“Etherblade.net – Ver.2”

(proposed architecture of the receiver)

This document is complemented with a video called “Etherblade.net Ver.2 (assembler block conceptual model demonstration)”.

Vladimir Efimov

https://www.linkedin.com/in/vladimir-efimov

(Jun. 2017)

The goal of Version2 of the project is to design a receiver block with "multiple_vlans/multiple_transmitters" and "fragmentation" support.

"Multiple_vlans/multiple_transmitters" support:

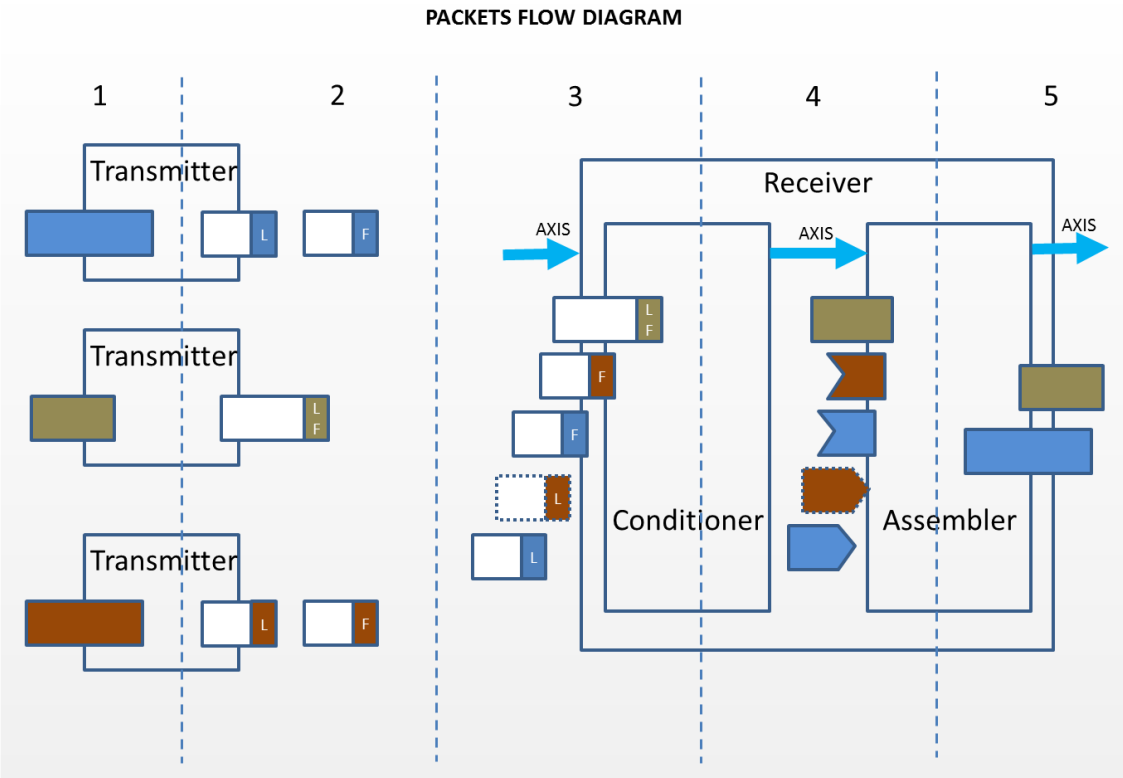
A single receiver shall be able to assemble fragments belonging to different vlans (pseudowire circuits) originated either by a single or different transmitters. This capability will make "point-to-multipoint" network implementations possible where a single central appliance installed in the datacentre can handle multiple vlans provided to different geographically dispersed clients. Internet service providers can benefit from "point-to-multipoint" model because alternative "point-to-point" model requires twice the number of the Etherblade appliances used in the network design.

"Fragmentation" support:

A receiver must have a capability to assemble original ethernet frames from received fragments. (This will allow the system to be used in any type of transport network with MTU size greater than 850 bytes).

Packet flow phases:

The following diagram shows flow of ethernet frames from transmitters (developed in Ver.1 of the project) to the receiver with both "multiple vlans" and "fragmentation" support in place. This diagram is split in 5 logical phases which an ethernet frame goes through.



Phase1 - the original ethernet frame comes to the transmitter (frame colour represents "vlan number"). Incoming frame is stored in the transmitter memory.

Phase2- transmitter encapsulates the frame by prepending it with “outer header” prior sending it out to the transport network. If stored frame’s length exceeds fragmentation threshold (1024 bytes) then the frame is sent as two equal size encapsulated fragments. Splitting frame in two equal size fragments doesn’t have a negative effect on jitter parameter and easily implementable in hardware. The “outer header” contains necessary information that can help assemble this frame on the receiver. The minimum sets of attributes required is: “vlan number” of the original frame, first fragment flag – “F-flag” and last fragment flag – “L-flag”. The first fragment will have “F-flag”-bit high in the outer header, the last (second) fragment will have “L-flag”-bit high, non-fragmented frame will have both “F-flag” and “L-flag” bits high.

The encapsulated fragment traverses through transport network and arrives to the receiver’s input at Phase3. Some of the fragments can be lost in transit and receiver’s design must be able to address this problem.

