Data Center Bridging Attributes

John Fastabend
LAN Access Division,
Intel Corp.

LINUX PLUMBERS CONFERENCE
SEPTEMBER 7-9, 2011 SANTA ROSA, CALIFORNIA

Agenda

History & Background Knowledge

Use Cases (Do we need a single API)

DCB Infrastructure net_device model

DCB Infrastructure HBA model

Common API HBA+net_device

Discussion/Next Steps

History and Background

Data Center Bridging (DCB):

- Originated as Converged Enhanced Ethernet
- currently IEEE standards 802.1Qaz, 802.1Qbb, 802.1Qau
- better known as ETS, PFC, and CN
- or, enhanced transmission selection,
  priority flow control and
  congestion notification

Fiber Channel over Ethernet (FCoE):

- Fiber Channel requires a lossless link
- PFC provides the “lossless” link characteristic for FCoE

iSCSI with DCB

- open-iSCSI DCB enabled, *Mon, 20 Jun 2011 – iscsid: Add IEEE DCB support - Mark Rustad*
- Allows networking stack to use ETS to provide minimum bandwidth guarantees and PFC

Can a common DCB API be used to improve application and management policies?

Multipath I/O

ETS: {LAN, SAN, Control}
PFC: {TC1, TC2, TC3}

PATH A:

ETS = {20%, 60%, 20%}
PFC = {0, 1, 0}

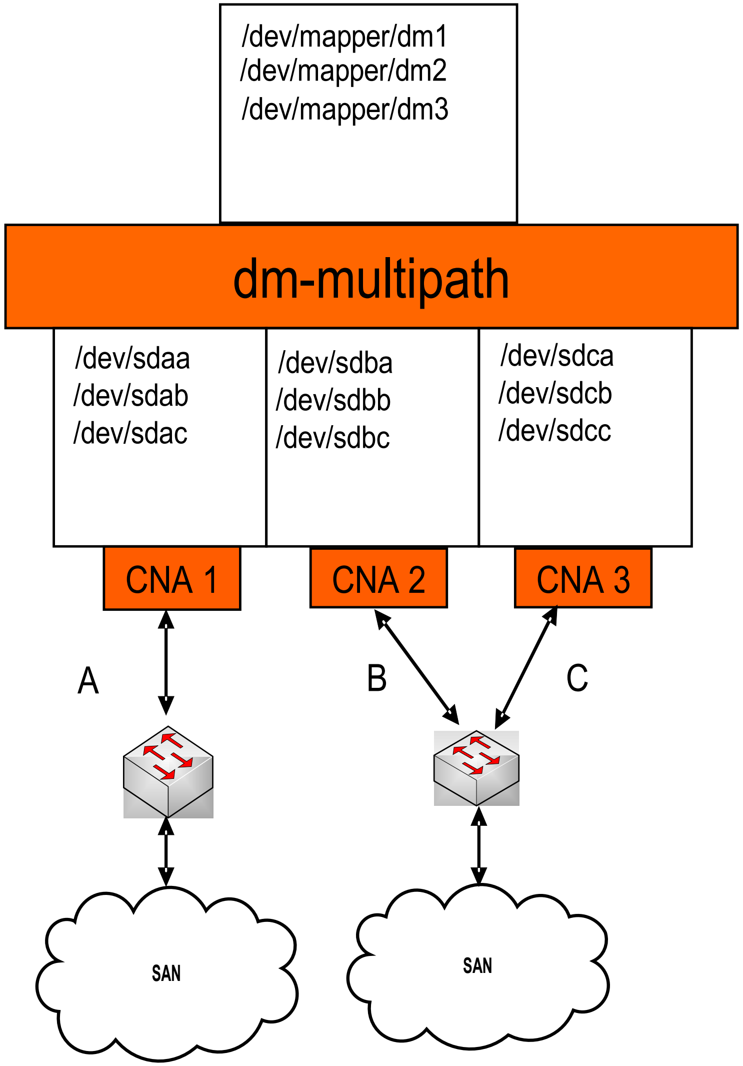
PATH B:

ETS = {60%, 20%, 20%}
PFC = {0, 1, 0}

PATH C:

ETS = N/A
PFC = N/A

Not all links are created equal!



