Civil Infrastructure Platform: Industrial Grade SLTS Kernel and Base-Layer Development

Yoshitake Kobayashi
Open Source Summit Japan, Tokyo, June 2, 2017
Our Civilization is Run by Linux
But there are issues to be solved…
A Railway System:

25-50 years products life-cycle

with very reluctant nature for product update and upgrade of hardware and base software platform
Railway Example

3 – 5 years development time

2 – 4 years customer specific extensions

1 year initial safety certifications / authorization

3 – 6 months safety certifications / authorization for follow-up releases (depending on amount of changes)

25 – 50 years lifetime

Image: http://www.deutschebahn.com/contentblob/10862328/20160301+Stw+M%C3%BClheim+Innenansicht+1+(1)/data.jpg
What we have done on Linux for civil infrastructure systems

• Improve real-time performance and test
• Improve reliability and test
• Improve security and test
• Improve stability and test
• Create a lot of documents and review
  • Open source software licenses compliance
  • Export control classification

• Then, support for long-time such as 20-60 years
• ...
We have a problem...
The Problems we face …

- The systems that support our modern civilization need to **survive for a VERY LONG TIME**. Until now the corresponding industrial grade super long term maintenance has been **done individually by each company**.

- These systems not only have to survive for a long time, they must be “**INDUSTRIAL GRADE**” (robust, secure and reliable). And at the same time the industry will also need to **catch up with the latest technology trends**.
The Solutions we need ...

- **We need a Collaborative framework** to maintain the same open source based system for many, many, many years to keep it secure, robust and reliable.
- **AND most importantly, we need to do this collaboratively in the upstream communities**, not locally.
Establishing an Open Source Base Layer of industrial-grade software to enable the use and implementation of software building blocks for Civil Infrastructure Systems

https://www.cip-project.org/
Requirements for the Civil infrastructure systems

Industrial Grade
- Reliability
- Functional Safety
- Security
- Real-time capabilities

Sustainability
- Product life-cycles of 10 – 60 years

Conservative Upgrade/Update Strategy
- Firmware updates only if industrial grade is jeopardized
- Minimize the risk of regressions
- Keeping regression test and certification efforts low

This has to be achieve with ...

Maintenance costs
- Low maintenance costs for commonly uses software components
- Low commissioning and update costs

Development costs
- Don’t re-invent the wheel

Development time
- Shorter development times for more complex systems

Open Source Summit Japan 2017
Things to be done: Creation of an “Open Source Base Layer”

- Open source based reference implementation
- Start from a minimal set for controllers in industrial grade systems

Non-CIP packages
Any Linux distribution (e.g. Yocto Project, Debian, openSUSE, etc.) may extend/include CIP packages.

- CIP Reference Filesystem image with SDK (CIP Core packages)
- CIP SLTS Kernel
- CIP Reference Hardware
Scope of activities

**User space**
- App container infrastructure (mid-term)
- Shared config. & logging
- Multimedia

**Kernel space**
- Safe & Secure Update
- Monitoring
- Security
- Real-time support
- Real-time / safe virtualization

**Middleware/Libraries**
- Domain Specific communication (e.g. OPC UA)

**Tools**
- Build environment (e.g. yocto recipes)
- Test automation
- Tracing & reporting tools
- Configuration management
- Device management (update, download)
- Application life-cycle management

**Concepts**
- Functional safety architecture/strategy, including compliance w/ standards (e.g., NERC CIP, IEC61508)
- Long-term support Strategy: security patch management
- Standardization collaborative effort with others
- License clearing
- Export Control Classification

**On device software stack**

**Product development and maintenance**

Open Source Summit Japan 2017
### Technical topics and related projects (Feb. 2017 version)

