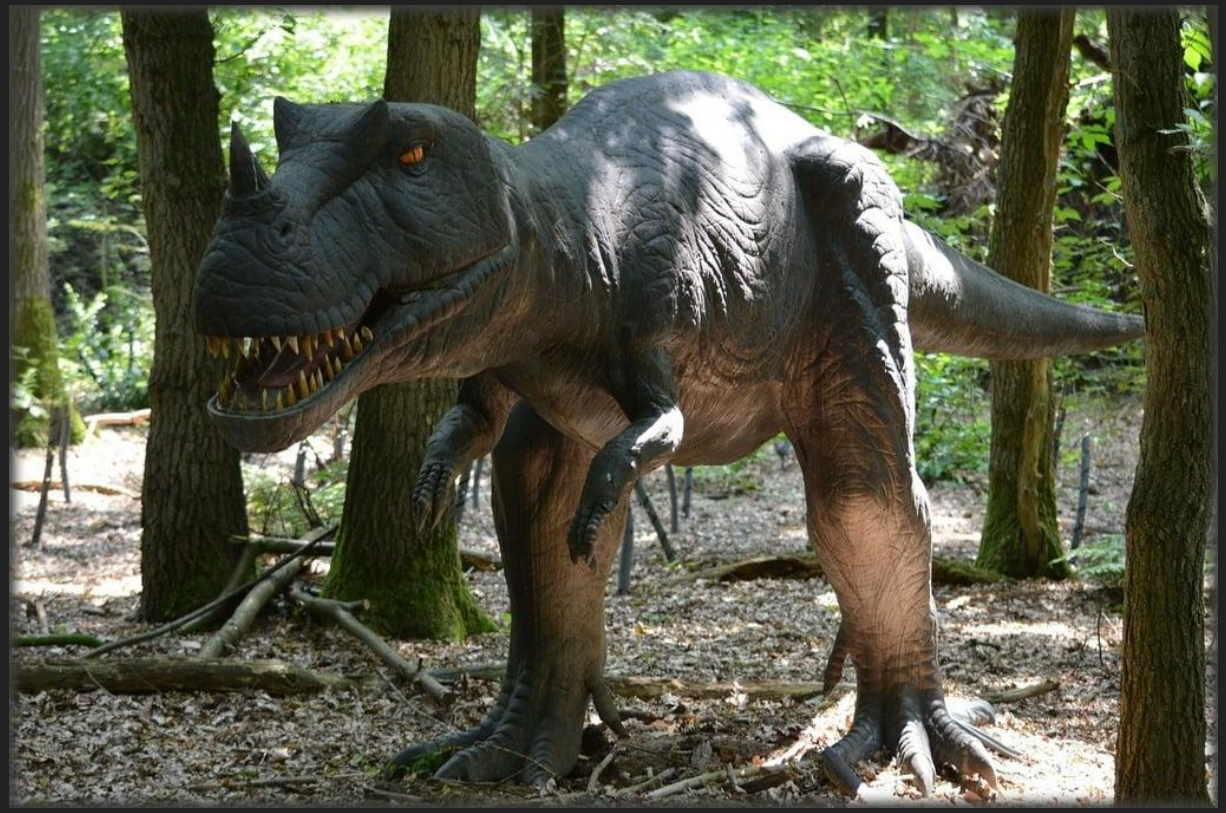
Linux: Hardware Switch Support

Pavel Šimerda



Prehistory

< 2020

2008: DSA tagging protocol & switch port devices

net: Distributed Switch Architecture protocol support

Distributed Switch Architecture is a protocol for managing hardware switch chips. It consists of a set of MII management registers and commands to configure the switch, and an ethernet header format to signal which of the ports of the switch a packet was received from or is intended to be sent to.

The switches that this driver supports are typically embedded in access points and routers, and a typical setup with a DSA switch looks something like this:

2013: swconfig & VLAN tagging

VLANs on "switch0"

VLAN ID	CPU (eth0)	CPU (eth1)	LAN 1	LAN 2	LAN 3	LAN 4	WAN	
Port status:	1000baseT full-duplex	1000baseT full-duplex	1000baseT full-duplex	no link	no link	1000baseT full-duplex	1000baseT full-duplex	
1	tagged	off	untagged	untagged	untagged	untagged	off	Delete
2	off	tagged	off	off	off	off	untagged	Delete

Add VLAN

```
config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '1'

config switch_vlan
        option device 'switch0'
        option vlan '1'
        option ports '0 1 2 3 5t'
        option vid '1'

config switch_vlan
        option device 'switch0'
        option vlan '2'
        option ports '4 6t'
        option vid '2'
```

