IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 2013, pp. 2297-2302, doi: 10.1109/IECON.2013.6699489.

Memory Requirements Analysis for PRP and HSR Hardware Implementations on FPGAs

J.A. Araujo, J. Lázaro, A. Astarloa, N. Moreira
Department of Electronics Technology
University of the Basque Country UPV/EHU
Bilbao, Spain

A. García
SoC-e (System-on-Chip engineering, S.L.)
Zitek Bilbao (ETSI)
Bilbao, Spain

Abstract— **The IEC62439-3 defines two ways to obtain a high availability automation network: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR). In order to do that, those methods include different paths to send information frames from source to destination and add a redundancy field to the frames. Nodes in the network must remember arrived frames so as to manage the duplicated information. In this paper the requirements of memory needed for a hardware implementation are analyzed.**

Keywords— Ethernet; availability; redundancy; HSR; PRP; IEC62439-3

I. INTRODUCTION

The use of Ethernet for industrial applications has being increasing for last years. This migration is linked to new requirements for Ethernet networks which are in need of the incorporation of higher restrictions looking to availability and synchronization.

The IEC62439 [1] series analyses different redundant methods depending on the recovery time: seconds, milliseconds, even 0s. The unique methods that offers zero time recovery, that is to say bumpless redundancy, are collected in the IEC 62439-3 Industrial communication networks – High availability automation Networks – Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR) [2].

One of the most important applications of those protocols is in substation automation networks based on the IEC61850 [3]. The part 90-4 (Communication Networks and Systems in Substations – Part 90-4: Network Engineering Guidelines) [4] chooses PRP and HSR for reliability in networks, and they are applicable to substations for Station and Process Buses.

In [5] architectures for substation automation systems are analyzed and designs related redundancy technologies such as PRP and HSR are discussed. In [6] different redundancy methods for substation automation are analyzed and PRP and HSR are presented for Station Bus and Process Bus respectively. HSR is presented in [7] as low-cost redundancy method for substation automation system; the paper analyzes redundancy in Ethernet and focuses on HSR solution.

Present paper firstly introduces the basic concepts of PRP and HSR redundancy protocols, then, it continues with the analysis of time-memory requirements to detect duplicated and circulating frames (they will be defined in following sections) and, after it, proposes and analyzes some solutions for those key points for the implementation of the standard.

II. STATE OF ART

Those redundancy methods which provide zero time recovery obtain high availability introducing more than one path (different and independent) to send information from source to destination.

PRP introduces two parallel networks to which nodes are connected. Those nodes named Doubly Attached Nodes (DAN) are linked to two networks. Source information is duplicated and sent through the two different networks; on the other side information is received duplicated (if no error happens). Destination takes the first frame and discards the one that arrives after, the duplicate, which arrives through the other link as sketched in Figure 1.

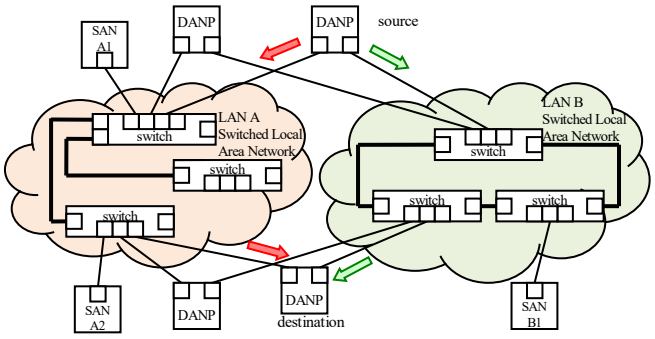

Figure 1. PRP network.

In case of PRP, Single Attached Nodes (SAN) are accepted and can be directly connected to one of the LANs, this element will only be able to communicate with nodes of that LAN.

There is another way to connect a SAN and it is by using a Redundancy Box (RedBox) which permits SANs to be connected to two LANs.

Meanwhile HSR uses two different paths using only one network. Basic topology for HSR is a closed ring as depicted in Figure 2. In this case the source node, Doubly Attached Node with HSR, sends information in two directions and those arrive to destination having traveled different paths.

