Full Duplex Capture in Industrial Networks

Why SPAN Ports Should Not be Used in Security Solutions

By Thomas Tannhäuser and Alexander Pirogov

GARLAND TECHNOLOGY
See every bit, byte, and packet®

The convergence of IT and OT in the context of Industry 4.0 has led to a crowded market of security solutions targeting the shop floor on different levels. While the security of the legacy IT systems was part of the initial planning of those systems, the industry now faces the challenge to integrate security solutions in legacy OT systems.

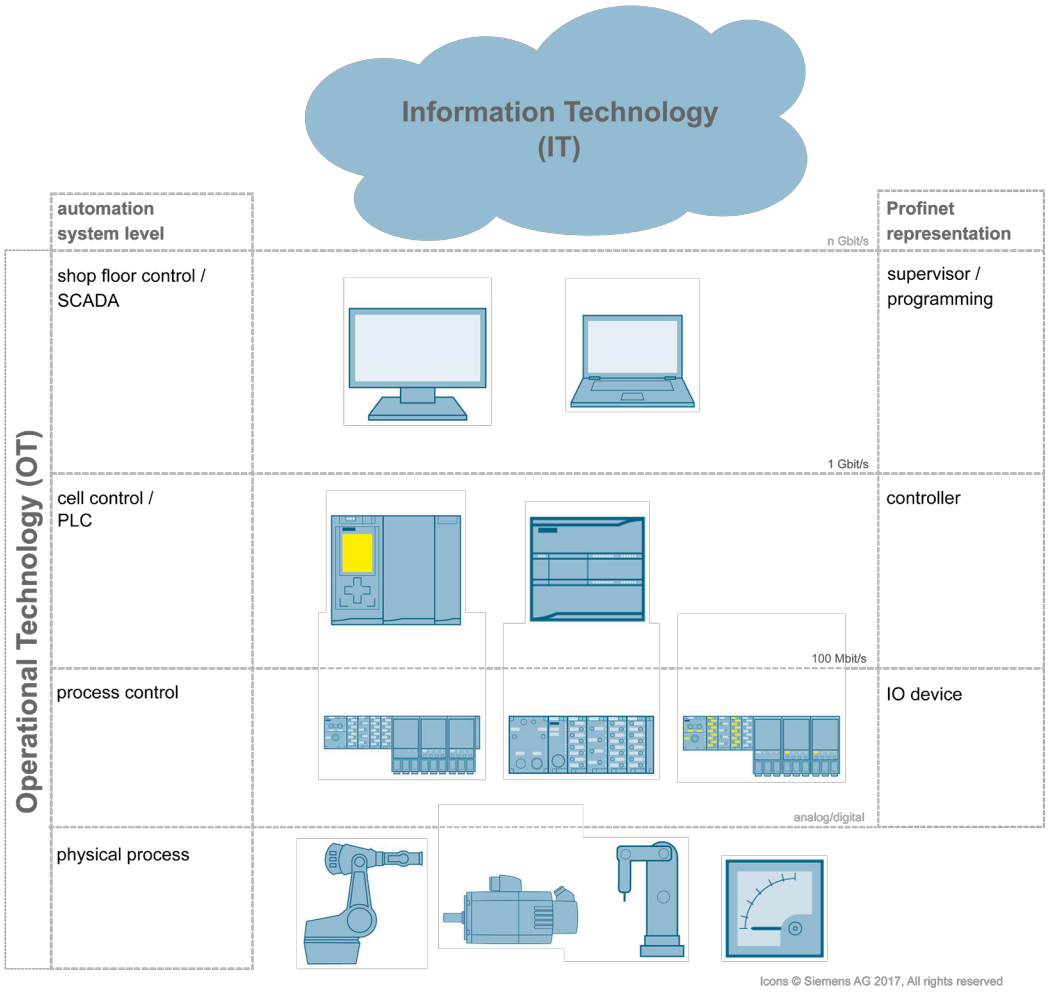

Figure 1: Layers of a typical automation system using Profinet

At least for the lower levels of the Industry 4.0 infrastructure (control and device level) security solution vendors tend to use the mirror/span ports offered by the network switches to integrate their solutions into the infrastructure.

This tech note will explore why those mirror ports shouldn't be used to build security solutions. Based on the hardware commonly used in Industrial Networks, we will specifically look at:

- How it all works
- Why a mirror port will drop frames even if a link is not saturated

Saturation Overload

I came across some research data Garland posted from Packet Pioneer that said a mirror port will drop up to 8% of the frames.

I took a deep breath because an 8% loss is a lot, especially if you are going to build security applications on top of the mirrored network traffic from systems using Profinet in different flavors (configuration, real-time and alarm IO).

I continued reading and calmed down, in their research, they used iperf to saturate a link that was mirrored at the same time.

Saturation means the system is in an overload condition, so drops are acceptable. And every network engineer will agree, that using more than 80% of the bandwidth of an Ethernet link will result in weird behavior at some point.