*Topics will be added or removed to reflect CIP technical interests*

#### Application support
- App Framework
- HMI Framework
- FW update
- App deployment

#### Middleware / Tools

##### Security
- Anomaly detection
- Live patching
- LSM
- SELinux

##### Functional Safety
- Monitoring/error detection
- SIL2LinuxMP
- SIL3 support

##### Userland Isolation
- LXC
- Cgroups

##### Kernel Isolation
- SafeG
- Jailhouse
- Communication

##### Real-time support
- GPGPU/FPGA real-time
- RT/non-RT communication
- Xenomai
- PREEMPT-RT

##### Monitoring / Tracing
- Ftrace
- ktap
- RAS

##### Heterogeneous Computing
- SoC FPGA

#### Linux Kernel

##### Verification
- kselftest
- CIP test suite

##### Build and production
- Yocto Project
- Debian build system

##### Testing
- LTP
- Fuego
- CIP test suite

##### Support
- SLTS
- Backwards compatibility

##### Development process
- SIL2 support
- SIL3 support

##### Legal topics
- SPDX
- FOSSology

##### License Clearing
- Export Control

#### Hardware / SoC (x86 or ARM based)

#### Coherent Security Mechanisms

- Industrial special-purpose protocols
- Avnu

#### Industrial special-purpose protocols
- OPC UA
- TSN
- Echonet
- OM2M

#### Multimedia
- Configuration
- Industrial Zeroconf

#### Common issues
- Y2038

#### Industrial Zeroconf
- Domain specific and IoT communication
- Industrial Zeroconf

#### Functional Safety
- SIL2LinuxMP
- SIL3 support

#### Communication
- Jailhouse
- Communication

#### Real-time support
- GPGPU/FPGA real-time
- PREEMPT-RT

#### Monitoring / Tracing
- Ftrace
- ktap
- RAS

#### Hardware / SoC (x86 or ARM based)
- Industrial special-purpose protocols
- Avnu

Legend
- To be specified / implemented by CIP
- Integration / cooperation
Scope of activities

User space

- App container infrastructure (mid-term)
- App Framework (optionally, mid-term)
- Domain Specific communication (e.g. OPC UA)
- Shared config. & logging
- Middleware/Libraries
- Multimedia
- Monitoring
- Security
- Safe & Secure Update
- Real-time support
- Real-time / safe virtualization

Kernel space

- Linux Kernel
- On-device software stack

Tools

- Build environment (e.g. yocto recipes)
- Test automation
- Tracing & reporting tools
- Configuration management
- Device management (update, download)
- Application life-cycle management

Concepts

- Functional safety architecture/strategy, including compliance w/ standards (e.g., NERC CIP, IEC61508)
- Long-term support
  Strategy: security patch management
- Standardization collaborative effort with others
- License clearing
- Export Control Classification

Product development and maintenance

On-device software stack

Open Source Summit Japan 2017 16
Current status of CIP base layer development

• CIP SLTS kernel development
  • Decide the CIP kernel version
    • 4.4 is the first CIP kernel. Maintenance expected for 10 years and more (SLTS).
  • Select a maintainer
    • Ben Hutchings is the initial CIP-kernel maintainer
  • Define a kernel maintenance policies
    • https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance
  • Start maintenance
    • Linux 4.4.69-cip4 released on 25th May 2017
  • Create CIP kernel test framework

• CIP core package development
  • Define an initial component set
  • Define component version
  • Contribute to upstream project
  • Start maintenance for SLTS
CIP SLTS Kernel Development
Overview of CIP SLTS kernel

• Kernel trees
  • CIP SLTS (linux-4.4.y-cip)
    • Official CIP SLTS kernel tree
      • Based on linux-stable.git
    • Maintainer: Ben Hutchings
    • Validation will be done by CIP
  • CIP SLTS+PREEMPT_RT (will be separately maintained by CIP members)
    • CIP kernel tree based on linux-stable-rt and patches from CIP SLTS
    • Validation will be done by CIP

• Maintenance period
  • 10 years and more (10-20 years)
CIP SLTS Kernel development trees

Mainline

Stable (linux-stable)

CIP SLTS (linux-4.4.y-cip)

Stable-rt

CIP SLTS-rt/with FB

1. Feature backports

2. Security fix only

Follow the CIP SLTS with PREEMPT_RT

Take over from maintainer

Take over from maintainer

Maintained by Ben Hutchings

Backported patches

+PREEMPT_RT

Validate by CIP members

Open Source Summit Japan 2017
CIP SLTS Kernel development

• Kernel maintenance policy
  • [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance)
  • Follow the stable kernel development rule as the basis
  • Feature backports are acceptable
    • All features has to be in upstream kernel before backport to CIP kernel
    • **CIP has “Upstream first” policy**
  • Validation will be done by CIP test infrastructure and/or members

