Introduction to the Transponder Abstraction Interface - TAI

Scott Emery and Wataru Ishida

What’s the Problem?

- Optical transceivers are complex and expensive parts of an optical system
  - 100’s or 1000’s of registers/memories
  - Most have SDKs, NDAs, Licensing, etc.
  - Each have different interfaces
- Optical transceiver vendors invest a lot of money to develop their parts
  - Want to maximize their investment, not give it away
  - Would like to operate in as many systems as possible
- System vendors invest a lot of money to develop systems
  - Want to maximize their investment
  - Would like to support as many optical components as possible
- Open optical packet transponder systems have created this opportunity for component and system vendors to work with a larger ecosystem

Has this problem already been solved?

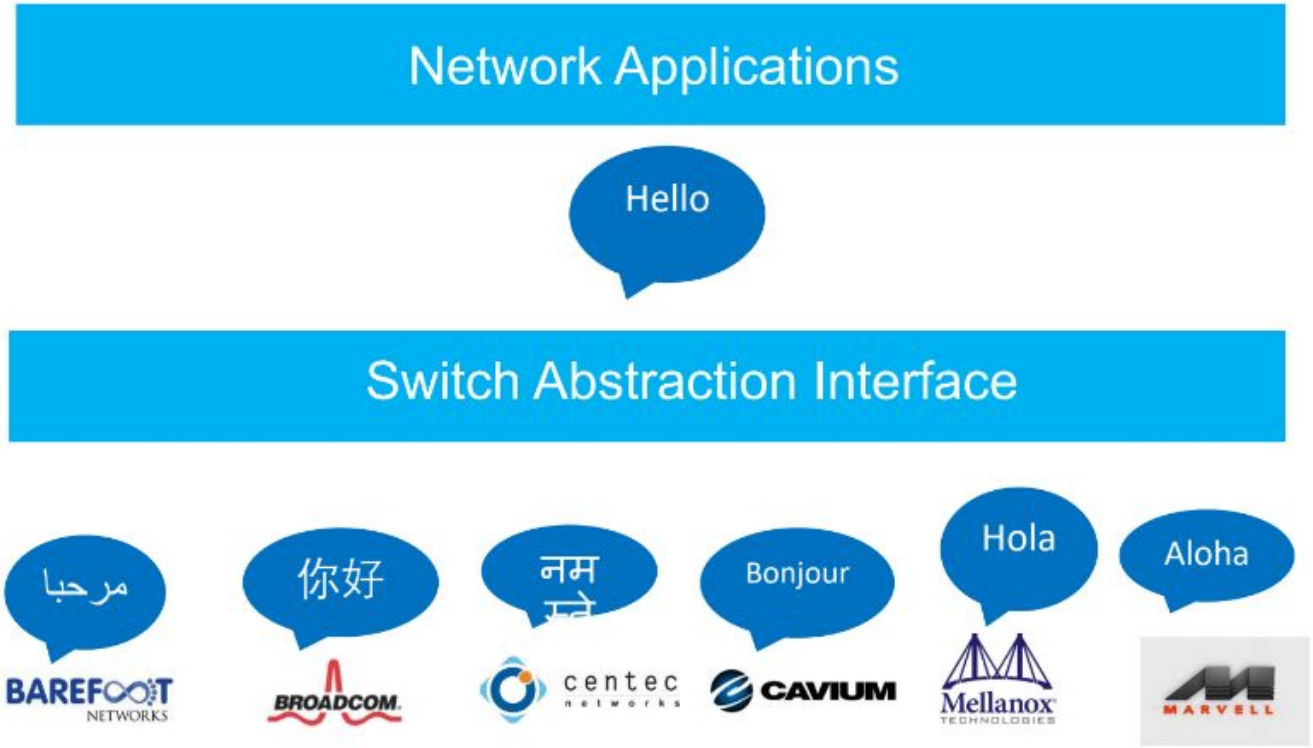
Switching ASICs vs. Optical Transceivers

- Both are very complex chips
- Both have many high speed interfaces
- Both commonly have embedded CPUs and are controlled by external CPUs
- Vendors of both don’t like to expose the details of their implementation
  - NDAs, Specs marked Confidential
- Both have 100s/1000s of registers/memories for status/control
- Both commonly have SDKs
- Optical modules can be modular, switching ASIC are fixed
  - Voyager optics are fixed, Cassini optics are modular
- Since TIP has the same “roots” as OCP, TIP looked to OCP for how to handle this...