But I couldn't get past these questions, 'What happens, if only small frames are transmitted?' The average frame size in industrial networks is ~130 bytes, (calculated from a sampling of shop floor traces containing mostly Profinet) but the aforementioned test was done using 1500 byte frames (iperf usually uses TCP and that will send the data in chunks of MTU size). 'What if we use more than one link?' On the process level usually one controller talks to a number of IO-devices via one switch.

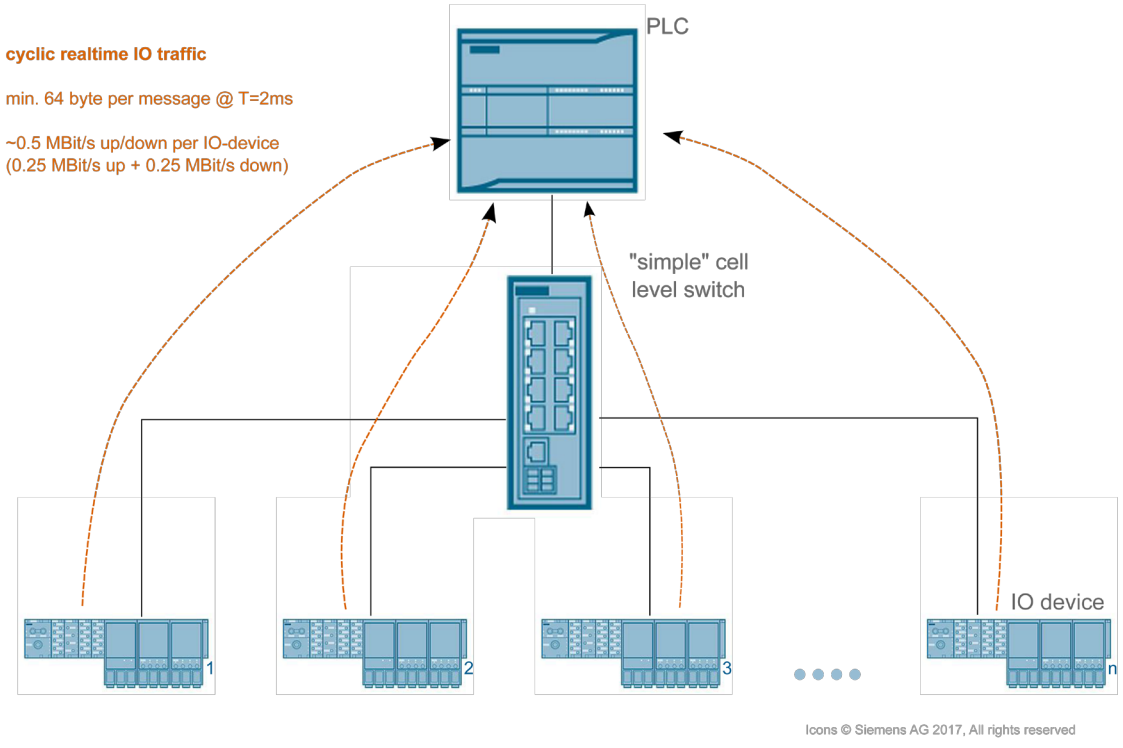

Figure 2: Process level - Profinet master and IO-Devices

In the context of industrial automation systems a test setup using a one-to-one connection running 1500 byte frames does not reflect a common scenario. And regarding automation systems, what if the traffic on those multiple links shows different bandwidth patterns?

Test Setup

Keeping those questions in mind I built a small test setup based on the open source network traffic generator and analyzer Ostinato. This tool offers a nice interface to create well defined streams of Ethernet frames for multiple network interfaces while intercepting traffic at the same time. This is not a hardware frame generator, but capable of handling multiple 100 Mbps streams with relative low-tech network interface cards. 100 Mbps is absolutely sufficient for running tests against network switches commonly used on the lower levels in a shop floor automation environment, as this is the usual network speed used there.