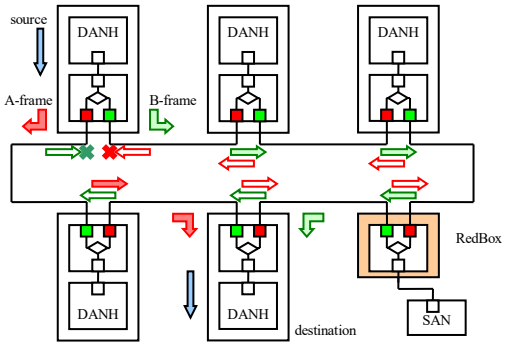

Figure 2. HSR Network.

In case of HSR, SANs cannot be directly connected to the network, HSR nodes not only receive frames, they must forward those frames for which they are not destination or multicast. This function is not implemented in SANs so that RedBox is completely necessary in this case.

Redundancy included by those protocols is at link level and is transparent for upper OSI layers. A Link Redundancy Entity (LRE) is included and is the element which manages PRP/HSR functions.

The approach to manage duplicate information is adding extra information to original frames. PRP adds a trailer at the end of the frame: Redundancy Control Trailer (RCT); and HSR adds a HSR tag just before addresses with a new ethertype. IEC62439-3 Ed.1 [8] defines a 4bytes RCT while the second edition [2] PRP-1 extends it to 6Bytes adding a new PRP-suffix (0x88FB). HSR does not change length of the tag from first to second version, but modifies the value of the ethertype (0x892F). Those modifications of the frames are depicted in Figure 3 and Figure 4.

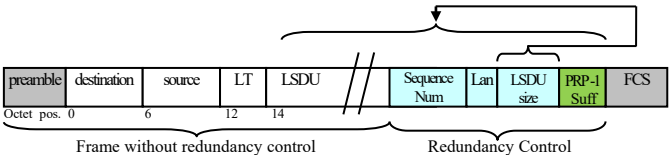

Figure 3. PRP frame.

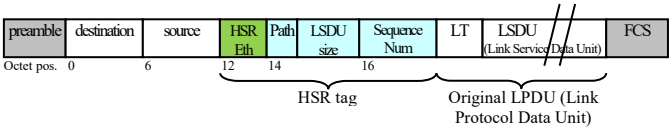

Figure 4. HSR frame.

The detection and elimination of duplicates is based on the Sequence Number, it is a 2Bytes field that appears in both, PRP trailer and HSR tag. This number is common to the two frames generated from an original frame delivered from upper layers and sent through the two different paths. Destination can detect if a frame has been received before or not looking in the frame the source and this sequence number.

HSR not only has to worry about duplicates, but also about another type of undesirable frames: circulating frames. If those frames were not eliminated they would produce the failure of the HSR network because of the never-ending circulation of them. The method to eliminate them will be based in remembering frames which already crossed a node in the network.

In PRP/HSR version 0 (IEC 62439-3 Ed.1) each node had a counter for each other nodes in the network, and when a frame was duplicated and sent through different LANs/Paths to one destination address, the value of the correspondent sequence counter was taken and added to frames in the field Sequence Number of the trailer/tag. That sequence number counter was then incremented for the next frame. Furthermore the destination node could detect duplicated frames just looking to source, destination and sequence number. Those three fields identified each frame and remembering which of them had arrived, duplicated and circulating frames could be deleted using an effective algorithm.

In the amendment 1 of the standard and in the second edition (version 1 of the PRP and HSR protocols) a great improvement has been done in frame identification. Nodes use the same sequence number counter for every destination, so that to identify a frame only source and sequence number fields are needed. This can significantly simplify duplicate and circulating frames deleting algorithms and reduces significantly the information to be remembered (42% per entry), and therefore memory requirements.

The standard does not specify algorithms, the unique condition is that a legitimate frame must never be rejected while occasional duplicates can be tolerated (Those would be treated by upper layers TCP/UDP…).

It could be thought as a simple solution to look at the sequence number from different sources to see if they arrived in increasing sequence number order. There is a problem to this approach since the standard states the fact that the sequence numbers of the frames sent by one source being not monotonically increasing is not a reason for discarding a frame. The standard also states that the duplicate discard method of PRP is not the preferred method for discarding duplicates in HSR because in that case circulating frames must be avoided too, and they must be taken into account [2].