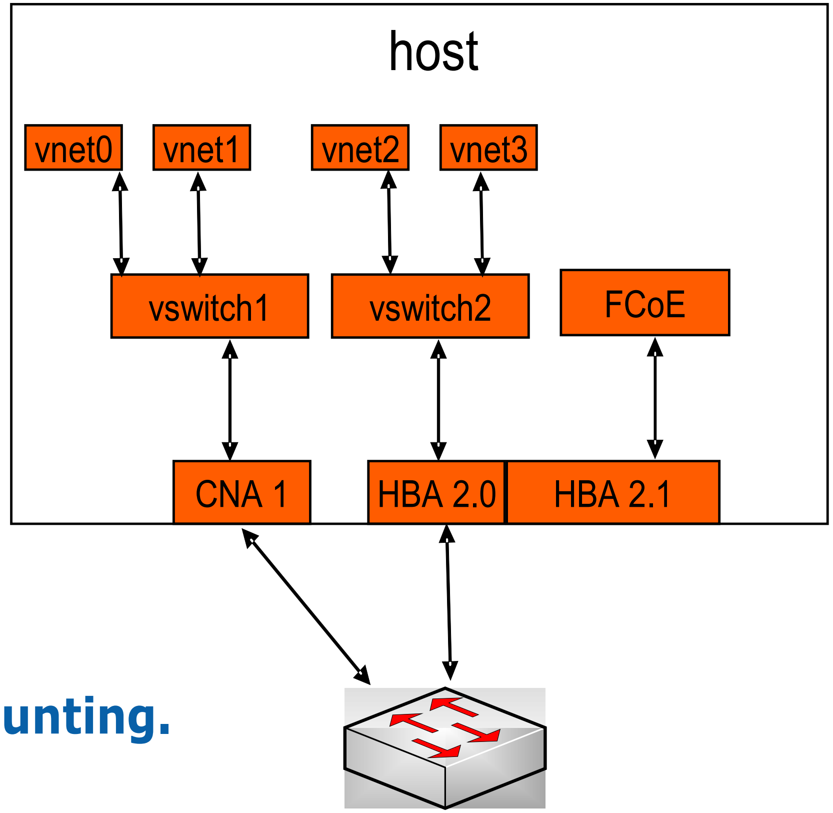
Multi-function devices

ETS: {LAN, SAN, Control}
PFC: {TC1, TC2, TC3}

CNA1:
ETS = {20%, 60%, 20%}
PFC = {0, 1, 0}

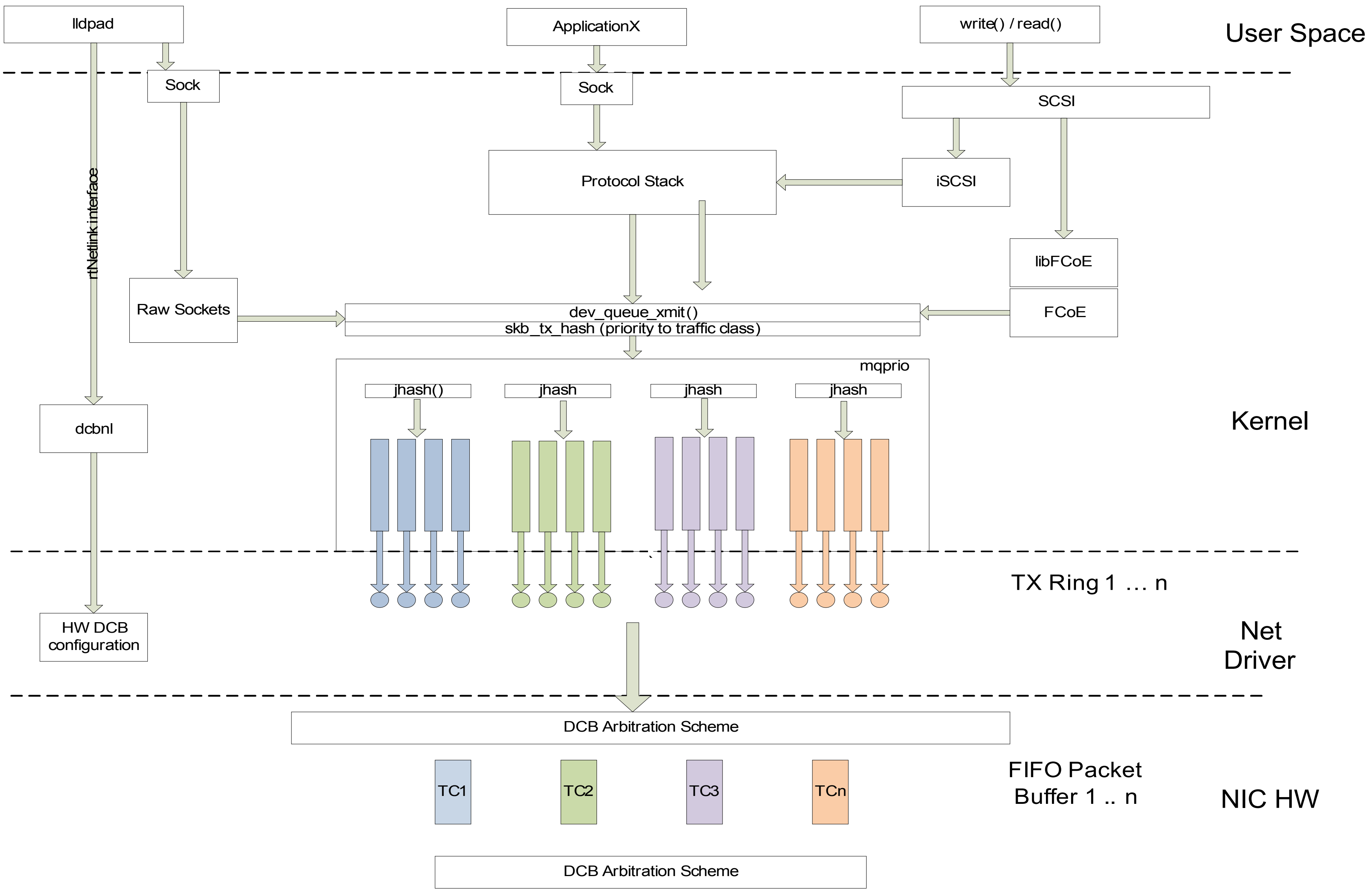
HBA2.0: (LAN function)
ETS = {30%, 60%, 10%}
PFC = {0, 1, 0}

HBA2.1: (SAN function)
ETS = 20%
PFC = enabled



HBA2.1 outside of LAN QOS and accounting.

Linux: Xmit Converged DCB Stack



dcbnl

- L2 net devices standardizing on DCB netlink
- Uses rtnetlink interface and a series of dcbnl_rtnl_ops

```
./include/net/dcbnl.h:

struct dcbnl_rtnl_ops {
        /* IEEE 802.1Qaz std */
        int (*ieee_getets) (struct net_device *, struct ieee_ets *);
        int (*ieee_setets) (struct net_device *, struct ieee_ets *);
        int (*ieee_getpfc) (struct net_device *, struct ieee_pfc *);
        int (*ieee_setpfc) (struct net_device *, struct ieee_pfc *);
        int (*ieee_getapp) (struct net_device *, struct dcb_app *);
        int (*ieee_setapp) (struct net_device *, struct dcb_app *);
        int (*ieee_delapp) (struct net_device *, struct dcb_app *);
        int (*ieee_peer_getets) (struct net_device *, struct ieee_ets *);
        int (*ieee_peer_getpfc) (struct net_device *, struct ieee_pfc *);
```

```
./include/net/linux.h

enum dcbnl_commands {
                    […]
        DCB_CMD_IEEE_SET,
        DCB_CMD_IEEE_GET,
```

- Updates generate events and kernel notifier hooks.
- Supports firmware and user space implementations.

But it uses the net_device as a key!

HBA DCB Attributes

- Qlogic (qla2xxx), sysfs entry “dcbx_tlv”
  - read only, mailbox command to firmware
- Brocade (bfa), Block SCSI Generic vendor requests
  - read only, CEE/DCBX parameters ‘bfa_cee_dcbx_cfg_s’
- Emulex (lpfc), ???
  - Did not find any interface to expose attributes.
- Other Drivers

No standard mechanism to export DCB attributes for HBAs.