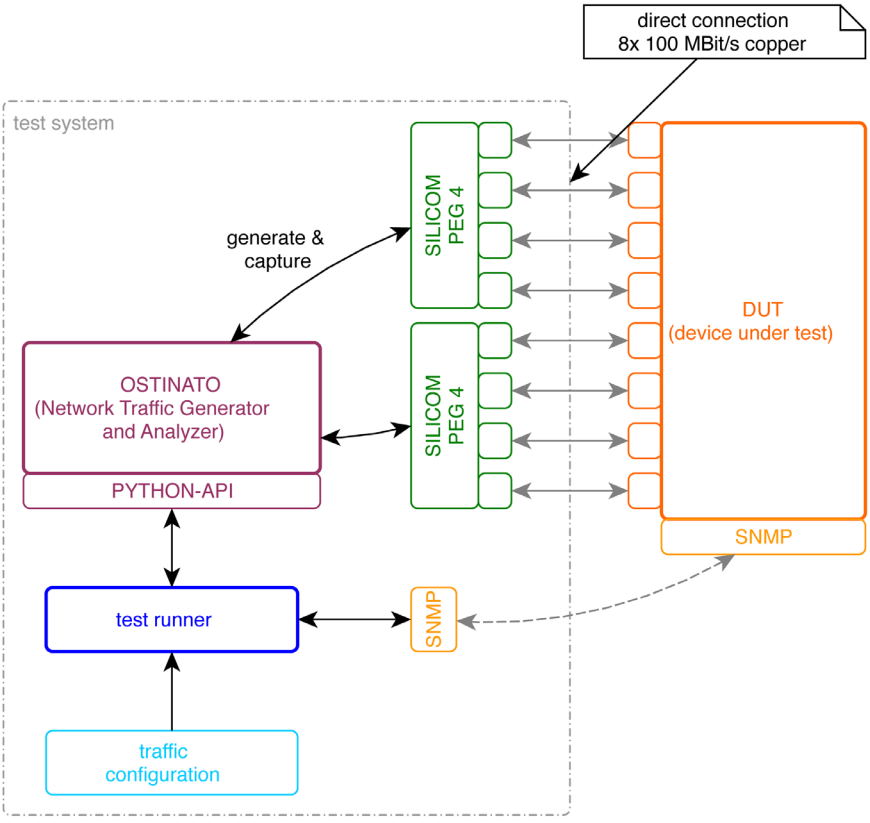

Figure 3: Test setup

Let's have a closer look at those network switches and how they work.

In-depth Intermezzo: Switches

Switches used at the lower end of a factory automation system are simple devices. They usually consist of a ready-to-use network switch chip connected to an inexpensive CPU. Add a little RAM and some persistent storage and voila, a network switch with (compared to switches used in the IT world) limited management capabilities is ready.

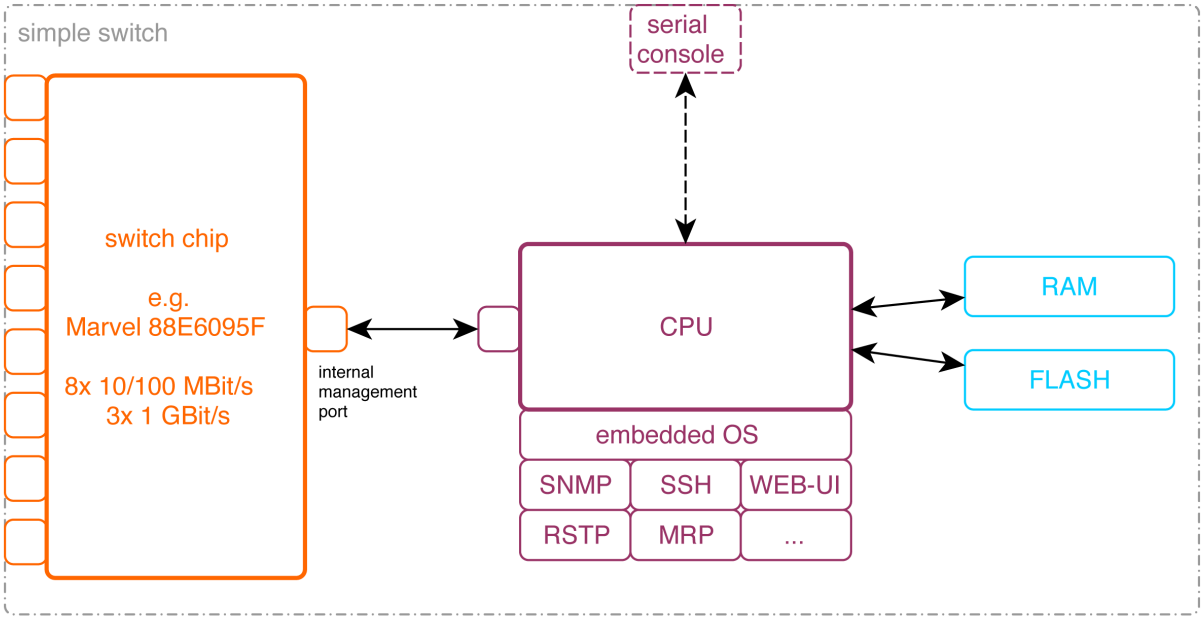

Figure 4: Principle switch architecture

The CPU runs a vendor specific embedded system (like VxWorks) used to implement:

- A User-Interface (usually via Web, serial port, SSH or SNMP)
- Handling of protocols related to topology management (i.e. LLDP, RSTP, MRP)
- The control of the used switch chip.

The last point is the important one. To be able to offer robust but also attractively priced products, usually a switch vendor uses the functionality of the provided switch chip as it is so he does not need to implement (and thus test and certify) the switch core functionality on its own.

So it is no surprise that you can find the same or similar chips in devices offered by different vendors or brands for the same purpose.