The receiver is comprised of two blocks “conditioner” and “assembler” connected internally via AXI-stream interface. The role of conditioner block is to strip incoming encapsulated fragment of the “outer header” and pass valid data of the original frame to the assembler block. The attributes from the discarded “outer header” shall be retrieved and presented to the assembler block together with original frame data in form of AXI-stream sideband signals. The fragments that contain only valid data of original ethernet frame with associated attributes are depicted on the Phase4 of the diagram.

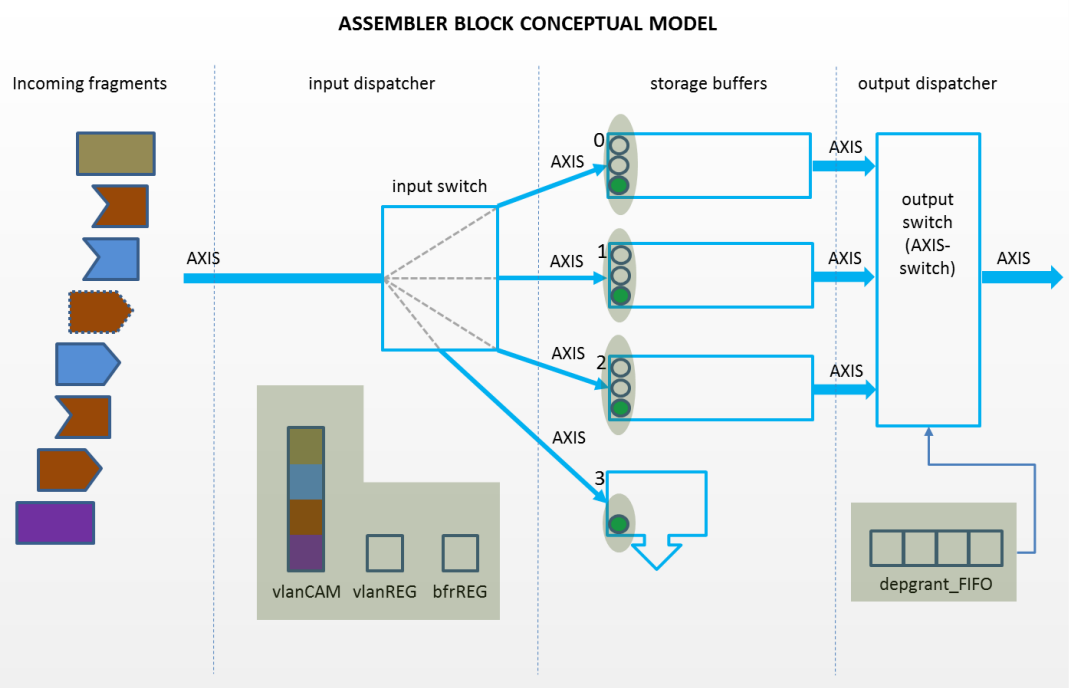
In this project the main attention will be given to the assembler block of the receiver. Assembler block accepts incoming fragments, temporarily store them and finally produce original ethernet frames seen at the Phase5 of the diagram.

Satisfying both project requirements for “multiple vlans” and “fragmentation” support is a challenge because fragments arriving to the receiver have to be assembled into original ethernet frames and the sequence of the arriving fragments is different to the sequence in that finally assembled frames are being produced.

This is the reason why a standard circular storage (or FIFO buffer) is considered unsuitable and a special “assembler” block performing “out-of-sequence” frame processing is being designed in this project.

Proposed conceptual model for “Assembler block”:

A conceptual model of the assembler block is proposed and depicted below.



This conceptual model is split horizontally into three logical parts (“input dispatcher”, set of “storage buffers” and “output dispatcher”). Each part plays its segregated functional role. Because of this separation the internal logic of each single part is relatively simple as it only have to control a limited set of registers and external signals.

For example out of all parameters associated with an arriving fragment the “Input dispatcher” is only interested in fragment’s ”vlan number” parameter. On the other hand the logic of “storage buffer” considers “F-flag” and “L-flag” values but not “vlan number” (see the operational principle of the assembler block).

On the higher level of abstraction an intercommunication protocol and set of external signals is introduced linking these logical entities together.

Operational principle of the assembler block:

-The routing decision for newly arrived fragment is made based on this fragment’s “vlan number”, set of registers within “input dispatcher” and storage buffer signals indicating presence of vacant space in these buffers.

-The “storage buffer” accepts the fragment and by the end of the reception it infers corresponding “external signal”. The storage buffer’s logic considers “F-flag” and “L-flag” parameters of the received fragment in order to set these “external signals”.

-When the frame is successfully assembled in the “storage buffer” the “input dispatcher” instructs the “output dispatcher” to schedule that frame for departure.

-The “output dispatcher” makes the stored in the buffer frame be dispatched asynchronously (independently) from the input logic.

This video demonstrates this conceptual model in action:

Please watch the video called "Etherblade.net Ver.2 (assembler block conceptual model demonstration)".

Transition from conceptual model diagram to hardware architecture:

In addition to the horizontal separation all the assembler block logic will be split vertically into separate "signalling-control" block and "dataplane". This is common hardware architecture of many communication systems. The pieces of logic with grey background depicted on the "assembler block conceptual model diagram" will be gathered into "signalling-control" block whereas remaining modules outlined in light-blue colour such as "input switch", "storage buffers", "output-switch" and AXIS buses will constitute the "dataplane".

For more information, please go to:

https://etherblade.net