A Scalable, Cloud-Based Device Reprogramming Architecture

Panasonic
About Me

James Simister
Director of Consulting Services

• Panasonic Research & Development Company of America, Salt Lake City Lab
  – Software Developer for 30+ years
  – 20 years experience with Linux
  – 15 years working with embedded systems
  – Interests: Networking, Security, Cloud, …
Introduction
What Is a Device?

• A thing made or adapted for a particular purpose, especially a piece of mechanical or *electronic equipment*.¹

• Any piece of electronic equipment capable of executing code to perform some function.

¹. Google definition, emphasis added
Introduction
Is There a Problem?

• Abundance
  – Breadth: More kinds of devices available
  – Depth: More demand for each kind
• Device lifetime of 10+ (20+) years
Introduction
Is There a Problem?

• Time-to-market
  – Increasingly demanding
  – Dropped/incomplete features & enhancements

• Crowd-funded projects, small start-ups
  – Lack of experienced engineers
  – Lack of security experts
Introduction
What Is Device Reprogramming?

• Changing software (firmware) of a device
  – Updates
  – Enhancements
  – Add [or remove] features
  – Bug fixes
    • Application errors, security vulnerabilities, etc.
Introduction
Device Reprogramming: Challenges

• Current cost vs. future capabilities
• CPU capability/speed
• Memory & storage (disk/flash) capacity
• Connectivity & accessibility
• Bandwidth
Update Strategy
Manual or Automatic?

Manual Updates
• User in full control
• Inform user
• Motivate user
• Unknown timing

Automatic Updates
• Mfr. in full control
• Mandatory
• Scheduled
• Controlled
General Requirements

Fundamental Issues

• Security
  – How do you prevent attack (or loss of control)?

• Reliability
  – How do you account for failure?

• Scalability
  – How do you handle millions of updates?
Security

How Do You Prevent Attack?

Trusted Sources

- Where did the update originate?
- Should the user/device trust the source?
- Would source tampering be evident?

- Hashes, Digital Signatures
- Proof-Carrying Code
- Verification/Validation
Security

How Do You Prevent Attack?

Trusted Targets

• Where did the update go?
• Is the target authorized to accept update?
• Are the assets protected?

• Authentication
• Authorization
• Confidentiality
Security
How Do You Prevent Attack?

Trusted Channels

• Who has access to the infrastructure?
• Would in-transit tampering be evident?
• Can the installation be verified?

• End-to-end key distribution & encryption
• Non-repudiation
Reliability
How Do You Account for Failure?

• Failure is not an option

• Failure is reality
Reliability
How Do You Account for Failure?
Gracefully Adapt

- Storage issues
- Adjust size, bandwidth
- Retry, with back-off
- Verification
- Validation
Reliability
How Do You Account for Failure?
Roll Back

• Keep the previous image, revert
• Update again, to previous image
• Update the updater
  – Try again
Scalability

How Do You Handle Millions of Updates?

Convenience

• Enhancements
• Minor bug fixes
• Deploy slowly, at your convenience
• Low server capacity & bandwidth
Scalability
How Do You Handle Millions of Updates?

Urgency

• Security vulnerabilities
• Major bugs
• Deploy quickly
• High server capacity & bandwidth
Scalability
How Do You Handle Millions of Updates?
Shared, Cloud-Based Infrastructure

• Scale up to meet demand
• Scale down to reduce costs
• Share costs of setup & maintenance
• Pay for what you use
Requirements→Implementation

Defining a General Process for Scalable, Cloud-Based Device Reprogramming

1. Publish the update image
2. Determine population of eligible targets
3. Determine scheduling constraints
4. Reprogram each eligible target
5. Report progress
Requirements → Implementation
Reprogramming Each Eligible Target

A. Obtain authorization for update
B. Failsafe transition to Reprogram mode
   – Failure reverts to Normal mode, no change
C. Transfer new image and update
D. Failsafe transition to Normal mode
   – Failure reverts to Reprogram mode, retry
Requirements → Implementation

Two Images: Normal, Reprogram

- Reprogram image significantly smaller
  - Custom Linux kernel and/or initrd
  - Reduce dependencies & features
- Objectives:
  - Obtain updated image
  - Roll back
Requirements→Implementation
Bootloader, Hardware Support

• Atomic switching of boot image
• Atomic acceptance of booted image
  – Failure reverts to last accepted boot image
• Power failure detection, protection
  – Guarantee atomicity, quality of writes
The Update Process

1. Publish the Update Image

• OpenDOF provider
  – Image owner retains full ownership, control
  – Complete security model
    • Image owner (Trusted Source)
    • Device (Trusted Target)
    • Sessions (Trusted Channel)
The Update Process

2. Determine Population of Eligible Targets

- Version Service using OpenDOF libraries
  - Devices report type and software version
  - Authorized clients may query database
    - Devices of specific type
    - Devices running specific software version
    - Devices *not* running specific software version
The Update Process

3. Determine Scheduling Constraints

- Population size
- Time constraints
- Cost constraints
- Determine required scale
The Update Process

4A. Obtain Authorization

• Update Service using OpenDOF libraries
  – Notifies device of time frame to update
  – May include additional authorizations by
    • Manufacturer
    • Service provider
    • User
    • Device
The Update Process

4B. Failsafe Transition to Reprogram mode

- Atomically switch to Reprogram mode
- Reboot
- Reconnect
- Update Service accepts booted image
  - Verification of connectivity
The Update Process

4C. Transfer Image and Update

• OpenDOF requestor to image provider
  – Transfer image blocks
  – Leverage UDP
    • Reduce buffering
    • Block caching
  – Verify image, signatures, etc.
The Update Process

4D. Failsafe Transition to Normal mode

• Atomically switch to Normal mode
• Reboot
• Reconnect
• Update Service verifies new version
• Update Service accepts booted image
  – Verification of connectivity
The Update Process
5. Create a Report

- Update Service tracks progress of devices
- Generate report
  - Scheduled
  - Started
  - Succeeded
  - Failed
Summary
A Scalable, Cloud-Based Device Reprogramming Architecture

• General, robust update process

• Services to automate process
  – Image
  – Version
  – Update

• Flexible OpenDOF libraries & protocols
Questions & Answers

https://opendof.org/