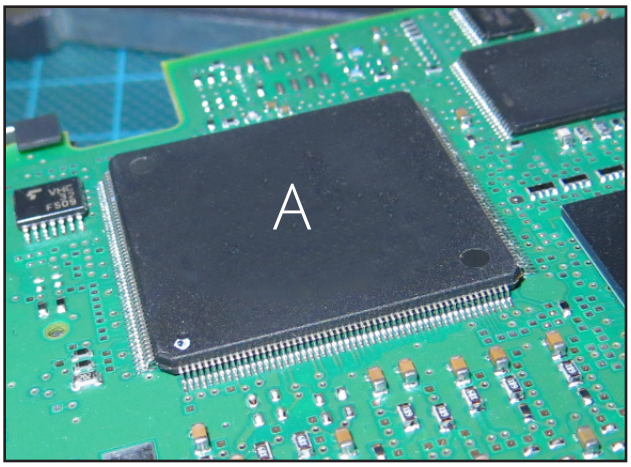

Figure 5: Vendor A - switch chip on the main board

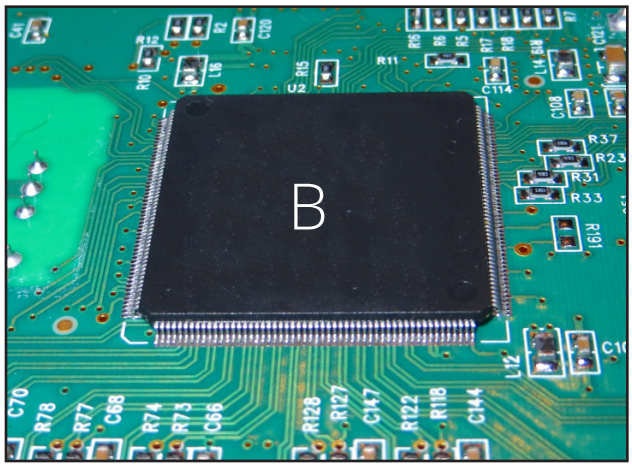

Figure 6: Vendor B - Similar switch chip

The 'sample switch chips' shown in the pictures above gives a good example. It offers, out of the box, eight 10/100 Mbps and two 10/100/1000 Mbps network ports. The 10/100 Mbps ports can be more or less directly connected to an Ethernet-based network as they already contain the required physical interface (PHY). The gigabit ports can be used to connect additional optical or electrical network interfaces using a common GMII interface. One network port is used to connect the switch chip to the management CPU. This allows a simple design combined with a low parts count.

Let's review some switch fundamentals before we explore their architecture. There are two main concepts on how to process network frames within a switch: *cut through* and *store-and-forward*.

A switch using *cut through* stores the first 6 bytes of the Ethernet frame, commonly known as the destination MAC address. It uses the address to check a lookup table on which port the frame should egress, and starts forwarding the data immediately. It only needs a few bytes of frame storage and forwards the frame with minimum latency.

If the switch runs in *store-and-forward-mode* the whole frame is stored within a buffer, then the lookup is done and the frame is transmitted on the matching egress port(s). This approach requires a lot of memory and adds a latency depending on the frame size (1500 byte frame @ 100 Mbts takes 240 µs).

While the *cut-through* version requires free ports to be able to transmit the frame directly a store-and-forward based switch could keep some frames in memory until the required egress ports are free. In the wild the cut-through-approach is usually combined with a store-and-forward-engine so the switch can adapt its mode of operation to the outer conditions/the network utilization.

Back to the 'sample switch' - This chip implements the *store-and-forward* switching methodology.

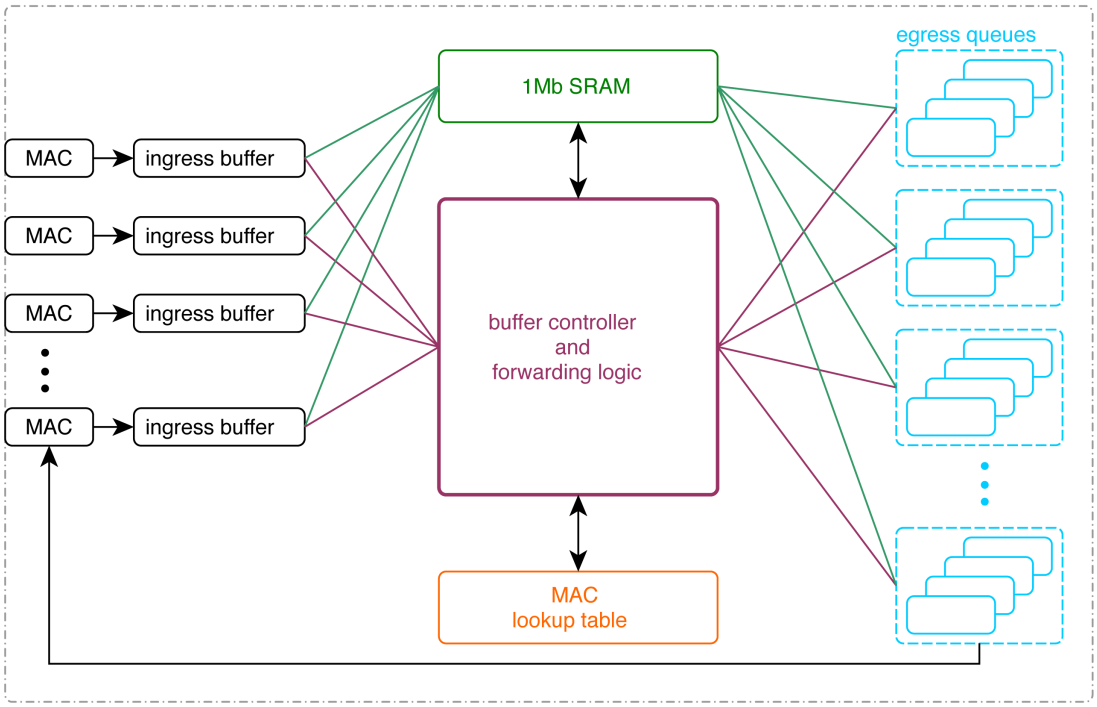

Figure 7: Simplified schematic of the 'sample switch"

