



Generic Boot Loader on Android platforms

Naina Mehta
CPSE Linux Kernel Team, Qualcomm India Private Limited
<nainmeht@qti.qualcomm.com>

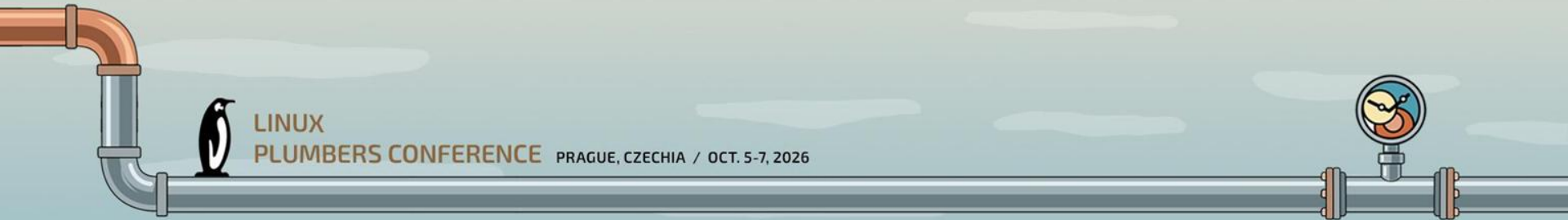


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

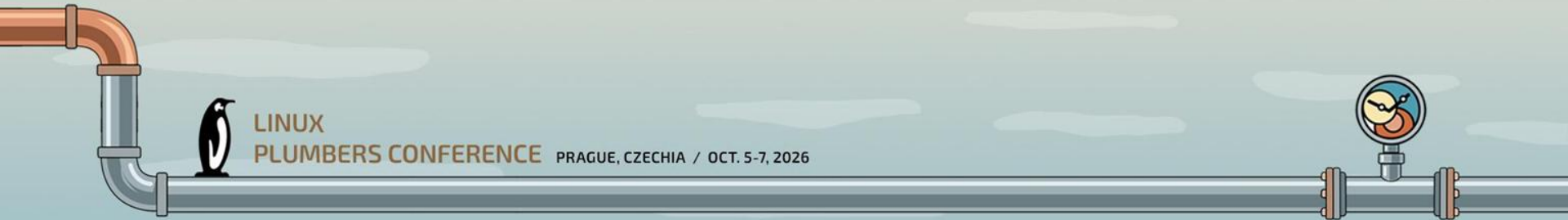
Agenda

- Implementation Overview
- Maintaining Feature Parity
- Device Tree Handling
- GBL Image Handling
- Debugging GBL with T32
- Challenges and Key Learnings



Implementation Overview

- Core UEFI services exposed - Refactored firmware to align with standard protocols such as DT fixup.
- GBL protocols implemented - Boot Control (A/B), OS Configuration, AVB and Fastboot protocols brought up, aligned to GBL's expected revisions.
- Fastboot preserved - Vendor and OEM commands retained via CommandExec, with custom transport and UI supported alongside GBL's generic implementation.
- Booting today - GBL loaded from the new android_esp partition, validated with VTS test.



Maintaining Feature Parity

Custom Bootloader

- Image loading and verification
- DT handling and selection
- Command line updates
- Boot config updates
- Runtime DT patching
- Fastboot handling

GBL

- Image loading and verification
- DT handling
- Call into UEFI for command line, bootconfig update
- Fastboot interface with host

Firmware Stack (U-Boot/EDK2/LK)

- DT selection
- Command line, bootconfig update
- Custom fastboot command implementation
- Custom fastboot UI



LINUX

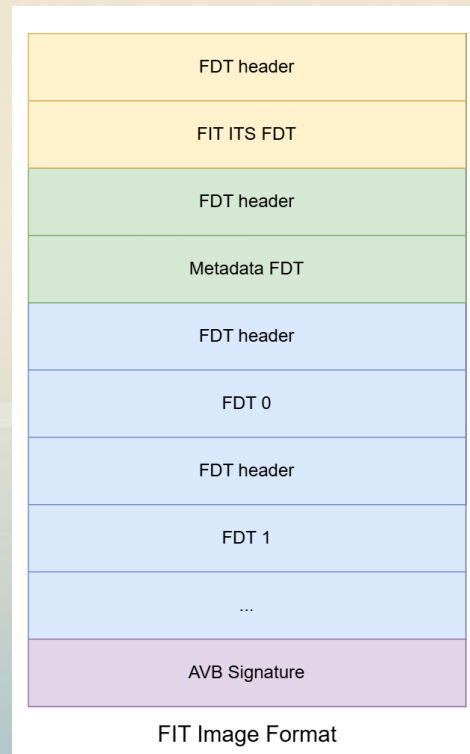
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



