NDNoT
A Framework for Named Data Network of Things

Zhiyi Zhang, Yanbiao Li, Edward Lu, Tianyuan Yu,

Alex Afanasyev, Lixia Zhang

NDNcomm

Sept 2018

NDN fits IoT ideally

- **Securing data directly** instead of relying on secured sessions.
- Secure IoT system with **local trust anchors** instead of remote cloud servers
- Naming Convention -> An open environment for applications and services to cooperate and function together.
- By naming data instead of IP address, NDN enables host multihoming and seamlessly utilizes all communication interfaces
- **Content multicast** and in-network **caching**.
- Simpler application development

Motivations of NDNoT

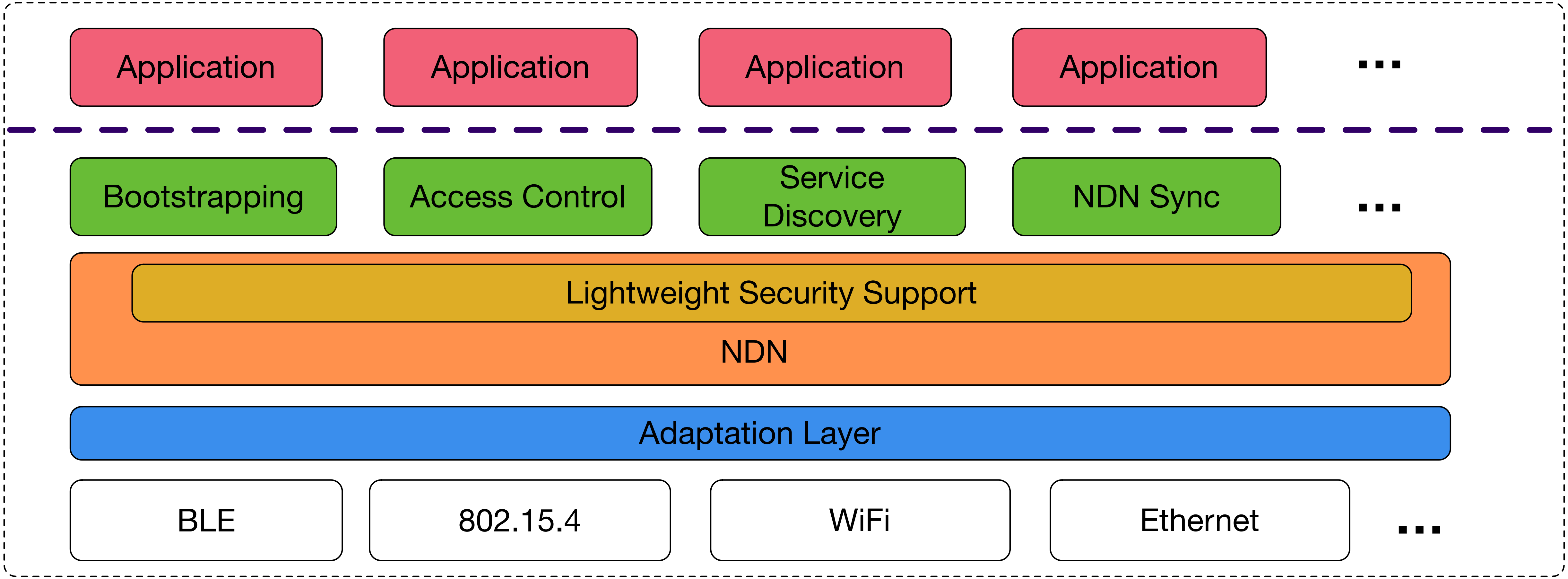
- Providing a integrate and modularized open-source library with well-documented APIs for developers and users to easily develop IoT applications with NDN.
- A community for IoT developers that are interested in applying NDN in IoT scenarios.
  - Some pre-defined naming convention for different services to cooperate
  - Encouraging developers to give comments or to add new functionality modules into the framework to make the framework better

Goals of NDNoT Library

Providing integrated and lightweight NDN support in IoT scenario:

- Basic NDN protocol stack and content-centric features
- Pure NDN running over link layer
- Security bootstrapping
- Service discovery
- Schematized Trust
- Access Control for constrained devices
- NDN Sync support

The Framework



A simple story

- You bought a smart home temperature sensor with a IoT board that only has 32k RAM and 48MHz
- What’s next?