It contains 1 MB (=128kByte) of fast SRAM used as a buffer to store frames received from the network - so it could keep ~2000 64 byte frames or 85 1500 byte frames in memory. This buffer is handled by a central controller. As soon as there is some free memory available it is assigned to one of the network ports. After receiving a frame, the portion of memory is returned to the controller that is responsible for looking up the egress port(s). After getting the required target information the memory is assigned to the egress buffer of the first port that should transmit the frame. This buffer consists of four queues each representing a different priority level.

Those egress buffers can be used to implement different kinds of quality of service mechanisms (e.g. the Priority Code Point of a VLAN tag is used to assign frames to the egress queues).

If there is no memory left (all memory assigned to ingress ports or egress queues) the switch starts to drop frames - usually on the receiving side.

Beside the process of transmitting and receiving a frame (big 1500 byte frame: 120us, small 64 byte frame: 5us) and the required lookups, the buffer handling and the memory access (for SRAM typically 5ns ..100ns) takes some time. This could be ignored for big frames. But for small frames the processing overhead compared to the time required to receive and transmit the frame adds up quickly.

This becomes even worse if a frame must be transmitted by more than one egress port. This is required for broadcast frames (unknown target or broadcast target address) and mirrored traffic. In that case the frame is not just moved to one egress queue, but many, and each egress port needs to access the SRAM to fetch the frame data for transmission. As only one entity at a time can access the SRAM, all consumers need to line up. In the worst case scenario, a frame needs to stay in memory for the number of ports it should be transmitted to, multiplied by the time it takes to transmit that frame, plus the time of the processing overhead.

I think now it's clear why I titled the scenario used by Packet Pioneer "not common." A one-to-one connection using 1500 byte frames creates a way less processing overhead in a switch than multiple connections running different bandwidth patterns. But with multiple connections we will see a lot of buffer back and forth, memory access and maybe congested egress queues.

Full-duplex Ethernet and Its Use in Industrial Control Protocols

Using independent ingress and egress queues connected to the receive and transmit paths of the physical layer of an network interface allows parallel transmission and reception of Ethernet frames - the so called full duplex (FDX) mode.

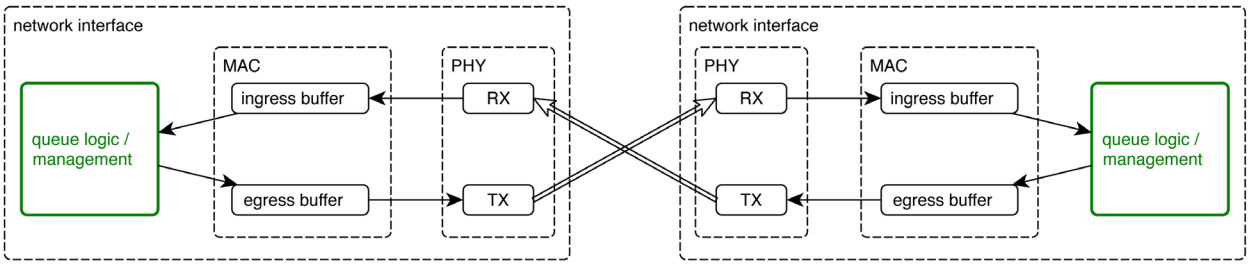

Figure 8: Full duplex connection

This is the standard mode of operation for Ethernet based networks. The ability to send and receive data at the same time using one logical channel is used to increase the total available bandwidth, (you can run 100 Mbps up and downstream in parallel) and can be used to reduce the round trip time for protocols requiring an explicit knowledge of transmitted data.

A good example for a protocol that explicitly uses the full duplex property of Ethernet as a characteristic of its implementation is EtherCAT. To be able to meet real time requirements with a guaranteed cycle time <= 100us in standard Ethernet based local networks the associated standard (IEC 61158-3-12) enforces the use of full duplex capable links (Sec. 4.1 - Operating principle). Based on this, EtherCAT is able to build some kind of a tight communication loop that results in very short cycle times:

1. The master node sends a frame to the first slave.
2. The slave processes (read data addresses for him, add data to the frame that was requested from him) the frame byte by byte, while receiving and forwarding it directly to the next slave.
3. The last slave in the chain just sends the frame back to the master, while also processing it on the fly.