Nevertheless in both cases the only way to know if a frame has arrived before, that is to say, if the frame is a duplicate or circulating frame is remembering arriving frames (source and sequence number), which port they arrived through, and when they arrived.

Another thing that has to be borne in mind is that in HSR ring frames must pass through every node; this fact adds a delay to frames from source to destination. So as to reduce that delay, forwarding frames is expected to do in cut-through mode, that is to say, just receiving destination address, source address and sequence number of a frame, the node can decide to forward it and to start forwarding it before the frame is

entirely received. In order to do the forwarding at such a high rate it should be done in hardware, and the most suitable way to do it is using FPGAs.

In the following section we are going to present an analysis to select an efficient way to remember this information in order to implement a method to eliminate duplicates and circulating frames in an FPGA.

III. TIME-MEMORY REQUIREMENTS

A. *Minimum time gap in which the sequence number may be properly repeated*

At most the last 65536 received frames (2^{16},16bits of the field Sequence Number seen before) should be remembered and taken into account; no more, because the next one with the same source and sequence number, could be a new frame and neither a duplicate or circulating frame.

On one hand, the minimum time in which a node can receive a frame with the same source and sequence number will be when a source node sends 65536 frames one after another and those frames are the smallest frames that can be sent in PRP/HSR networks. Figure 5 shows two nodes, one next to the other between which the maximum rate of frames could appear, it could represent a PRP (one LAN could be a direct connection, or not) or HSR configuration.

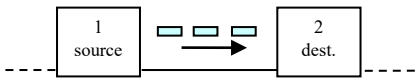

Figure 5. Minimum time in with one "source + sequence number" can be reused.

The smallest frame in PRP/HSR is fixed by the standard to 70Bytes, if a frame was smaller padding would be added.

And if the time of the preamble + SoF (Start of Frame), 8 Bytes, and the minimum gap between frame and frame, 12 Bytes duration, are added, that minimum time in which a source + sequence number can be properly received again is (1).

$$t_1 = \frac{65536\,frames \cdot (70+20)\frac{Bytes}{frame} \cdot 8\frac{bits}{Byte}}{100Mbps} = \\ t_1 = 65536 \cdot (5.6+1.6)\mu s = 367ms + 105ms = 472ms \quad (1)$$

Every time calculations are made for Fast Ethernet (100Mbps), if rate was 1000Mbps (Gigabit Ethernet), time results would be divided by 10:

Second edition of the standard states that entries must be forgotten in 400ms [2] avoiding in this way remembering sequence numbers that can be reused properly.

B. *Maximum delay between 2 frames in the worst case in an HSR network: Duplicates*

On the other hand, in HSR the maximum delay difference between a frame arrived through one port and the other is when the source and destination are one next to the other; besides this difference depends on the number of nodes, as sketched in Figure 6.

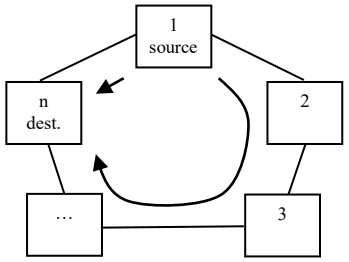

Figure 6. Time delay between frames with different path in the worst case in HSR.

The worst case, longest delay difference between the two frames through different paths, will be when:

- The longest frame is sent by the source node and,
- When this frame must wait to be forwarded in each node, between source and destination, because each intermediate node has just begun to send another frame (of maximum permitted length too).

The longest frame would be formed in the following way (2):

$$12B(addresses) + 4B(VLAN\_Tag) + 2B(Ethertype) + 1500B(data) + \\ + 6B(RCT\_trailer / HSR\_tag) + 4B(CRC) = \\ = 1528Bytes = Max\_length\_frame \quad (2)$$

And if the preamble + SoF and the gap between frames are added, in this case the delay difference of that frame will be (3):

$$t_2 = \frac{((n-1)-1)frames \cdot (1528+20)\frac{Bytes}{frame} \cdot 8\frac{bits}{Byte}}{100Mbps} = \\ t_2 = (n-2) \cdot (122.24+1.6)\mu s \quad (3)$$

For 50 nodes: t_2 =5967.52μs+76.80 μs = 5.94ms

For 512 nodes: t_2 = 62342.4 μs +816.0 μs =63.16ms

I.e. when a frame arrives, the duplicate expected through the other port will arrive in the gap of t_2 or it will not arrive, that is to say, a frame must be remembered at least t_2 to detect duplicate frames.