2014: switchdev framework

author	Jiri Pirko <jiri@resnulli.us>	2014-11-28 14:34:17 +0100
committer	David S. Miller <davem@davemloft.net>	2014-12-02 20:01:20 -0800
commit	007f790c8276271de26416f90d55561bcc96588a (patch)	
tree	03a55b7897402e9daa8af64ea2c81d5236f77367	
parent	02637fce3e0103ba086b9c33b6d529e69460e4b6 (diff)	
download	linux-007f790c8276271de26416f90d55561bcc96588.tar.gz	

net: introduce generic switch devices support

```
The goal of this is to provide a possibility to support various switch
chips. Drivers should implement relevant ndos to do so. Now there is
only one ndo defined:
- for getting physical switch id is in place.

Note that user can use random port netdevice to access the switch.
```

2015: Bridge offloading via DSA

author	Florian Fainelli <f.fainelli@gmail.com>	2015-02-24 13:15:33 -0800
committer	David S. Miller <davem@davemloft.net>	2015-02-25 17:03:38 -0500
commit	b73adef67765b72f2a0d01ef15aff9d784dc85da (patch)	
tree	829c9c90cffe94d00a1a7ee568c9464fdcf0efae	
parent	d87d6f44d7c1254fd9560a5191659cb00882db56 (diff)	
download	linux-b73adef67765b72f2a0d01ef15aff9d784dc85da.tar.gz	

net: dsa: integrate with SWITCHDEV for HW bridging

```
In order to support bridging offloads in DSA switch drivers, select
NET_SWITCHDEV to get access to the port_stp_update and parent_get_id
NDOs that we are required to implement.

To facilitate the integratation at the DSA driver level, we implement 3
types of operations:

- port_join_bridge
- port_leave_bridge
- port_stp_update

DSA will resolve which switch ports that are currently bridge port
members as some Switch hardware/drivers need to know about that to limit
the register programming to just the relevant registers (especially for
slow MDIO buses).
```

2017: DSA @ Netdev Conf

Distributed Switch Architecture, A.K.A. DSA

1st Andrew Lunn, 2nd Vivien Didelot, 3th Florian Fainelli

[1]andrew@lunn.ch, [2]vivien.didelot@savoirfairelinux.com, [3]f.fainelli@gmail.com

Abstract

The Distributed Switch Architecture was first introduced to Linux nearly 10 years ago. After being mostly quiet for 6 years, it recently became actively worked on again by a group of tenacious contributors.

In this paper, we will cover its design goals and paradigms and why they make it a good fit for supporting small home/office routers and switches. We will also cover the work that was done over the past 4 years, the relationship with switchdev and the networking stack, and finally give a heads-up on the up-coming developments to be expected.

Keywords

DSA, Distributed Switch Architecture, Linux kernel network stack, SOHO switches, switchdev.

Introduction

Distributed Switch Architecture is a Marvell SOHO switch term. However, as is often the case with the Linux Kernel, the code to support it has been generalised, and now supports

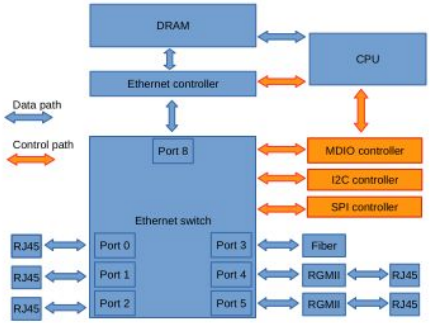
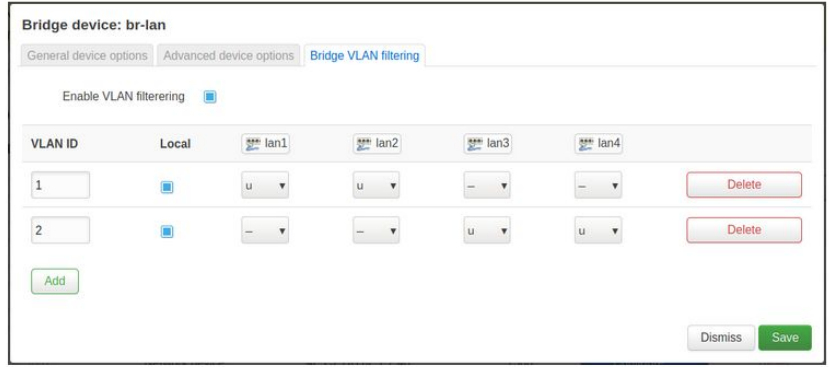


