DPDK LTS
- Biennial release cadence
- Current LTS DPDK 16.11
- Next LTS DPDK 18.11

Only Bug fixes backported

Customer does not expect new features, API or ABI LTS changes until DPDK 18.11.
LTS is a very good thing!

- DPDK 16.11 LTS has ~300 post 16.11.0 bug fixes
- Without LTS a DPDK 16.11 stable user would be missing ~200 bug fixes.

Kudos to Yuanhan
OVS has 6 month release cadence with back ports to recent releases.

OVS 2.7 supported DPDK 16.11 LTS

- LTS support a contributing factor to removing experimental tag from OVS DPDK.

Internal debate within OVS community

- When to change DPDK version?
- Should OVS stick with DPDK LTS only?
  - Yes: Provides stability and clear roadmap visibility for new features, api/abi changes for OVS DPDK.
  - No: Waiting for LTS (2 years) – too long, no new features until 18.11.
LTS: Recommendations

- 17.11 should be LTS
- 2 years support to be maintained for both 16.11 and 17.11.
- Review approach to LTS over the coming year
  - Should all .11 releases be LTS?
  - Is 2 years support still required?
- Thoughts?
What is API/ABI stability?
- The helpful thing that stops my DPDK application breaking
- The annoying thing that stops my DPDK code merging now

Why is API/ABI stability important?
- Allows users who dynamically link easy update to new DPDK releases
- After some stability, most recent DPDK releases are breaking API/ABI compatibility
- Open vSwitch on Fedora had to revert to static linking because of ABI breaks
- At least knowing about API/ABI break in advance allows for planning
- Balance of stability and allowing progress needed
Changes to DPDK tool names and file structure are considered API breakages.
- Example: dpdk-devbind.py name or file path changes.
- Deployment code must be re-written to facilitate these changes.

Changes to Makefiles/Build System count as ABI breakage from packaging perspective.
- Example 1: Previously disabling KNI required only setting CONFIG_RTE_LIBRTE_KNI=n.
- CONFIG_RTE_KNI_KMOD=n added to allow disabling the kni kernel module separately.
- CONFIG_RTE_KNI_KMOD did not respect the value of CONFIG_RTE_LIBRTE_KNI.
- Example 2: Build system changes requiring new dependencies.
- New options should respect previous behavior.

Customer ideal deployment:
- A point where DPDK dlls could be used with OVS DPDK across DPDK versions without requiring recompilations.

Debian/Ubuntu: Painful moving up one version, libfoo1 links to libbar1 in 16.11 but to libbar2 in 17.02, causing breakages.
Try and avoid API/ABI breaks
- Try to make API/ABI more resilient against breaks
  - [dpdk-dev] [PATCH v3 00/20] vhost ABI/API refactoring
- Use multi-lib versioning where possible
  - major version as ABI revision - CONFIG_RTE_MAJOR_ABI
- Deprecate with 1 release notice
- Give time for discussion, 3 Acks required
- Collate changes to try and avoid multiple API/ABI breaks
- Run ABI checker tool
API/ABI: Discussion

- No API/ABI breakage between LTS’s (More stable)
  - + Offers multi-release stability for users
  - - Can impact complexity of code for devs
  - - Was proposed last year, but turned down due to stable tree availability

- Allow API/ABI breakage with 1 release deprecation notice (Current)
  - + Gives warning to users and avoids code complexity for devs
  - - Not much incentive to keep stable, results in API/ABI breakages

- Allow API/ABI breakage on every release (Less stable)
  - + Allows new code to be implemented in simplest form for devs
  - - Users will have no stability or even warning of breakages
Questions?

John Mcnamara: john.mcnamara@intel.com
Ian Stokes: ian.stokes@intel.com
Kevin Traynor: ktraynor@redhat.com
Luca Boccassi: bluca@debian.org