C. *Maximum delay between 2 frames. Extension to PRP: Duplicates*

In a PRP network, delay between frames arrived through one LAN and the other one will depend on elements and topology of each network. In an extreme case in which one LAN is a direct link between two nodes and the other one has some switches, nodes and so on, expected delay could be considered as the worst case in HSR having analogous delays, as depicted in Figure 7.

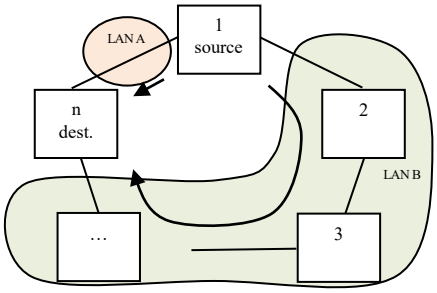

Figure 7. Time delay between frames in PRP.

In order to avoid duplicated frames, entries should be remembered at least t_2 as seen in the previous point.

For a given LANs (A and B) in PRP, time delay difference between ports could be estimated and calculated more accurately, but estimating an extreme case could be a good approximation.

D. *Maximum delay between one frame and the same frame after rounding whole HSR Ring: Circulating frames*

Calculations are similar to those in the previous case, but in this case it is multiplied by "n" because the frame gives an entire round to the ring as depicted in Figure 8.

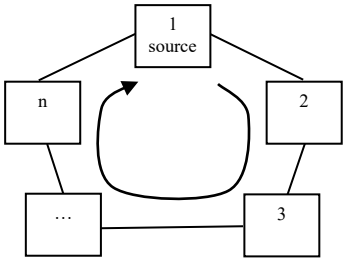

Figure 8: Time delay for circulating frame in the worst case.

So that, the time that a frame takes to round the whole ring will be (4), the preamble + SoF and the gap between frames is considered too:

$$t_3 = \frac{n \cdot frames \cdot (1528+20)\frac{Bytes}{frame} \cdot 8\frac{bits}{Byte}}{100Mbps} = \\ t_3 = n \cdot (122.24+1.6)\mu s \quad (4)$$

For 50 nodes: t_3 =6112µs+78.4 µs = 6.19ms

For 512 nodes: t_3 = 62586.88 µs +819.2 µs =63.41ms

This is only applicable to HSR because PRP networks have not the problem of circulating frames. Therefore, to avoid circulating frames, entries should be remembered at least t_3.

Moreover, having only those time requirements into account, the maximum theoretical number of nodes in an HSR ring would be when $t_3 = t_1$, giving a maximum number of nodes of 3811. In reality they are far fewer.

IV. Solution Proposals and Analysis

Discarding duplicate and circulating frames means remembering frames arrived to a node, and the way to do that is to store information about arrived frames, that is to say, memories are required.

A memory where arrived frames are stored sequentially one after another, and which must be completely run over to find data, is not suitable for this application in cut-through mode. It will take too much time to receive data (PRP/HSR) and to forward frames (HSR). For example, IEC61850-90-4 indicates a forward time for HSR in substations of less than 5µs [4]. Furthermore to know if a frame arrived before, a method to improve the search must be achieved and hardware solution must be used: FPGAs. Different memory solutions are going to be analyzed.

A. *Content-Addressable Memory (CAM)*

The first solution analyzed was a CAM memory or associative memory. Those memories can be addressed by content. The data you are looking for is compared with data stored in the memory to find it in one or few clock cycles. This kind of memories is used in switches with IP addresses.

The idea of a CAM memory is to achieve a good response rate to forwarding frames in cut-through mode for HSR. Reading a memory linearly to find data is in-viable for this mode and a better way must be implemented.