• Current backported features on 4.4.y-CIP
  • Kernel Self Protection Project related features
    • Address Space Layout Randomization for user space process (ASLR)
    • GCC’s undefined behaviour Sanitizer (UBSAN)
    • Faster page poisoning
CIP’s participation in the Real-time Linux Project

• CIP has become a Gold Member of the Real Time Linux Project

• What’s next
  • Work together with the RTL Project
  • CIP member Daniel Wagner (Siemens) is trying to become the maintainer of 4.4.y-stable-rt, the base version of the CIP Kernel.

• More information
  • https://wiki.linuxfoundation.org/realtime/rtl/start
Out-of-tree drivers

• In general, all out-of-tree drivers are unsupported by CIP
• Users can use CIP kernel with out-of-tree drivers
  • If a bug is found in such a modified kernel, users will first demonstrate that it exists in the CIP kernel source release in order for the CIP maintainers to act on it.
Major version release cycle (Next CIP SLTS kernel version)

- CIP will take a LTS kernel every **2-4 years**
- Planning to synchronize with LTSI for next CIP SLTS kernel
  - LTSI: http://ltsi.linuxfoundation.org/
CIP Kernel testing
Purpose of CIP testing

- Detecting bugs
- Detecting regressions
- Provide test results in a timely manner
Milestones of CIP testing and current status

1. Board at desk - single dev
   - A setup that allows a developer to test the CIP kernel on the CIP selected hardware platform connected locally to her development machine using kernelCI tools.

2. CIP kernel testing
   - Test the CIP kernel on a regular basis and share the results with other CIP community members.

3. Define kernel testing as a service within CIP
   - Define the testing environment within CIP assuming that, in some cases, some members may share the tests, test results or laboratories while others may not.

4. From kernel testing to system testing
   - Once the testing environment has been ready and works for the kernel, explore how to extend it to the entire CIP platform.

https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting
CIP testing

• Goal
  • Create and publish a VM image that contains KernelCI & LAVA
  • Single developer can test the CIP kernel (or any other kernels)

• News
  • B@D v0.9.1 has been released at OSSJ 2017
    • https://www.cip-project.org/news/2017/05/30/bd-v0-9-1
  • Download the VM or deploy the environment through Vagrant
    • https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipdownload
  • Check the tools and software packages included in this release.
    • https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptestingboardatdesk singledevfeaturepage
    • The CIP testing team has invested a significant effort in writing step by step instructions to deploy, configure and run tests.

• Check the source code involved
  • https://gitlab.com/cip-project/cip-testing/board-at-desk-single-dev/tree/master
CIP testing: next steps

• During the coming months the team will focus on:
  • Defining how tests should look like.
  • Defining how results should be shared.
  • Increasing the test coverage of the CIP Kernel

• More updates at Embedded Linux Conference Europe 2017 this October
CIP Core Package Development
Current status of the Base layer development

1. Define an initial component set
2. Define component version
3. Contribute to upstream project
4. Start maintenance for SLTS
Current status of the Base layer development

1. Define an initial component set
   1.5 Talk to upstream maintainer
2. Define component version
3. Contribute to upstream project
4. Start maintenance for SLTS
Initial component set for CIP base layer

CIP will start with a minimal set of packages. “CIP kernel” and “CIP core” packages run on hardware.

Candidates for initial component set

- **Kernel**
  - Linux kernel 4.4 + backported patches
  - PREEMPT_RT patch
- **Bootloader**
  - U-boot
- **Shells / Utilities**
  - Busybox
- **Base libraries**
  - Glibc
- **Tool Chain**
  - Binutils
  - GCC
- **Security**
  - OpenSSL

**CIP Core Packages**

Keep these packages for Reproducible build

- **Flex**
- **Bison**
- **autoconf**
- **automake**
- **bc**
- **bison**
- **Bzip2**
- **Curl**
- **Db**
- **Dbus**
- **Expat**
- **Flex**
- **gawk**
- **Gdb**