SAI - Switch Abstraction Interface

- A project within the OCP networking group
- Began in 2015
  - Significant adoption since inception
- Designed to be as light-weight as possible
  - Modeled after SDKs
- Uses CRUD operations (create, read, update, delete) over an extensible data model
  - Objects are created/deleted which have attributes that can be read/updated
  - Attributes can be easily extended
- TAI “stole” much of its philosophy from SAI
  - 99% of the time, the answer to “Why does TAI do …?” is “Because that’s the way SAI does it”
- https://github.com/opencomputeproject/SAI

Switch Abstraction Interface (SAI)



Simple, consistent, and stable network application stack

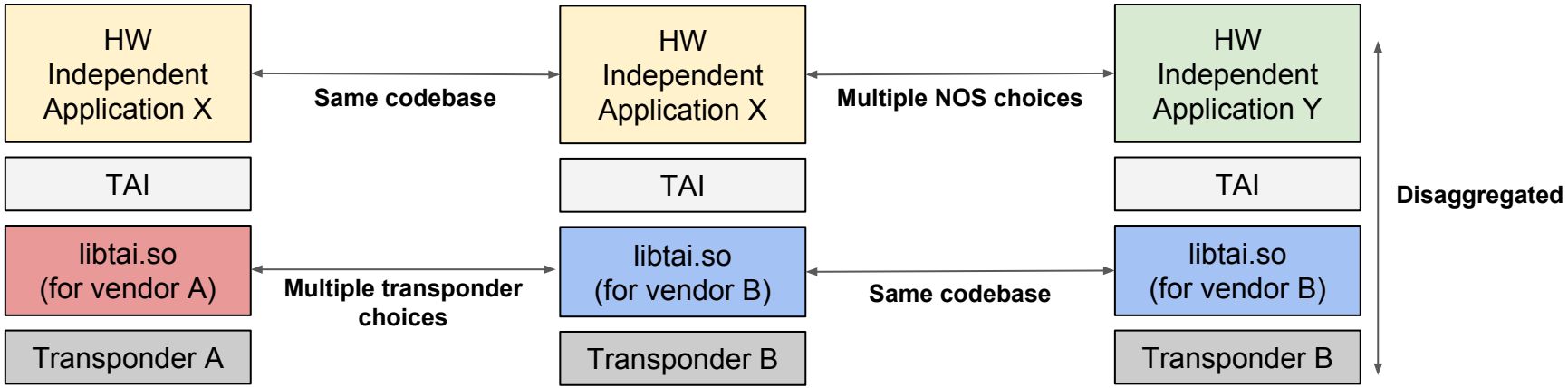
Helps consume the underlying complex, heterogeneous hardware easily and faster

SAI Becomes TAI

- **T**ransponder **A**bstraction **I**nterface
- Collection of C header files
- Implementation (C shared library - libtai.so) will be provided by each component vendor
- Users link libtai.so with their application
- Provides a common interface to optical modules
- libtai.so is binary blob, protecting optical vendors
- Current version:
  - https://github.com/Telecominfraproject/OOPT-TAI

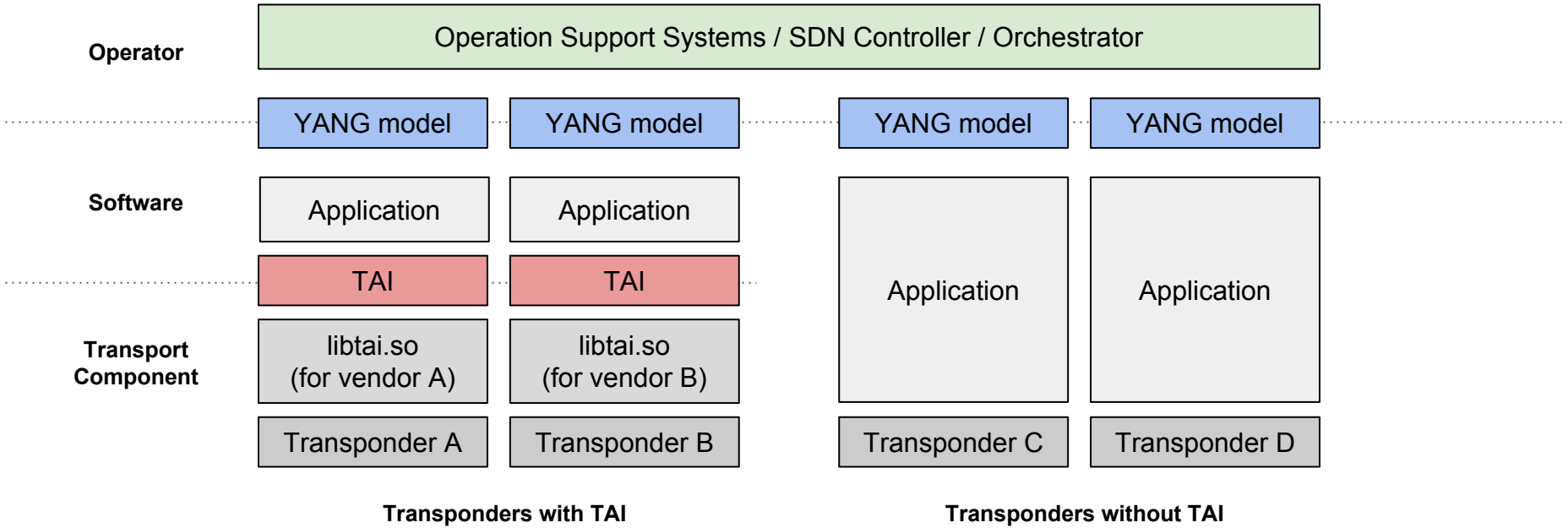
What is TAI?