Figure 1: The Basic DSA setup

2021: OpenWRT & DSA

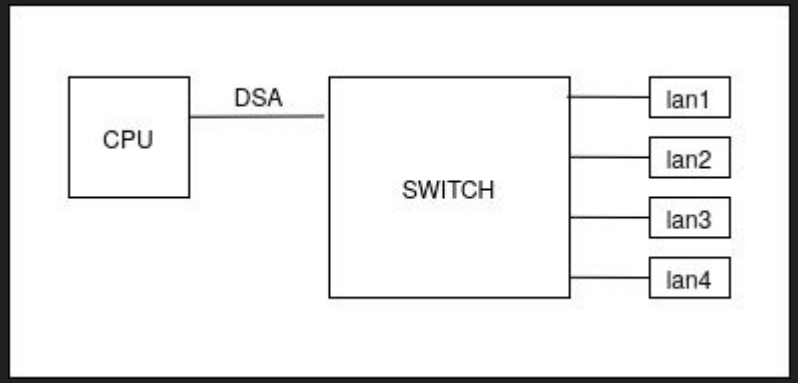



```
config device
        option name 'br-lan'
        option type 'bridge'
        list ports 'lan1'
        list ports 'lan2'
        list ports 'lan3'
        list ports 'lan4'

config bridge-vlan
        option device 'br-lan'
        option vlan '1'
        list ports 'lan1'
        list ports 'lan2'

config bridge-vlan
        option device 'br-lan'
        option vlan '2'
        list ports 'lan3'
        list ports 'lan4'
```

Why is DSA so important?



- Forward
- From CPU
- To CPU

Special incoming/outgoing frames

- Spanning tree – STP, RSTP, MSTP
- Discovery – LLDP
- Bonding – LACP
- …

2022: MSTI state setting

- Incompatible per-VLAN state since 2020
- Per-MSTI state setting
- CST state support
- Offloading via switchdev

Merge branch 'net-bridge-multiple-spanning-trees'

Tobias Waldekranz says:

```
====================
net: bridge: Multiple Spanning Trees

The bridge has had per-VLAN STP support for a while now, since:

https://lore.kernel.org/netdev/20200124114022.10883-1-nikolay@cumulusnetworks.com/

The current implementation has some problems:

- The mapping from VLAN to STP state is fixed as 1:1, i.e. each VLAN
  is managed independently. This is awkward from an MSTP (802.1Q-2018,
  Clause 13.5) point of view, where the model is that multiple VLANs
  are grouped into MST instances.

  Because of the way that the standard is written, presumably, this is
  also reflected in hardware implementations. It is not uncommon for a
  switch to support the full 4k range of VIDs, but that the pool of
  MST instances is much smaller. Some examples:

  Marvell LinkStreet (mv88e6xxx): 4k VLANs, but only 64 MSTIs
  Marvell Prestera: 4k VLANs, but only 128 MSTIs
  Microchip SparX-5i: 4k VLANs, but only 128 MSTIs

- By default, the feature is enabled, and there is no way to disable
  it. This makes it hard to add offloading in a backwards compatible
  way, since any underlying switchdevs have no way to refuse the
  function if the hardware does not support it

- The port-global STP state has precedence over per-VLAN states. In
  MSTP, as far as I understand it, all VLANs will use the common
  spanning tree (CST) by default - through traffic engineering you can
  then optimize your network to group subsets of VLANs to use
  different trees (MSTI). To my understanding, the way this is
  typically managed in silicon is roughly:

  Incoming packet:
  .----.----.--------------.----.-------------
  | DA | SA | 802.1Q VID=X | ET | Payload ...
  '----'----'--------------'----'-------------
                          |
                          '->|\     .-----------------------------.
                             | +--> | VID | Members | ... | MSTI |
                       PVID -->|/    |-----|---------|-----|------|
                                     |   1 | 0001001 | ... |    0 |
                                     |   2 | 0001010 | ... |   10 |
                                     |   3 | 0001100 | ... |   10 |
                                     '-----------------------------'
                                                                   |
                                  .--------------------------------'
                                  |  .-------------------------.
                                  '->| MSTI | Fwding | Lrning |
                                     |------|--------|--------|
                                     |    0 | 111110 | 111110 |
                                     |   10 | 110111 | 110111 |
                                     '-----------------------'
```

Linux as a switch operating system

Pavel Šimerda

speaker@simerda.eu