AGENDA

• Bootloader Goals
• Typical Firmware Makeup on RDK devices
• Startup Flow
• Secure Boot Flow
• Bootloader characteristics
• Advanced Bootloader Requirements
• Advanced Bootloader Standard UI
• System Boot Elements and functionality
• Challenges
Main goals of the bootloader
- Security
- Robustness
- Performance
- Flexibility
**Typical Firmware Makeup on RDK Devices**

|--------------------|--------------------|----------------------------------------|----------------------------------------|-------------------------------|-------------------------------|-------------------------------|

- Stage-2 Bootloader (goal flexibility & robustness) also known as the advanced bootloader (ABL) is responsible for validating and loading the correct platform image.
- Stage-1 Bootloader is rather dumb (goal is robustness) and is mainly responsible for booting the stage-2 bootloader.
- There are two platform code images to ensure that a platform image is readily available as a backup in case the active image fails.
- Stage-2 Bootloader – validates each image in the order mentioned and launches the first valid image.
- If an image is valid, the stage-2 bootloader will attempt to launch it. If launch fails, it will update a failed retry count and continue launching till failed retry count reaches a configured threshold, after which it will proceed to launch the next valid image.
- The disaster recovery images have limited functionality and will need to download new platform code images for the device to be fully functional.
**Secure Boot Flow**

- **Bootloader may run image from flash or RAM depending on device memory constraints.**

---

**Device Key Store (Factory Programmed)**
**BOOTLOADER CHARACTERISTICS**

- **Stage-1 Bootloader**
  - Factory Programmed
  - Can validate signatures
  - Not field upgradeable
  - Chain of trust leading to SoC bootstrap
  - No support for firmware upgrade. Only launches the stage-2 bootloader.

- **Stage-2 Bootloader**
  - Factory Programmed
  - Field Upgradeable under special circumstances using a special firmware image. Strongly discouraged.
  - Supports Comcast Configuration Management System based firmware upgrade using TFTP, TR-69 and HTTP protocols.
  - Support for UBI, UBIFS, JFFS2, SQUASHFS, MTD based partitions.
**Advanced Bootloader Requirements**

- Bootloader is not field upgradable, there must be another mechanism to upgrade it.
  - Upgrade via console ports
  - Special firmware images
- Advanced Bootloader invocation
  - Front Panel Key combination on device
  - Software APIs
- Can work with SNMP signals to trigger update
- Return valid system descriptor
- Returns valid DHCP options
- Log transitions into advanced bootloader and initialization
Advanced Bootloader Standard UI

Get-top needs to download new software. This could take several minutes. Thank you for your patience.

DOCSIS COMMON DOWNLOAD

DOCSIS STATUS: <DOCSIS STATUS>
DOWNLOAD STATUS: <NOT STARTED/IN PROGRESS>
FILE NAME: <FILENAME>
TFTP SERVER: <TFTP SERVER IP ADDRESS>

PROGRESS INDICATOR:
SYSTEM BOOT ELEMENTS

- Disaster Recovery Image - Backup
  - Can Download Image
  - Performs Validation
  - Write image to flash
  - Supports IPv4
  - HTTP and TFTP protocol support
  - Can support DOCSIS
  - Supports DHCP
SYSTEM BOOT ELEMENTS

• Disaster Recovery Image
  – Supports TFTP/HTTP/HTTPS
  – IPv6/IPv4
  – external storage media (Developer mode)
  – Implements TR-069 Triggers for Code Download
  – Can upgrade platform firmware as well as itself
  – Works over ethernet/WiFi/MoCA
  – Supports Device Initiated Firmware download
  – Perform various image Validations
  – Cold Factory Reset
  – Remote Control Support
  – Front Panel buttons support
  – Supports DHCP
**CODE DOWNLOAD PROCESS**

- Device Initiated
- TR-069 Triggered
- IP Download
  - IP devices
  - Development Devices
NETWORKING

- Supports multiple network topologies
- DRI bridges all available I/Fs and initializes with DHCP if available
- Support for auto-IP/Zeroconf
USER INTERFACE

- Screen Diagnostics via Video Port
- Fixed resolution 720p e.g.
- Software Update progress Bar (OSD Screen)
CHALLENGES

• Non-standard bootloader across devices.
  – Inconsistences in extent to which they are file-system aware leading to special filesystems on different devices.
  – Inability to create a standard end-to-end reference device.

• Opensource bootloaders have stringent licensing requirements
  – Would be ideal to standardize on an available OSS bootloader.
  – However, bootloaders typically have OEM or SoC vendor secret sauce related to secure boot and other mandatory security functionality. GPL v2 and similar license interfere with their ability to protect their IP while leveraging these opensource bootloaders.
RDK BOOTLOADER

Thank You