NDN Internet of Things Toolkit for Raspberry Pi

Adeola Bannis

UCLA

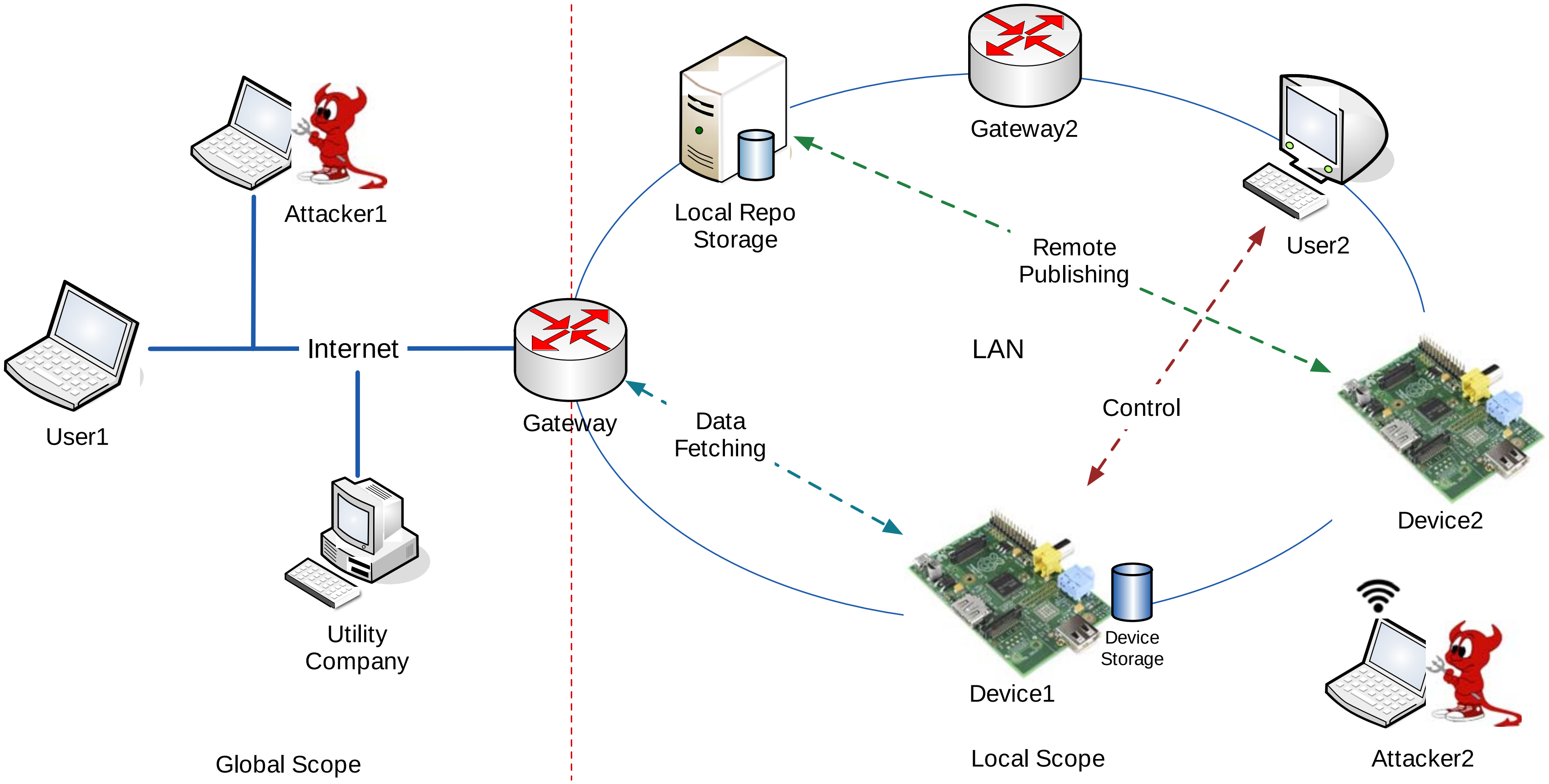
September 4, 2014

Goals

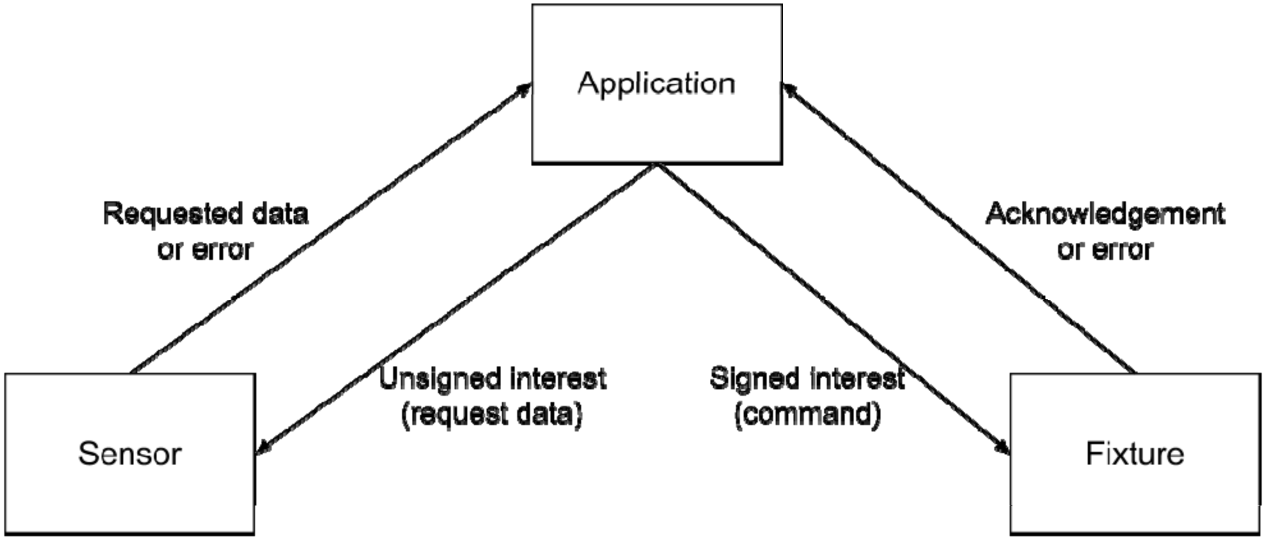
- To provide a framework for users to explore Named Data Networking
- To provide starter code and examples for a home network using NDN on Raspberry Pi
- To make network design and setup easy without hiding too much of NDN’s behavior

What is a Home Network of Things like?

Deployment Scenario



Main Components



- Application nodes issue commands to fixtures via signed interests and request readings from sensors via unsigned or signed interests
- Fixtures and sensors generally do not initiate communication with other nodes
- Fixtures and sensors must have a registered network name to receive interests; applications need a registered name to sign interests
