Introduction and Problem Statement draft-deng-lwip-ps

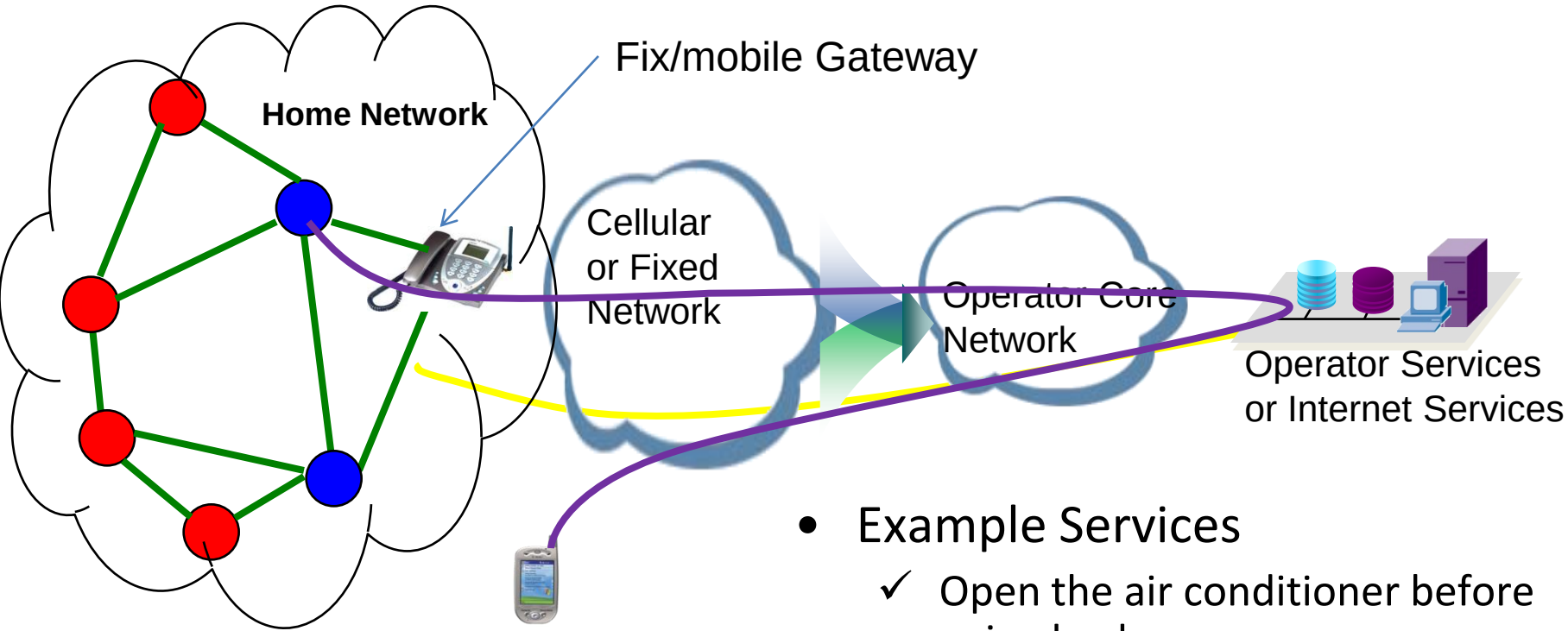
Hui Deng, China Mobile
Shoichi Sakane, Cisco
Wassim Haddad, Ericsson
Ning Kong, CNNIC

Introduction: Dancing with shackles

The Evolution of Computing: We keep inventing tiny and smart devices

- Challenges from Tiny and Small Devices
  - Communication overhead
  - Computation overhead, especially security computations
  - Energy Efficient: each byte transmitted consumes energy
  - Design challenge: dancing with shackles

Example Scenario



- Example Services
  - ✓ Open the air conditioner before going back
  - ✓ The sensor will send an alarming message to the host's mobile phone if home security issues are identified

Why IP?

- Technically: IP is fully inter-operable
  - IP runs over different lower layers
  - IP bridges different MAC protocols
- Service development:
  - IP facilitates the new service development
  - For private protocols, it is always difficult to add new services to the network
- Network operating cost:
  - IP connects products from different vendors, that's what people want
  - Simple network management, friendly UI

Lightweight TCP/IP stack

- Linux TCP/IP stack
  - ~100KB code, ~400KB RAM
  - µCLinux kernel ~400KB code, ~1 MB RAM
- Devices used in the Internet of Things

Chips	ROM	RAM
CC2431	128K Flash	8K
EM351	128K Flash	12K
ATMegal	128K+512K	4K

??

- Problem: how to implement the TCP/IP stack on the constrained devices

Existing Practices

- uIP
  - On 8-bit micro-controllers
  - less then 10kB ROM and 2kB RAM
- LwIP
  - around 40kB of RAM and 30kB ROM
- uC/IP
  - Based on BSD TCP/IP implementations
- BLIP
  - Berkeley Low-power IP stack on TinyOS
- TINYTCP
  - does not implement urgent pointers, and discards segments that are received out of order.

Implementation Issues

- Modularity and Layering
  - OS does not maintain a strict protection barrier between kernel and application
  - Layering: TCP checksum validation
- Memory Usage Constraints
  - Dynamic memory allocation could not go through on constrained devices
- Inefficient Socket APIs
  - The BSD socket API requires the support of a multitasking system which imposes a significant overhead due to the need for task management, context switching and allocation of stack space.
- Protocol Interoperability
  - Some do not handle options
  - Some do not handle fragmentation and reassembly

Root of question: Lack of Implementation Guideline

Thank you !