Bootstrapping

Goal

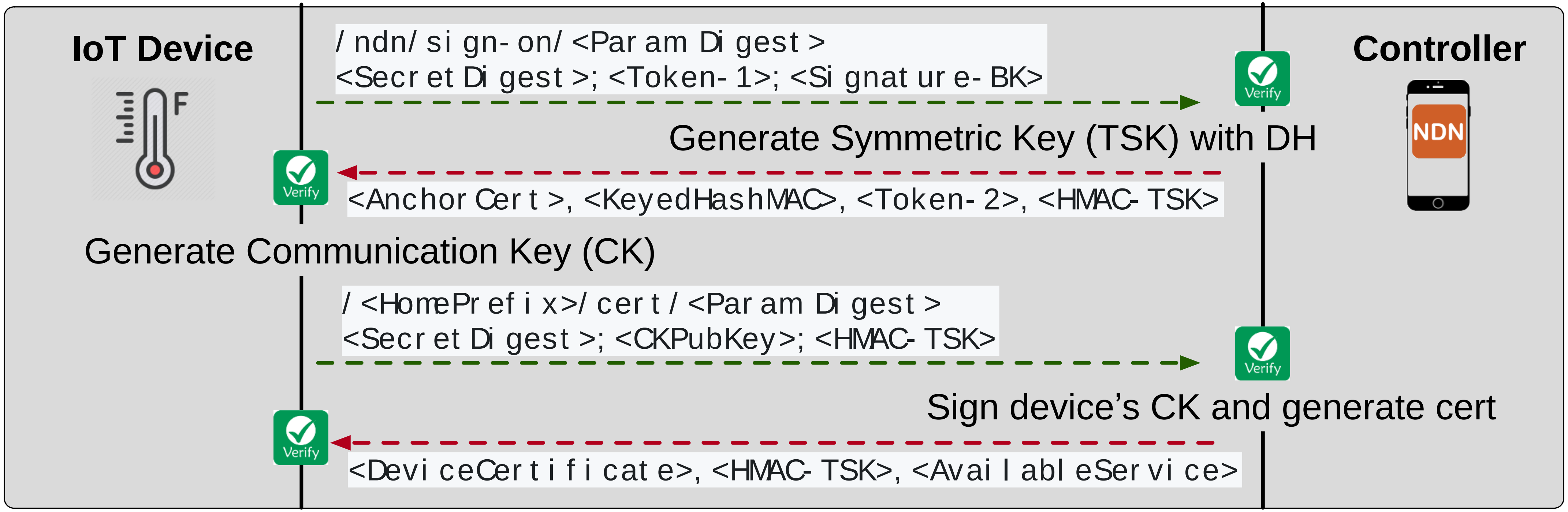
- The IoT device (e.g., Temperature Sensor) **learns the trust anchor** of the system and **obtain an identity certificate** issued by the system controller (e.g., Android Phone)

Assumptions

- The IoT device and the controller have **shared secret** through out-of-band means, e.g., The user scans the QR code on sensor with phone
- The pre-shared secret is a crypto **public key** (BK), e.g., ECC/RSA public key

Bootstrapping

- Identify each other by verifying the possession of shared secret.
- Negotiate a symmetric key for better performance
- Utilize uniqueness to prevent **replay attack**
- Use Interest parameter to save bandwidth



Bootstrapping Assessment and Performance

Assessment

- One asymmetric signature signing and verification (I1)
- One Diffie Hellman Process
- Three HMAC signing and verification (D1, I2, D3)

Performance:

