Remote Firmware Update Manager for Embedded Systems

Authors: Santoshkumar Pattar, Venkata Narayanababu, Vasudev Prasad
Outline

- Introduction
- Takeaways
- Requirements/Challenges
- High Level Design (HLD) of Remote Firmware Update Manager (RFUM)
- RFUM @ Ittiam
- Questions
Introduction
Remote Firmware Update

- Upgrading/downgrading the networked embedded system’s firmware remotely
Need for Remote Firmware update

- Time to market
- Bug fixes
- Constant evolution/enhancement of software features
- Customer/Installation friendly
- Downgrade for emergency purposes
  - Unforeseen installation specific issues may force downgrade temporarily
- Cost reduction
Applications

Any **networked** embedded system
- Enterprise, military, government, defense, infrastructure or industrial systems

**Ittiam focus: embedded multimedia systems**
- Video Surveillance Systems
- Health Care Systems
- Telepresence and Video Conferencing Devices
- Home Networks
- Enterprise Streaming Systems, Digital Signage
- Digital Broadcast Systems
Firmware Introduction

Firmware Components (Generic embedded systems)

- **Boot Loader**
  - System boot loader
  - Loads the kernel

- **Boot configuration file**
  - Boot variables for the kernel

- **Kernel**
  - OS kernel

- **File System**
  - Compressed file system

- **FPGA Binary**
  - Software to be loaded on FPGA.

- **Slave Processor Firmware**
  - Firmware for the slave processors

**Boot Loader execution**

- Read boot configurations

- Locate OS kernel and boot the kernel

- Loading of other slave processors

- Kernel boot and loading the file system

- Execution of application

- FPGA binary Load

Execution of application
Non Volatile Memory Layout

- Boot Loader
- Factory Default Boot Loader
- Image 1
- Image 2
- Boot Config
- OS Kernel
- File System
- FPGA Binary
- Slave Processor Binary
Takeaways
Takeaways

- Requirements/Challenges
  - Considerations while designing RFUM
- High level Design of RFUM
Fail-Safe

- A failure in the update process should not make the device unusable
- A working firmware should always be present
- Interruptions to update process should be gracefully handled
Reliability

- Check if firmware image matches the device
- Should take care of transmission errors – Integrity check
- Verify the contents written on memory – compute checksum
- Should not exceed allocated non-volatile memory partition size
Requirements/Challenges (3)

Security

- Should be secure from “Man In the middle attacks”
- Authenticity of source of firmware
- Firmware image should be encrypted
- Firmware image should not contain any information which makes the end device vulnerable to attacks
Requirements/Challenges (4)

Low Downtime

- System should have a low downtime during the update process
- Update should happen in the background
- Application features should not be disrupted
- Update process should be light and should not load the system
Buffer/Memory constraints

- Firmware image could be in order of few MBs
- Lesser free space in embedded systems
- Mechanism to split the firmware into smaller manageable chunks
Requirements/Challenges (6)

Different file transfer mediums

- TFTP
- FTP
- HTTP
Requirements/Challenges (7)

- Different storage mediums (with or without filesystem)
  - NAND Flash
  - NOR Flash
  - Hard disk
  - USB
  - SD Cards
Smart boot loader
- Should be capable of updating itself
- Should have backup default boot loader
- Should be capable of decrypting firmware

Compatibility
- Should be forward and backward compatible
Design of RFUM
Proposed firmware storage layout on device

- A ping pong (two images) approach – Reliability, Fail safe
- Active firmware and Inactive firmware image
- Tracking of active firmware
- Always Boot with active firmware
- Upgrade/downgrade Inactive firmware
- Configurability for setting active firmware
Non Volatile Memory Layout

- Smart Boot Loader
- Factory Default
  - Smart Boot Loader
- Image 1
- Image 2
- Boot Config
- OS Kernel
- File System
- FPGA Binary
- Slave Processor Binary
Upgrade scenario

Before

v2.0
v1.0

Upgrade to v3.0

After

v2.0
v3.0

Active Image
Alternate Image
Downgrade scenario

Before

- v2.0
- v1.0

After

- v2.0
- v1.5

Downgrade to v1.5
Unsuccessful Update

Before
- v2.0
- v1.0

Update to vX.Y unsuccessful

After
- v2.0
- Corrupted image

Active Image

Alternate Image

Before
- v2.0
- Corrupted image

Update to vX.Y successful

After
- v2.0
- vX.Y
Design of RFUM contd.

Application Design

Firmware Update Manager

Transfer Module: Abstracts transfer mediums. [TFTP/FTP/HTTP/File/etc.]

Buffer Module: Manages the chunk size to be written

Writer Module: Abstracts memory type. [NAND/NOR/HDD/etc.]
Smart Bootloader

- Smart Boot loader design
  - Capable of updating itself
  - Reads Active Image and tries booting
  - In case of failure tries to boot Alternate Image
    - Since previous boot was from Alternate Image, Alternate Image should be valid
    - Alternate Image could also be invalid in case of storage corruption
  - In case both failure boot from network
RFUM @ Ittiam (1)

- Reliable and secure
  - Encryption support
- Fail Safe
  - Two firmware support
  - Error detection and correct (ECC / CRC)
- Control/configurability
  - Through different user interfaces
    - Web UI / ONVIF / Command Line Interface
- Minimal memory footprint
- Minimal downtime
  - Update along with other operations
- Multiple products in field
Updating individual components

Boot configurations are part of the Firmware image

- File system size can be increased/decreased in subsequent firmware’s
- Boot environment (memory map, etc.) can also be updated if required

Updating of specific components of file system
QUESTIONS
About The Authors

Santoshkumar Pattar
- Lead Engineer, Video Communication & Networking
- santoshkumar.pattar@ittiam.com

Venkata NarayanaBabu
- Senior Engineer, Video Communication & Networking
- venkata.laveti@ittiam.com

Vasudev Prasad
- Senior Engineer, Video Communication & Networking
- vasudev.prasad@ittiam.com

Ittiam Website (www.ittiam.com)
Thanks
i think therefore i am