- TAI is an interface between optical transponders and system software
- Allows system software to operate with any TAI-compliant transponders
- Allows transponders to operate in any system which supports TAI
- By decoupling the transponders from the rest of the system, it allows each to innovate independently
- Available here:
  - https://github.com/Telecominfraproject/oopt-tai
  - https://github.com/Telecominfraproject/oopt-tai-implementations



What is not TAI?

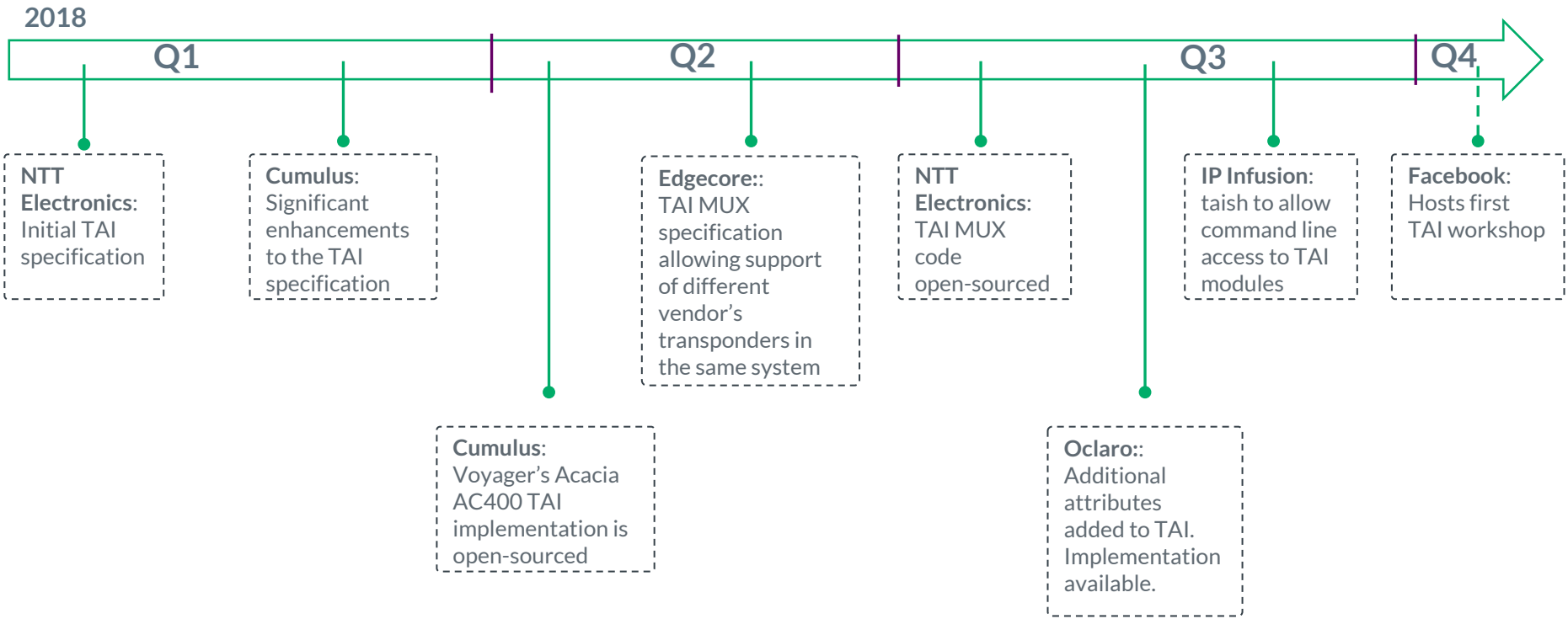
- TAI is not an API for operators like YANG models
- TAI is not trying to become a de jure standard or standardization body



Differences between Switches and Transponders

- Transponders are commonly modular - switching ASICs are fixed
  - Optical transceivers come and go as a system operates
    - Voyager transceivers are fixed
    - Cassini transceivers are modular
  - Switching ASICs are soldered down
    - There are no known open switch platforms where a switching ASIC can be removed/added in a running system
- There can be several, different optical transceivers in a system
  - Optical transceivers from different vendors can be present in the same system
    - Voyager has only one vendor - Acacia
    - Cassini supports ACO and DCO CFP2 modules from many vendors
  - Switches typically have a single ASIC from one vendor
- Both of these differences are handled in TAI, but not in SAI

TAI Timeline



TAI Supporters

What’s next for TAI?

- More features to exploit rich hardware capability
- More features to ease transponder software development
- Become a good mediator between YANG models (operator’s API) and MSA MIS register maps (hardware API)
  - TAI does not just expose the MSA MIS
- Enlarge the community and accelerate the disaggregation of transponders