- **Git**
- **Glib**
- **Gmp**
- **Gzip**
- **gettext**
- **kbd**
- **Libibverbs**
- **Libtool**
- **Libxml2**
- **Mpc$$\text{lib}$$**
- **Mpc$$\text{fr}$$**
- **Ncurses**
- **Make**
- **M4**
- **pax-utils**
- **Pciutils**
- **Perl**
- **pkg-config**
- **Popt**
- **Procps**
- **Quilt**
- **Readline**
- **sysfsutils**
- **Tar**
- **Unifdef**
- **Zlib**

**NOTE:** The maintenance effort varies considerably for different packages.
CIP Project X

• Started an incubation project for the minimum base system
  • This project will provide a way to test the installable image

• Goal
  • Input: Debian sources/binaries and cip kernel
  • Build mechanism: bitbake and/or Debian build system
  • Output: Minimum deployable base system

• Current status
  • Minimal rootfs available for the following hardware
    • QEMUx86
    • BeagleBone Black
    • Cyclone-V

• Source code
  • https://gitlab.com/cip-playground/project-x
Development plan

CIP will increase the development effort to create an industrial grade common base-layer

Phase 1:
- Define supported kernel subsystems, arch.
- Initial SLTS component selection
- Select SLTS versions
- Set-up maintenance infrastructure (build, test)

Phase 2:
- Patch collection, stabilization, back port of patches for CIP kernel packages
- Support more subsystems
- Additional core packages

Phase 3:
- Domain specific enhancements, e.g. communication protocols, industrial IoT middleware
- Optionally: more subsystems
- Optionally: more core packages
CIP whitepaper release

- Year One Update + Whitepaper Release
  - [https://www.cip-project.org/blog/2017/05/31/cip-year-one-update-whitepaper-release](https://www.cip-project.org/blog/2017/05/31/cip-year-one-update-whitepaper-release)

- Everyone can download the whitepaper
  - [https://wiki.linuxfoundation.org/_media/civilinfrastructureplatform/whitepaper_short.pdf](https://wiki.linuxfoundation.org/_media/civilinfrastructureplatform/whitepaper_short.pdf)
Summary

• Selected the first CIP kernel and initial maintainer
  • 4.4 as first CIP kernel. Maintenance expected for 10+ years (SLTS).
  • Ben Hutchings as initial CIP kernel maintainer.
  • Defined CIP Kernel maintenance policies.
  • Defining CIP kernel + RT maintenance.

• Defined initial board platforms and provide support for them.
  • Renesas RZ/G and Beaglebone Black

• Released Board @ Desk for CIP kernel testing

• Started CIP Project X

• Published a whitepaper
Next Steps
Next steps by CIP

• Board @desk - Single dev
  • Start Action-2.
    https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting
  • Increase test coverage.

• Kernel maintenance
  • Define Kernel features
  • Create a branch for 4.4-cip-rt

• Analysis
  • Select additional software as part of CIP base layer.
  • Review requirements from CIP members (e.g. Functional Safety)

• Collaboration: kernelCI, LAVA, Fuego, y2038, KSPP, Real-time Linux Project
CIP booth at OSSJ 2017

• CIP use cases
  • Industrial controller
  • Power plant simulator with real controller
  • IoT (OpenBlocks IoT)
  • CIP testing on reference board (Renesas RZ/G)

• Whitepaper
Please Join us!
Why joining CIP?

• **Steer**
  
  participate in project decisions and technical direction.

• **Participate**
  
  bring your use cases and ideas to the right forum.

• **Learn**
  
  by working on daily basis in the open with others with common interest.

• **Collaborate**
  
  share effort and knowledge. Stand on the shoulders of giants.
Contact Information and Resources

To get the latest information, please contact:

- Noriaki Fukuyasu: fukuyasu@linuxfoundation.org

Other resources

- CIP Web site: https://www.cip-project.org
- CIP Mailing list: cip-dev@lists.cip-project.org
- CIP Wiki: https://wiki.linuxfoundation.org/civilinfrastructureplatform/
- Collaboration at CIP: http://www.gitlab.com/cip-project
Call for new participants!

Provide a super long-term maintained industrial-grade embedded Linux platform.

Platinum Members

Silver Members

Open Source Summit Japan 2017
Questions?
Thank you!