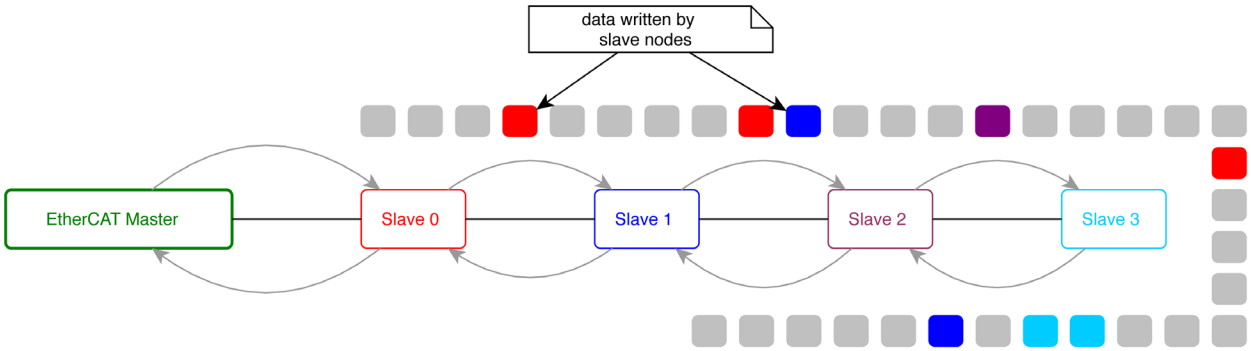

Figure 9: In flight modification of an Ethernet frame in full duplex mode

All slaves in an EtherCAT system must be able to process up and downstream frames at the same time, or in other words, in full duplex. This in-flight processing requires some special hardware and makes it impossible to implement that kind of node on common embedded platforms using standard network chips (while the master can be built using a standard network interface).

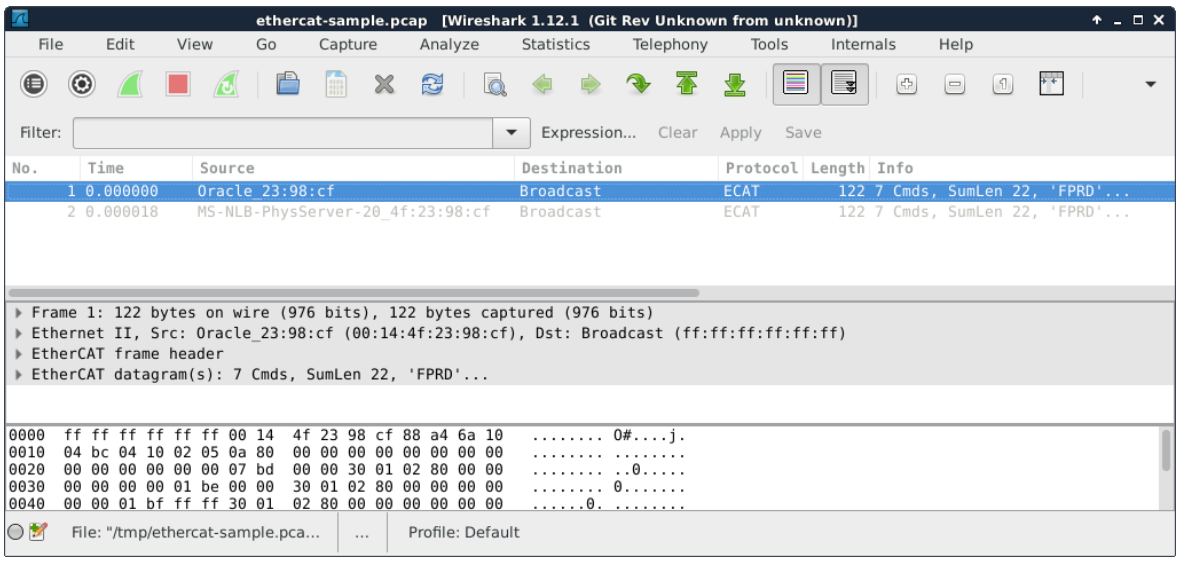

Figure 10: EtherCAT frame returns to the master after ~18 us (demo setup)

Another big player in the area of the Ethernet-based industrial control protocols is Profinet. It offers 3 different performance classes for implementing real-time control systems with increasing timing accuracy. Class two and three offer an accuracy down to one micro-second but require some extensions to the Ethernet-stack. Class one can be run over stock Ethernet-based networks - still guaranteeing cycle times down to 1 ms with a really small jitter.