Proposal #1 – DCBNL (preferred)

- Extend existing net_device DCB infrastructure
  - dcbnl_rtnl_ops uses net_device as lookup key.
  - Use unique_id of scsi_host as a key.
- Development

(1) provide dcbnl_rtnl_ops equivalent with unique_id lookup

(2) add new netlink attribute SCSI_HOST_ID

(3) wire up drivers

- Encourage driver vendors and application maintainers.

Proposal #2 – sysfs

- Add DCB attributes to SYSFS
- Where would these be placed?
  - Minimally iSCSI and FCoE should be able to lookup attributes.
  - L2 drivers may not be able to easily find scsi_host?
  - Possible options lport, scsi_host#, …

- This fragments CNA and HBA interfaces.

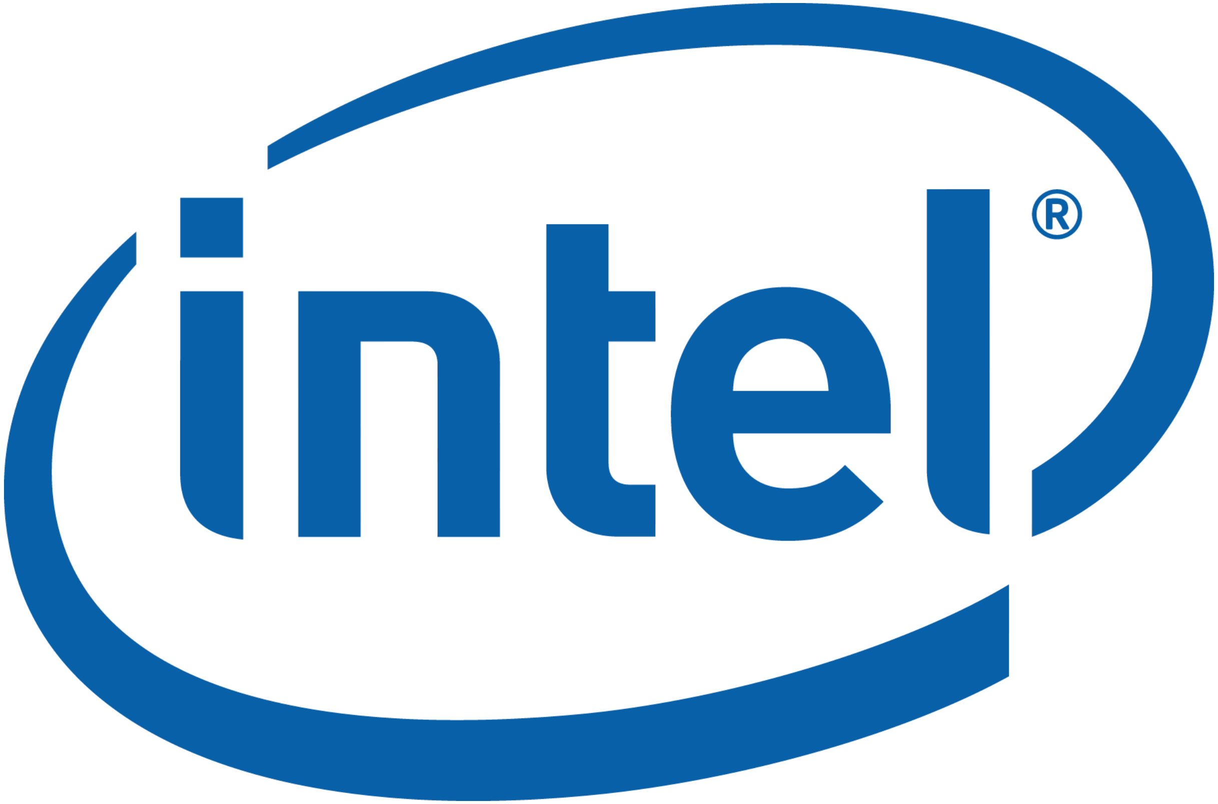
Proposal #X – Other Ideas

- BSG
  - Creates a standard for FCoE devices
- HBA-API
  - Requires updates to standards.
  - Does not preclude common kernel interfaces
- No Changes
  - Applications can continue to workaround fragmented interfaces

- Other idea I may not have thought of?

Comments / Next Steps

THANKS!


intel®