For each table, a capability of no more than 64k positions would be needed to store the worst case of one source sending 65536 frames one after another as in a circular buffer. In each position at least source address and sequence number (6B+2B=64bit) must be stored to remember that the frame has arrived before. If data are saved with the appropriate rate, entries will be automatically deleted/overwritten when corresponds and no time mark should be remembered for frame entries; if not, time marks should be saved too.

A way to solve this issue would be: when a frame arrives, source address and sequence number are taken:

- Frame id (source + seq. number) is stored one after the other in the order they arrive.
- Besides, we look up in the corresponding table to see if that frame arrived before. In this case two tables would be needed, one per each port. So that depending on what we are looking for, when a frame arrives we will look for circulating frames in the table of the port through which the frame has arrived (HSR when forwarding frames), and if we are looking for duplicates the table of the second port will be analyzed (PRP and HSR when receiving).

Figure 9 shows the way to write and read data:

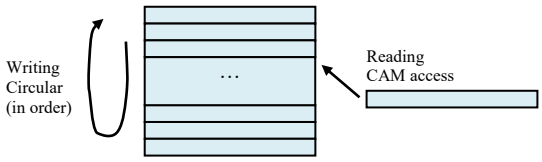

Figure 9. Circular writing and CAM access for reading.

At least two 64k64 CAM memories would be needed. The biggest problem with this solution is to implement so large CAM memories in a FPGA, for example, in application notes of Xilinx FPGAs much smaller memories does not fit in powerful FPGAs [9]. Apart of those memories smaller memory

or register would be needed for node tables (if implemented) and to store additional data.

Hence, despite the quickness of this solution it is not suitable to be implemented.

B. RAM memory (SRAM, SDRAM, DDR…)

Another idea could be to have a memory addressed by source (6 Bytes) and Sequence Number (2 Bytes). And in each position could be saved a time mark (and some extra information) with, for example, 16bits per position. This supposes a memory of 2^{8*8} positions = 2^{64} positions (Figure 10), beyond the current available memory sizes. Besides, memory should be run over periodically to avoid obsolete entries.

Furthermore, this memory would be extremely underutilized because it is never going to be 2^{48} MACs in use in a network at the same time.

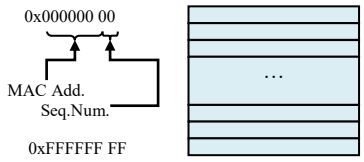

Figure 10. RAM Memory with 2^{64}positions.

This solution would require a too large memory to be used with an FPGA as seen.

C. Hash Tables

A way to increase the search efficiency and reduce memory size is using Hash Tables. The idea of hash tables is to apply a hash function to source address + sequence number, the result will be used as index to find the place in the memory to store the frame, or to find the frame if it has arrived before. Needed memory in this case is smaller than in the previous case. There are many algorithms to implement a hash table and resolve the collisions (when different source + sequence number have same hash). In [10] hash tables and circular frames were analyzed for HSR/PRP version 0 obtaining a combined method for networks up to 64 nodes.

In each position at least source address, sequence number and time mark must be saved and memory would need to be continually read to remove old entries, as see in Figure 11.

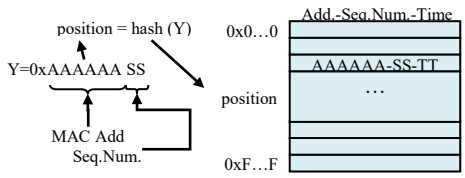

Figure 11. Hash Table.

In this case it is extremely important the chosen hash function and the size of the memory (occupation should be less than 75%) to reduce collisions to the minimum.

This solution could be appropriate, reduces the size of the memory and gives a better search rate of stored data, on the other hand it has the inconvenient of the hash function election, collision control and a suitable memory size.

D. Combination RAM + little CAM

In previous cases (A, B and C) source address and sequence number of the frames are used to address memories.

In case of B, if it is taken into account that not so many addresses are present in a network at the same time, the number of bits used to identify source can be reduced. For example if it is supposed that in the network no more than 512 MAC addresses (2^9), or less, are going to be used at the same time, the number of positions could be reduced to: $2^9 \bullet 2^{2*8}$positions=2^{25}positions = 32 Mpositions of 16 bits, this is a more feasible memory size.