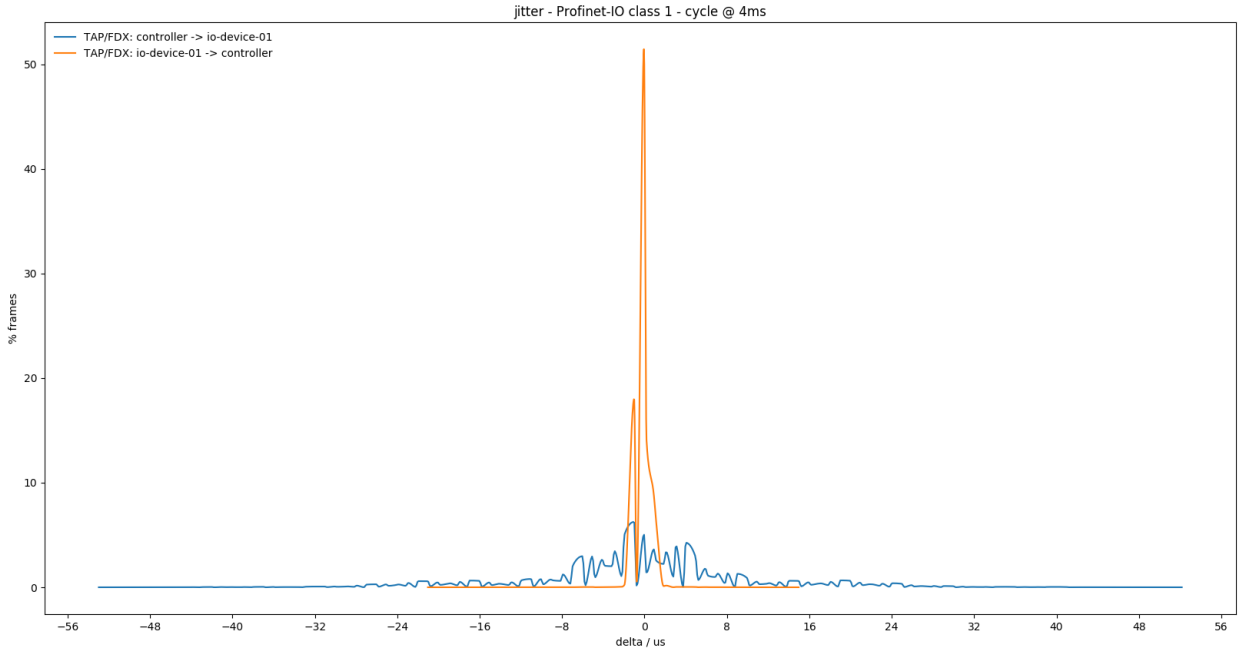

Figure 11: Jitter of cyclic Profinet-IO RT class 1 communication with a configured cycle time of 4 ms

To achieve this the related standard IEC 61158-5-10 explicitly requests the general usage of full-duplex networks (sec. 8.6.2 - Network topology: “shall support a bitrate of 100 MBit/s in full duplex mode”). As Profinet systems could - in terms of the number of devices and connections - become really big, this requirement aims to the efficient use of the available infrastructure. The maximum recommended network utilization in a Profinet system is 20 percent. Applying this as a hard limit the use of full duplex doubles the available bandwidth - or the number of devices that could participate in such a system (assuming equal distribution of transmitted data) - by keeping the jitter caused by temporary congestion low.

Test & Results

Here is a small set of different traffic patterns using different bandwidths on a varying number of ports.

Name	Pattern
single-mirrored	n-0
single-non-mirrored	0-n
dual-symmetric	n-n
dual-asymmetric	n-10
quad-asymmetric-left	n-10-10-10
quad-asymmetric-right	10-n-n-n
quad-asymmetric-unbalanced	10-n-(n+10)-(n+20)
quad-1:3	n-n-n-n

Those patterns could be seen as some kind of brute force attempt to find some special conditions that puts the switch in a I will-drop-early mode. But I think the quad-asymmetric-right and quad-1:3 could be seen as realistic patterns for automation systems. The first interface is the mirrored interface and it always sends frames to the second interface (if configured). All other interfaces send their traffic to the first interface. The test tool automatically increased the used bandwidth for each tested frame size (64, 130, 512, 1500 bytes) and monitored the number of transmitted and received frames on each participating port using the API offered by the Ostinato traffic generator and (as some kind of reference) the relevant entries in the SNMP MIB.

For the results...first, the good. I was not able to spot any traffic pattern that caused some weird behavior compared to the other scenarios. And I performed the same tests on a handful of different devices from different vendors. So the following results can be seen as some kind of general results - no special treatment required for any ‘special’ switch or vendor.

Not a single frame was dropped on the mirror when only one port was used to inject traffic (single-mirrored, single-non-mirrored). The switch was able to forward and mirror the whole 100 Mbps for all tested frame sizes. This proves the switch as capable to handle 100 Mbps.

But if you are going to use more than one port the performance of the mirror ports drop a lot. The switch is still able to forward all frames, but above 80 Mbps bandwidth consumed by all ports, not all frames are still forwarded to the mirror port.

Scenario	'Sample Switch'
dual-symmetric (n-n)	64: 80 Mbit/s 130: 90 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s
dual-asymmetric (n-10)	64: 80 Mbit/s 130: 90 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s
quad-asymmetric-unbalanced (10-n-(n+10)-(n+20))	64: 85 Mbit/s 130: 91 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s
quad-asymmetric-right (10-n-n-n)	64: 85 Mbit/s 130: 91 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s
quad-asymmetric-left (n-10-10-10)	64: 80 Mbit/s 130: 90 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s
quad-1:3 (n-n-n-n)	64: 80 Mbit/s 130: 88 Mbit/s 512: 100 Mbit/s 1500: 100 Mbit/s

It seems the mirroring adds some extra steps in the frame processing that are tagged with a lower priority. So if the switch runs out of memory it drops frames directed to the mirror port. Usually that kind of frame drop caused by an overload situation could be monitored by querying the SNMP attributes (if-MIB, discard/error/queue sizes) but none of the tested switches using the 'sample switch chip' showed any numbers that could be used to indicate a drop on the mirror port. The only way to detect those silent drops on the switch was to add up the ingress counter and compare them against the egress counter of the mirror port. But this only works if the traffic is stopped while fetching all counters, and not in a running automation system.