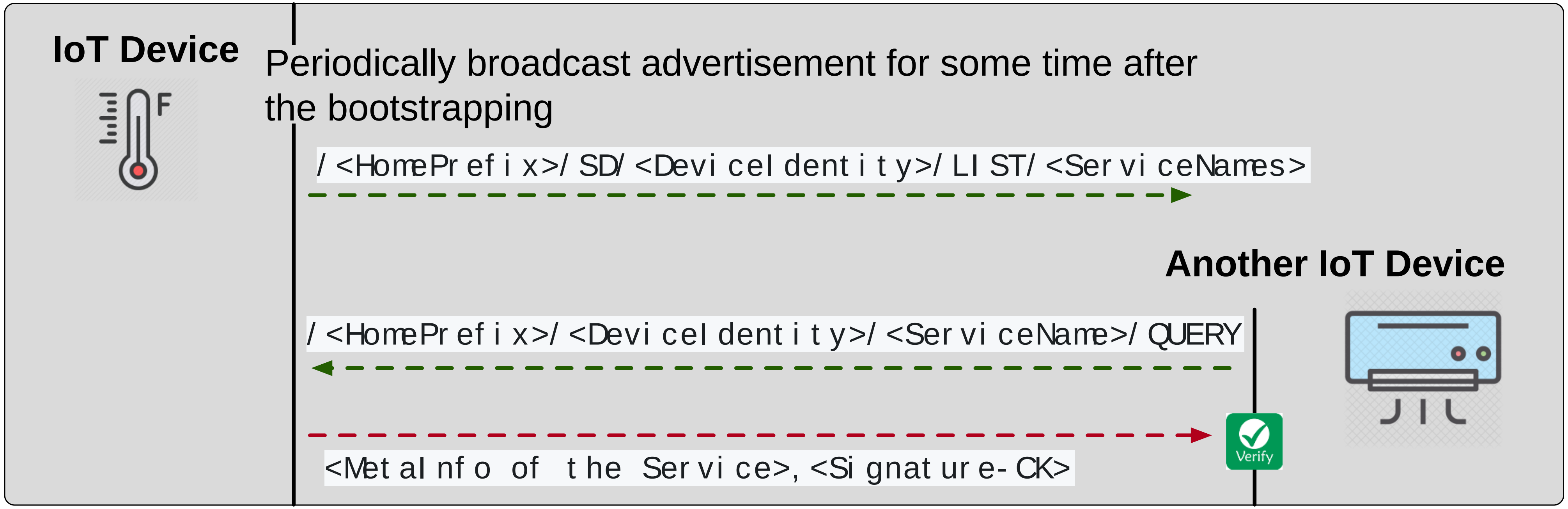
Time Consumption: **1.3s** (including network and system IO) for Xpro (with RIOT) board (32K RAM, 48MHz)

- Details: ECC key size 160 bits; DH key size 256 bits

Bandwidth Consumption: around 300 bits less by utilizing Interest parameters

Service Discovery

- Learning existing services from the controller in the last step of bootstrapping
- Advertising services by broadcasting advertisements after bootstrapping
- Broadcasting again when services change or restart (soft state)
- Query meta data before using a service



Schematized Trust

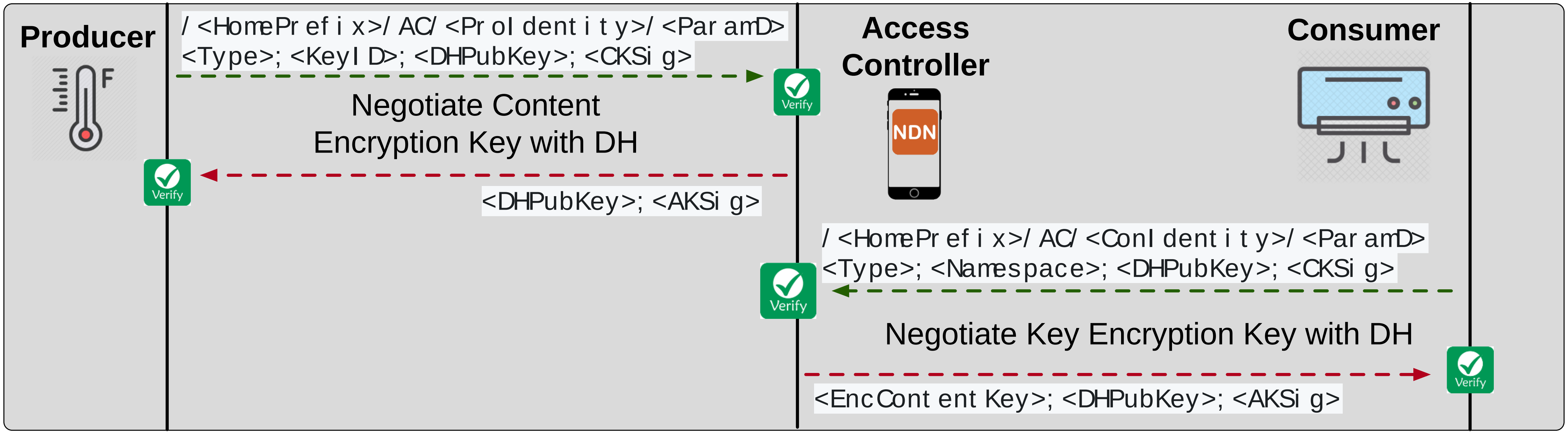
- Control your IoT device’s trust relationship with other devices in different scenarios

