High-availability Seamless Redundancy (HSR) - introduction, current Linux driver status and further development

Lukasz Majewski

18.09.2024 / Linux Plumbers Conference

Outline

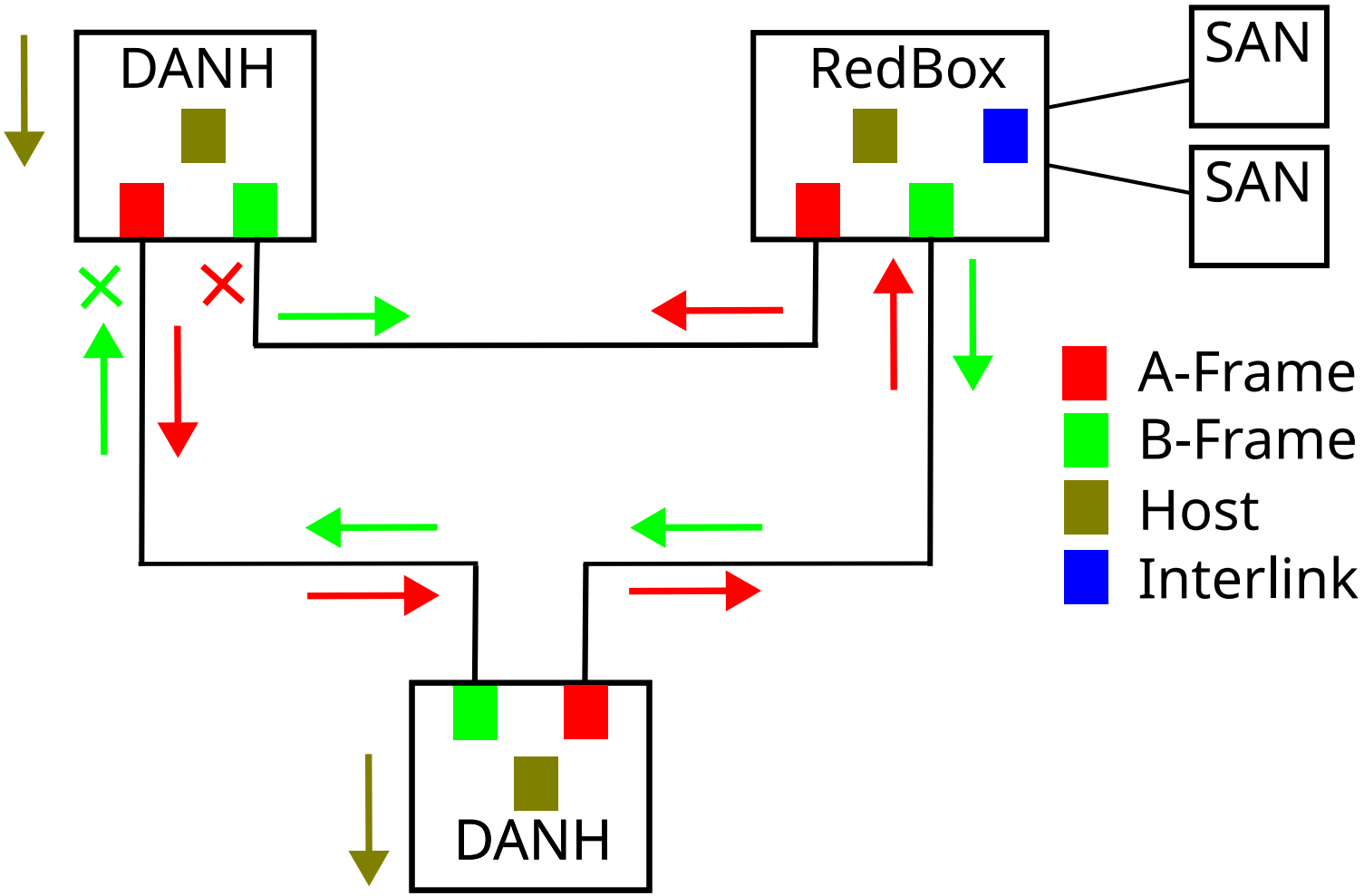
A few words about me

- Embedded software engineer at DENX
- Contact: `lukma@denx.de`
- Involved in:
  - Zephyr (networking - T1S, DSA)
  - U-Boot (USB/DFU)
  - Linux (networking)
  - glibc (y2038)