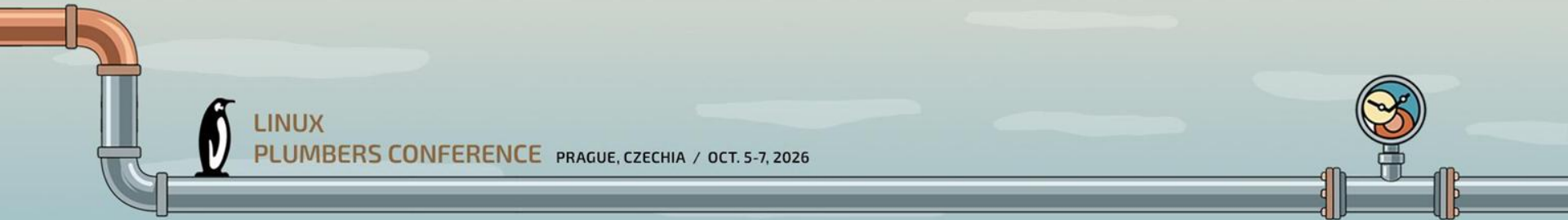
Device Tree Handling Via SelectFitConfiguration

- If FIT is found in dtbo partition, GBL loads FIT
- GBL passes FIT FDT and metadata to firmware
- Firmware selects appropriate FIT configuration and returns selected configuration node offset



GBL Image Handling

- Introduced new FAT partition android_esp_a/b
- Updated firmware for loading and running GBL application from /EFI/BOOT/BOOTAA64.EFI
- Updated A/B mechanism for GBL binary
- Added support for GBL binary authentication



Debugging GBL with Trace32

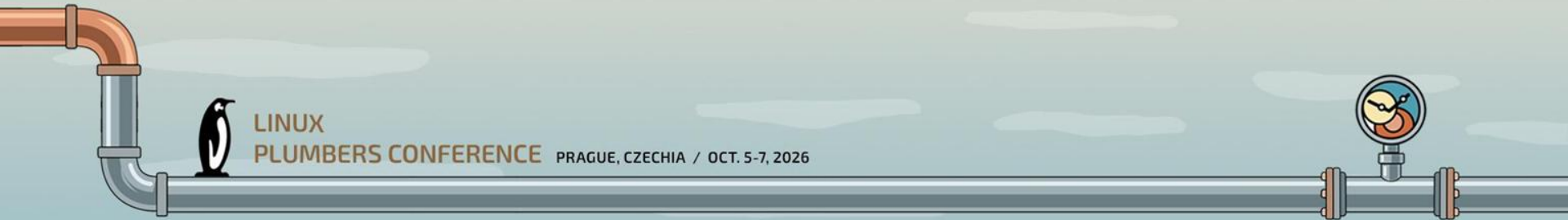
- Build GBL with debugging support

```
./tools/bazel run //bootable/libbootloader:gb1_efi_dist -c dbg  
--@gb1//toolchain:always_wait_gdb
```

Image base: <Start Address>

Set \$x0=0 from debugger to continue.