Here the first step would be to transform the source MAC address in a smaller number of bits, from 48bits to 9; to do that a little CAM could be implemented, a 512x48 CAM. Then with those 9 bits plus the 16bits of the sequence number the position for the frame in the memory is available, in that position a time mark should be saved, as depicted in Figure 12.

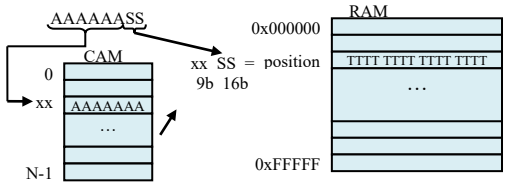

Figure 12. CAM + RAM.

This method requires more memory positions than hash tables but has no collisions, so that if the memory requirements are not too high, the way to implement this method could be simpler and quicker and moreover with no collision problem.

Finally, the last consideration is that frame entries must be forgotten in 400ms (as the standard states). Therefore, except in the case of the CAM in the other cases memories should be examined periodically to avoid an overflow. There will be a time counter, from which the time mark will be taken and with which time marks will be compared. So that, memory must be run over before time counter wraps around to avoid inconsistency in time marks and actual value of time counter. Another ways to have time into account could be considered.

V. Conclusions and Future Work

This paper presents an analysis of time and memory requirements for an FPGA implementation of the standard IEC62439-3. Hash tables and CAM+RAM are presented as the most suitable solutions. The memory requirements presented are normally available in the commercial boards of FPGA [11].

Between proposed solutions two of them could be appropriate for PRP/HSR implementation:

- CAM+RAM could be very effective because of its simplicity in front of hash tables despite the more memory positions needed which are not so disproportionate. Another advantage of CAM+RAM is that in parallel to the CAM there can be another RAM to save information about nodes. Version 0 [8] of the standard has node tables obligatory to store some information about nodes in the network, those tables are optional for the version 1 [2].

- Hash tables has the disadvantage of choose a good hash function and the treatment of collisions, but has the advantage of not to be designed for a maximum number of nodes, so that it can be a good solution for nodes without node tables. In this case memory must be oversized too.

The future work of this analysis is to analyze duplicate and circulating frames elimination methods, implement the selected options and probe that they function properly. With them nodes and RedBoxes can be developed.

ACKNOWLEDGMENT

This work has been carried out inside the Research and Education Unit UFI11/16 of the UPV/EHU and supported by the Department of Education, Universities and Research of the Basque Government within the fund for research groups of the Basque university system IT394-10 and the research program SAIOTEK (project S-PE12UN008).

REFERENCES

[1] *IEC 62439-1 Ed.01: Industrial Automation Networks - Part1: General concepsts and calculation methods,* IEC, 2010.

[2] *IEC 62439-3 Ed.02: Industrial Communications Networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and Hig-availability Seamless Redundancy (HSR),* IEC, 2012.

[3] *IEC 61850 Ed.2: Communication networks and systems for power utility automation,* Geneva: IEC, 2010.

[4] *IEC 61850-90-4 (draft): Communication Networks and Systems in Substations - Part 90-4: Network Engineering Guidelines.,* Geneva: IEC, 2012.

[5] J.-C. Tan and W. Luan, "IEC 61850 Based Substation Automation System Architecture Design," 2011.

[6] G. Antonovca, L. Frisk and J.-C. Tournier, "Communication Reduncancy for Substation Automation," pp. 344-355, 2011.

[7] H. Kirrmann, K. Weber, O. Kleineberg and H. Weibel, "Seamless and low cost redundancy for Substation Automation Systems (High availability Seamless Redundancy, HSR)," 2011.

[8] *IEC 62439-3 Ed.01: Industrial Communications Networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and Hig-availability Seamless Redundancy (HSR),* IEC, 2010.

[9] K. L. (Xilinx), "XAPP1151 (v1.0) (Parameterizable Content-Addressable - Application Note: Xilinx FPGAs)," Xilinx, http://www.xilinx.com/support/documentation/application_notes/xapp1151_Param_CAM.pdf, 2011.

[10] J. Xiaozhuo, High Availability Seamless Ring Protocol Inplementation in FPGA (Master Thesis), 2009.

[11] Xilinx Inc., [Online]. Available: www.xilinx.com.