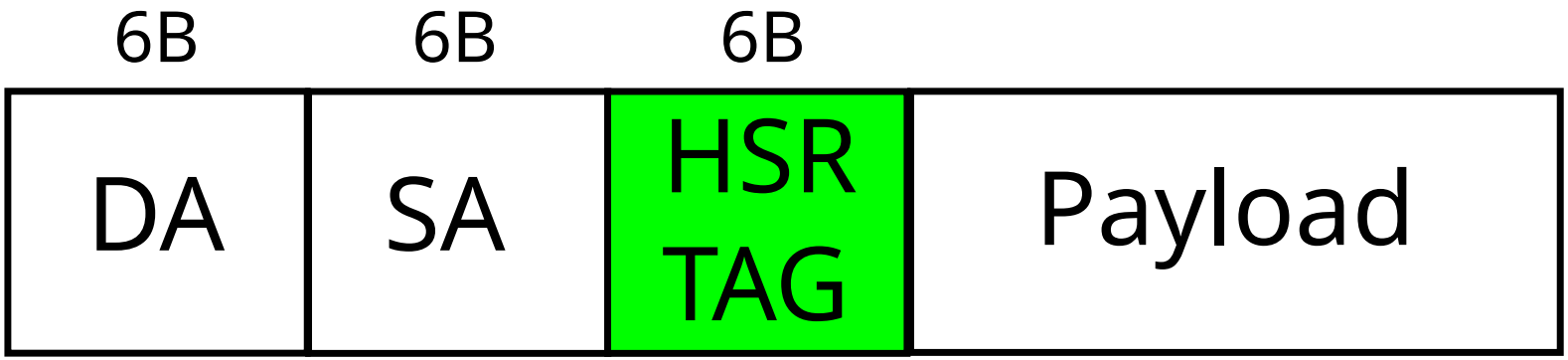
Glossary

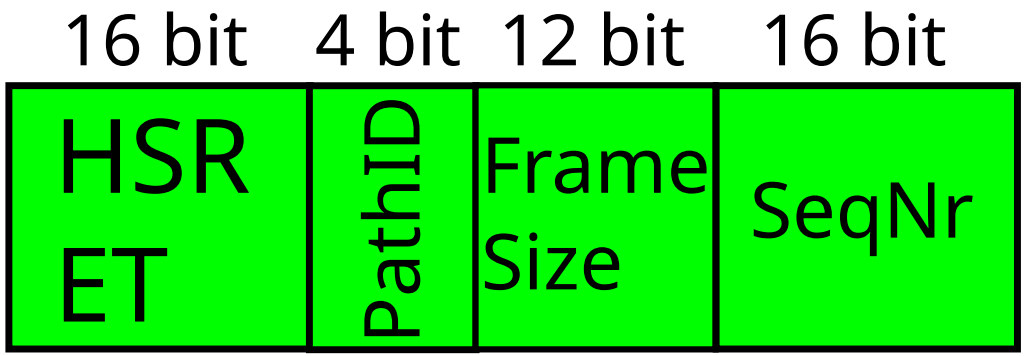
- **HSR** = High-availability Seamless Redundancy
- **DANH** = Doubly Attached Node - HSR
- **SAN** = Singly Attached Node
- **RedBox** = Redundancy Box
- **PRP** = Parallel Redundancy Protocol

Operation idea I



Operation idea II





Link Service Data Unit

Facts

- Standardized by the IEC 62439-3:2016 Clause 5 [2]
- Very similar to PRP (star vs. ring topology)
- **Two ETH ports** are used for a **single HSR** connection
- Frames are duplicated
- Frames are not compatible with standard ethernet (motivation for RedBox)
- No single point of failure
- Ring topology
- ZERO switch-over time with no packet loss
- **Usage:** industrial / transportation / aviation networks
- Possible HW offloading (e.g. KSZ9477, TI's PRU-ICSS)
- Monitoring HSR ring state by sending supervisory frames at fixed interval (2s)

Driver features I

- HSR `v0/v1` and PRP support in SW
- HSR RedBox support - `iproute2 v6.10+` required [3]
- Close cooperation with `DSA` subsystem
- RedBox filters out traffic from `HSR` ring to `SAN(s)` and opposite (performance improvement)
- Offloading:
  - *KSZ9477*

    `NETIF_F_HW_HSR_DUP | NETIF_F_HW_HSR_FWD`
  - *XRS700x*

    `NETIF_F_HW_HSR_DUP | NETIF_F_HW_HSR_FWD | NETIF_F_HW_HSR_TAG_INS | NETIF_F_HW_HSR_TAG_RM`
- `Node table` internals exported

Ongoing development I

- Use `atomic_inc_*` instead of plain ++
- Optimize the usage of `spinlock()`
- HSR offload support for `ICSSG` (TI's AM64)

HW testing I

```
# Ports 4/5 were used for SW managed HSR (hsr1) as first hsr0 for ports 1/2
# (with HW offloading for ksz9477) was created. Port 3 has been used as
# interlink port
#
# Only SW based HSR devices (hsr1).
#
# --------------    hsr1    ------------------        ------------
# DAN-H  Port5 | <------> | Port5           |        |
#        Port4 | <------> | Port4    Port3 | <---> | PC
#              |          | (RedBox)        |        | (USB-ETH)
# EVB-KSZ9477  |          | EVB-KSZ9477     |        |
#        Port1 | <------> | Port1           |        |
#        Port2 | <------> | Port2           |        |
# --------------    hsr0    ------------------        ------------
#
```