- The partition must be large enough to accommodate the debug image



Debugging GBL with Trace32

- Load GBL symbols, duplicate symbols can be seen due to inherent nature of RUST

```
d.load.exe <Path to gbl_aarch64_dev.pdb> <Image base address> /nocode  
/StripPART "gbl+" /SOURCEPATH <Path to \bootable\libbootloader\gbl>
```

```
symbol_-----  
-----  
|.:.android_main::closure_env$2<gbl_efi::ops::0ps,gbl_efi::android_boot::efi_android_load::closure_  
env$0> >  
|.:.android_main::closure_env$4<gbl_efi::ops::0ps,gbl_efi::android_boot::efi_android_load::closure_  
env$0> >
```

- Break, then set \$x0 register to 0 and go to resume



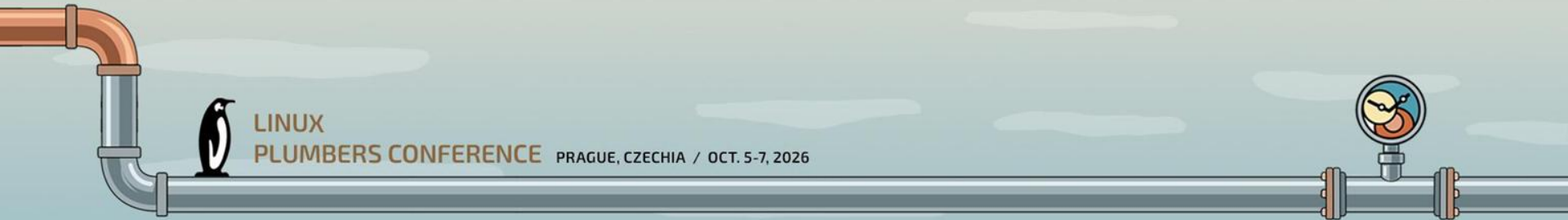
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



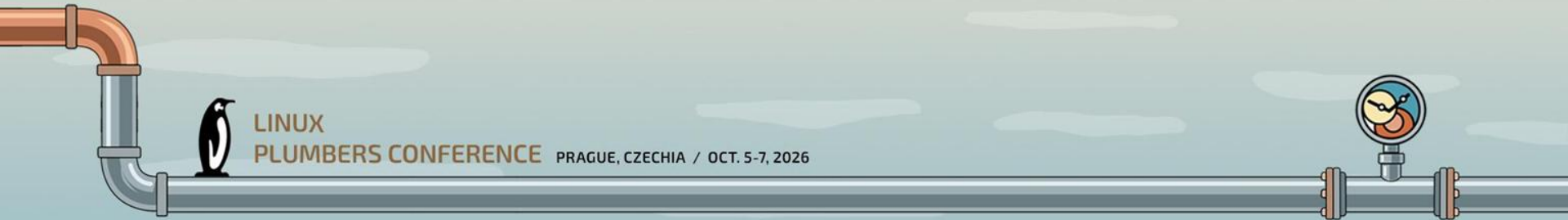
Challenges and Key Learnings

- The protocol version and interface in firmware must match with that defined in GBL.
- Appropriate android_esp partition size to be kept for long-term maintenance and debugging support.
- Care to be taken while debugging on T32 for duplicate symbols.
- Extensive test coverage to be maintained to avoid breaking feature parity.

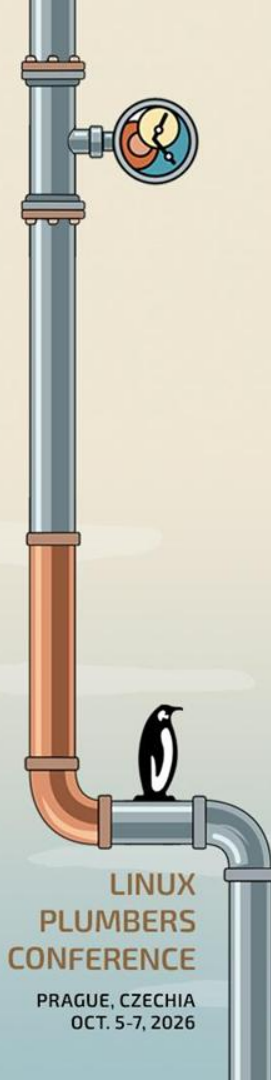


Resources

- [GBL Documentation](#)
- [FIT Documentation](#)
- Other Community Discussions:
 - <https://ossindia2026.sched.com/event/2KNHL/generic-bootloader-on-android-platforms-naina-mehta-qualcomm-india-private-limited>
 - <https://fossunited.org/c/indiafoss/2026/cfp/e08r7fa8ja>



Thank you



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026