Example:

- The AC (/home/living/AC) should only trust the temp data (/home/living/temp) under the same prefix
- The AC should only obey the command signed by the device with controller prefix (/home/control) or with specific format (/home/living/remote-<>)

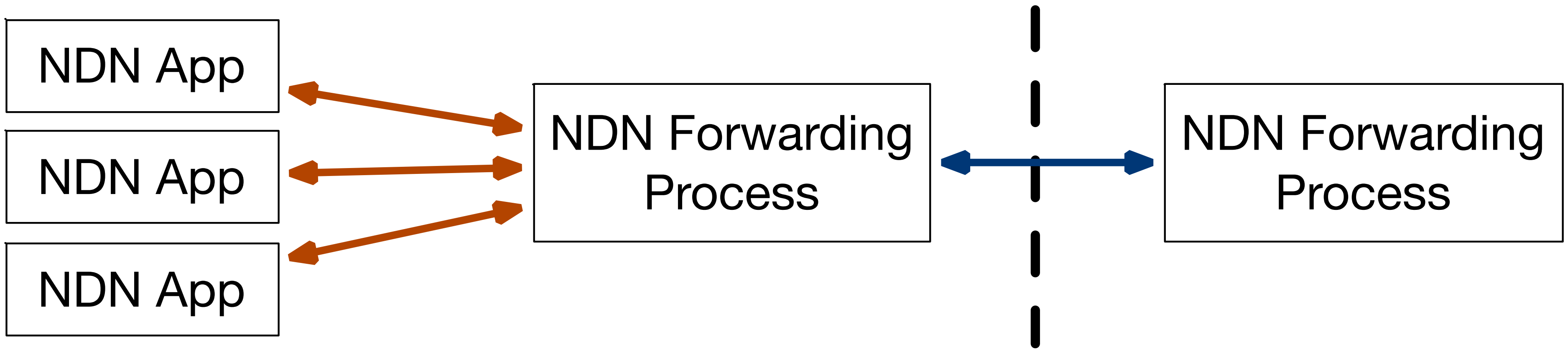
Lightweight Access Control

- Existing NDN access control systems don’t fit constrained devices
- All symmetric key encryption/decryption
- Use Interest parameter to save bandwidth



Adaptation Layer

- The adaptation layer abstracts different link-layer protocols and wraps the NDN Interest and Data packets into link-layer frames.
- Name Prefix <-> Interface mapping
- A separate process and communicates with NDN applications using Inter-Process Communication (IPC) or other equivalent mechanism.



Hardware

IOT devices

- Atmel Xpro (RIOT OS): 802.15.4
- ESP32: WiFi, BLE, Bluetooth

Controller

- Raspberry Pi
- Android Phone
- Linux/MacOS

Current status and future plan

- Finished with unit tests:
  - NDNoT for RIOT: Bootstrapping
  - NDNoT for RIOT: Service Discovery
  - NDNoT for RIOT: Access Control
- In Progress
  - Adaptation Layer
  - Specification
  - Tutorial
- Next stage
  - NDNoT for RIOT: schematized trust
  - NDNoT for RIOT: sync
  - NDNoT for RIOT: integrate test
  - NDNoT for ESP32

Thank You!

zhiyi@cs.ucla.edu