HW testing II

```
# Configuration - RedBox (EVB-KSZ9477):
if link set lan1 down;ip link set lan2 down
ip link add name hsr0 type hsr slave1 lan1 slave2 lan2 supervision 45 version 1
ip link add name hsr1 type hsr slave1 lan4 slave2 lan5 interlink lan3 supervision 45 version 1
ip link set lan4 up;ip link set lan5 up
ip link set lan3 up
ip addr add 192.168.0.11/24 dev hsr1
ip link set hsr1 up

# Configuration - DAN-H (EVB-KSZ9477):

ip link set lan1 down;ip link set lan2 down
ip link add name hsr0 type hsr slave1 lan1 slave2 lan2 supervision 45 version 1
ip link add name hsr1 type hsr slave1 lan4 slave2 lan5 supervision 45 version 1
ip link set lan4 up;ip link set lan5 up
ip addr add 192.168.0.12/24 dev hsr1
ip link set hsr1 up
```

Offloading vs. SW [KSZ9477-EVB]

```
# Performance SW used:
nuttcp -S --nofork
nuttcp -vv -T 60 -r 192.168.0.2
nuttcp -vv -T 60 -t 192.168.0.2
```

	RX [Mbps]	TX [Mbps]
HW offloading	100	98
Only SW	63	63

QEMU testing I

```
#
# IPv4 addresses (100.64.X.Y/24), and [X.Y] is presented on below diagram:
#
# |NS1                        |               |NS4                    |
# |       [0.1]               |               |                       |
# |    /-- hsr1 --\           |               |    [0.41]             |
# | ns1eth1     ns1eth2       |               |    ns4eth1 (SAN)      |
# |---------------------------|               |-----------------------|
#      |             |                                 |
#      |             |                                 |
#      |             |                                 |
# |---------------------------|   |-------------------------------------|
# | ns2eth1     ns2eth2       |   |                  ns3eth2            |
# |    \-- hsr2 --/           |   |                 /                   |
# |      [0.2] \              |   |                /                    |  |-----------|
# |             ns2eth3       |---| ns3eth1 -- ns3br1 -- ns3eth3--|--| ns5eth1   |
# |             (interlink)   |   | [0.3]      [0.11]             |  | [0.51]    |
# |NS2 (RedBOX)               |   |NS3 (BR)                       |  | NS5 (SAN) |
#
```

QEMU testing II

- Handy when no real HW available (`netns`, `tc`)
- Easy examination with Wireshark/tshark
- Setting up environment (`QEMU` + `Buildroot`) to test HSR operation (configuration files [1])
  1. Prepare buildroot:

```
git clone git://git.buildroot.net/buildroot buildroot-hsr-redbox-veth
cd buildroot-hsr-redbox-veth
# use qemu_x86_64_defconfig with extra tools - like bash - as .config
time make
```

  2. Compile kernel:

```
# Currently used: net-next/main
# SHA1: bfba7bc8b7c2
# Use proper config (setup for 6.11.0-rc6) -> support for veth, HSR, netns:
# config-x86_64-qemu-buildroot .config
# Build kernel (toolchain from buildroot):
CC="x86_64-buildroot-linux-gnu-" ARCH=x86_64 make -j8 bzImage && \
rsync -L ./arch/x86_64/boot/bzImage \
         ./buildroot-hsr-redbox-veth/output/images/bzImage
```

QEMU testing III

3. Run buildroot and test HSR:

```
cd ./buildroot-hsr-redbox-veth/output/images
./start-qemu.sh --serial-only
mkdir ./hsr-test/
# Mount the directory with HSR test scripts:
sshfs lukma@10.0.2.2:linux-net-next/tools/testing/selftests/net ./hsr-test/
cd ./hsr-test/hsr/
./hsr_redbox.sh
./hsr_ping.sh -4
# (exit from the QEMU with Ctrl+A X)
```

Future enhancements I

- Debugging:
  - The `Node Table` examination now is only possible via `debugfs` - deprecated approach
  - Required easy access to `Proxy Node Table` internals
- Refactor `hsr_forward_do()` function (`net/hsr/hsr_forward.c`)
- Set `PatchID` in sysfs
- Split out HSR and PRP specific code
- PRP RedBox
- Provide `HSR` support for:
  - PTP
  - VLAN

Future enhancements II

- Modeling and testing HSR operation with QuadBox
- ```
  linux-net-next (main)$ git grep -n -A3 "HSR" ./MAINTAINERS
  MAINTAINERS:10341:HSR NETWORK PROTOCOL
  MAINTAINERS-10342-L:    netdev@vger.kernel.org
  MAINTAINERS-10343-S:    Orphan
  MAINTAINERS-10344-F:    net/hsr/
  ```
  ```

Summary

- HSR driver is actively developed
- HW offloading is supported
- Interest in open source `HSR/PRP` implementation increases
- QEMU based test setup [1] can be used *easily* for validation

Q & A

Thank you !

Links

[1] git@github.com:lmajewski/HSR-QEMU-setup.git

[2] https://webstore.iec.ch/publication/24447

[3] https://github.com/iproute2/iproute2/commit/c72323d2ef60c4d8a5738a50bbe3a4eba1cc45b0