a scenario where, all of a sudden, device X is removed, as this is a hot-swap environment. Switch S will discover that the uplink Uxs is gone and will send this information to module Y (and all other modules), along with the assumption that Dtx is good (info from dependent module assumed good if module unable to send). Module Y forwards this information to T, but X is unable to uplink to T. T flags the bad Uxt uplink, and forwards assumed good Dsx downlink to Y. Y sends bad Uxt and assumed good Dsx back to S, and that info goes back to Y, though at this stage, this path is redundant because S now knows that there is at least an uplink fault in the module for both switches and can assume the module is removed or out of service. Though assumed-good Dtx is forwarded to T, T knows the same thing as S and therefore assumes module is removed or out of service. There is no fault or changing of the active switch.

- **Uxs, Dsx, Utx, Dtx are bad (=0)**
- **S: Uxs=0; Assume Dtx=1; -> Y**
- **Y: Uxs=0; Dtx=1 -> T**
- **T: Uxt=Uxs=0; Dtx=1; Assume Dsx=1 -> Y. T's path complete.**
- **Y: Uxt=0; Dsx=1 -> S**

- **S: Uxt=Uxs=0; Dsx=1; Assume Dtx=1 -> Y. Info complete.**
- **Switches assume X is removed.**

4.2 Port Down in Both Directions

What happens if X can no longer talk to S in either direction? Switch S reports uplink failure Uxs to module Y, along with assumed good Dtx, and module Y forwards all this to T. Module X reports to T that it has a downlink issue Dsx with S. T receives that issue and knows that it is genuine because it receives from X. Uxt is marked good by T and is sent to Y, along with the forwarded bad downlink condition Dsx. Y forwards good uplink Uxt and bad downlink Dsx to S. X cannot report to S about its good downlink Dtx from T, but S assumes good anyway, and forwards this back to Y. Both switches now know that X has a bad connection with S, and if S active and T is in standby, the roles need to reverse.

- **Uxs, Dsx, are bad.**
- **S: Uxs=0; Assume Dtx=1 -> Y**
- **Y: Uxs=0; Dtx=1 -> T. X: Dsx=0 -> T**
- **T: Uxs=0; Uxt=1; Dtx=1; Dsx=0 -> Y. T's path complete.**
- **Y: Uxt=1; Dsx=1 -> S.**
- **S: Uxt=1; Uxs=0; Dsx=1; Assume Dtx=1 -> Y. Info complete.**
- **Network switches to switch T.**

4.3 Port Down in One Direction

What happens if X can no longer talk to S in the uplink direction? Switch S reports uplink failure Uxs, and assumed good downlink Dtx to module Y, and module Y forwards this to T. Module X reports to T that it has good downlink Dsx with S. T receives that issue and knows that it is genuine because it receives from X. Uxt is marked good by T and is sent to Y, along with the forwarded good condition Dsx. Y forwards good uplink Uxt and Dsx to S. X cannot report to S about its good downlink Dtx from T, but S assumes good anyway, and forwards this back to Y. Both switches now know that X has a bad connection with S, and if S active, and T is in standby, the roles need to reverse.

- **Uxs is bad.**
- **S: Uxs=0; Assume Dtx=1; -> Y**
- **Y: Uxs=0; Dtx=1 -> T. X: Dsx=1 -> T**
- **T: Uxs=0; Uxt=1; Dtx=1; Dsx=1 -> Y. T's path complete.**
- **Y: Uxt=1; Dsx=1 -> S.**
- **S: Uxt=1; Uxs=0; Dsx=1; Assume Dtx=1 -> Y. Info complete.**
- **Network switches to switch T.**

4.4 Port Down in the Other Direction

What happens if X can no longer talk to S in the downlink direction? X is able to communicate to S that it has good downlink Dtx to S. Switch S reports good uplink Uxs to module Y, and of course, now-known status Dtx. Module Y forwards this to T. Module X reports to T that it has a downlink issue Dsx with S. T receives that issue and knows that it is genuine because it receives from X. Uxt is marked good by T and is sent to Y, along with the forwarded bad downlink condition Dsx. Y forwards good uplink Uxt and bad downlink Dsx to S. X reports good T downlink Dtx to S, and S forwards this back to Y. Both switches now know that X has a bad connection with S, and if S is active, and T is in standby, the roles need to reverse.

- **Dsx is bad**
- **X: Dtx=1 -> S.**
- **S: Uxs=1; Dtx=1; -> Y.**
- **Y: Uxs=1; Dtx=1 -> T. X: Dsx=0 -> T**
- **T: Uxs=1; Uxt=1; Dtx=1; Dsx=0 -> Y. T's path complete.**
- **Y: Uxt=1; Dsx=0 -> S. X: Dtx=1 ->**
- **S: Uxt=Uxs=1; Dsx=0; Dtx=1 -> Y. Info complete.**
- **Network switches to switch T.**

5. CONCLUSION

In today's multi-CPU designs for RF base stations in critical applications, which require closed embedded LANs, two redundant switches creating a dual-star network can be a desired approach. However, the software now requires every component and switch to be involved in determining which switch is non-functional on which port. While Layer 2 protocols like Spanning Tree may be incompatible with embedded development environments, a port-forwarding mechanism using UDP packets would be a possibility, if the port forwarding follows a set of procedures to ensure both switches know what to do under what circumstances.

[1] M. L. Young, D. Muder, D. Kay, K. Warfel, and A. Barrows, *Internet: The Complete Reference, Millennium Edition*, The McGraw-Hill Companies, Berkeley, CA 1999.

[2] M. Valentine, A. Whitaker, *CCNA Exam Cram Third Edition*, Que Publishing, Indianapolis, IN, 2008.