On the other hand: 80 Mbps doesn't sound so bad, right? But keep in mind, this is the sum of the bandwidth consumed by all ports on the switch. So if you connect 5 links producing 15 Mbps each you are already close to the limit if you are going to mirror a port that receives all that traffic (i.e. the port the PLC is connected to). Only 5 Mbps more (i.e. a user UI session, firmware update) and you will loose frames. And as we usually talk about full duplex mirrors, this additional traffic can be added on the ingress interface of the mirrored port or any other port that can send traffic to egress queue of the mirrored port.

So if you need to hide a single frame that carries an attack (like in the good old days: restart the switch by sending a really big SNMP request), you could try to overload the mirror temporally - and with a probability of 1:4 your frame will never egress the mirror port. You could also try to use some VLAN priority trickery to increase the probability the frame is silently dropped.

Conclusion

It turned out that even with non-saturated links, the frame drop on the mirror is a real issue. It's not some switch vendor specific issue, but rather a result of how the underlying switch chip works. Those chips were never made to be used as a network TAP, they were made for switching network frames to a single port.

In case of the 'sample switch chip' this is also reflected by the mirror limitations described in the chips manual as following frames are not forwarded to the mirror:

- All frames that couldn't be stored because of a shortage in memory
- Frames showing a broken CRC (if the fix-CRC bit is set the CRC is fixed and the
- Frame forwarded to the mirror - but in that case you will miss the error on the link)
- Pause frames (so an attacker could simulate a congestion situation - but the application connected to the mirror won't be able to detect the pause-attack)
- Frames smaller than 64 bytes or bigger than the configured MTU and
- Frames dropped because of an ingress rate limit.

There are also a handful of frame-types that will be forwarded only to the mirror port:

- Frames discarded for 802.1X Source MAC address or 802.1Q security
- Frames containing a/no VLAN tag if DiscardTagged/DiscardUntagged is enabled
- Frames that would be discarded because of the port state.

Some of them could be abused to inject traffic that would confuse a state protocol tracker (i.e. the state of a TCP connection) running on a connected (security) appliance.

Beside these, at least one switch manual added some restrictions and issues on top:

- Mirrored and mirror port must be part of the same VLAN
- Mirror may sometimes strip the VLAN tag
- Management frames like STP, GVRP may not be mirrored and
- Frames originating from the switch's CPU (usual management traffic from the WebUI, telnet or ssh) may not be mirrored.

If you are using the isochronous Profinet extension IRT (Real time traffic class 3) for the communication between your controllers and IO devices you are completely out of luck. Enabling mirroring on a IRT capable switch completely disables the IRT functionality. So be careful - enabling mirroring on such a switch could bring down your production process with one mouse click.

So you can use the mirror ports for building an Industry 4.0 security application if:

- Bandwidth always stays way below 80% - in sum for all frames to be mirrored (TX of the mirrored port and sum(TX) all ports feeding the mirrored port) **and**
- Your application can deal with injected traffic (so state-machines won't get confused) **and**
- You monitor the switch used as the mirror source using an additional interface like SNMP or RMON **and**
- The network is (physically) secured so an intruder can't overload network links by injecting fake traffic to hide its own (malicious) traffic.

The last point is important and the limiting factor: you won't be able to guarantee full physical protection by investing a reasonable amount of money.

You must use network TAPs if:

- The (peak/burst) network TX-utilization of the mirrored port is close to 80% or could reach this limit **or**
- The system uses full duplex and the RX/TX-bandwidth in sum is >= 80 Mbps **or**
- You use small frames and traffic showing sporadic bursts (e.g. substation systems using IEC 61850 SMV combined with GOOSE traffic) **or**
- Your system uses Profinet IRT, as switches usually won't be able to use IRT and mirroring at the same time (provided they support PN IRT at all) **or**
- You need to get defective frames (most switches running in cut through will forward them too, but this really depends on the switch model, configuration and type of the error) **or**
- You use VLANs, as some switches are not able to mirror more than one specific VLAN or VLANs at all.

To be able to capture all traffic in a full-duplex 100 Mbps network choose either:

- A non aggregating TAP forwarding RX/TX frames on separate ports (needs capture on two ports) or
- An aggregating TAP that is able to combine the RX/TX-stream of a 100Mbps link on a 1Gb port.

In most cases, it is clear, you must use a network TAP, especially for building Industry 4.0 security solutions.

Don't build security systems based on assumptions about static traffic patterns.

1. *There are a handful of exceptions. Some protocol specific functionality (i.e. Profinet-IRT) requires dedicated switch chips, and I have seen systems using ASICS for implementing the core network functionality*
2. *Per port one additional small transformer is required*
3. *I will do more tests as soon as I get new hardware and will update my statement if its starts to become 'wrong.'*

Written by:
Thomas Tannhäuser - thomas.tannhaeuser@tomeido.de
Alexander Pirogov - alexander.pirogov@tomeido.de