- Master node/gateway is not shown here

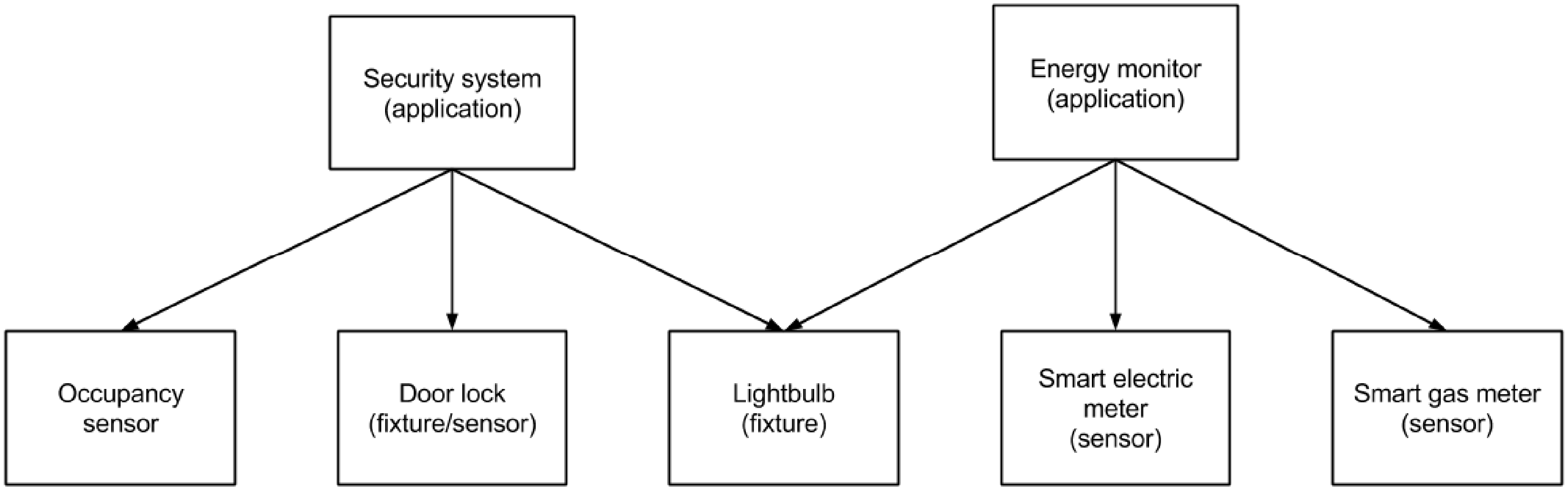
Home Network of Things

- May include many different types of sensors and fixtures from different vendors, e.g. thermostat, motion sensor, locks, lights
- Home automation uses aggregate sensor readings to change fixture state or alert users
- Users may also wish to check readings or control fixtures remotely
- Sensor readings may also provide home analytics, e.g. energy efficiency

Home Network of Things

- Requires interconnection layer accessible to all fixtures, sensors and user devices
- Must be accessible by low-resource microcontrollers as well as smartphones
- Must support at least requests for sensor readings and issuing of commands to fixtures
- Should prevent interference from unauthorized users (e.g. outside the home)
- Should minimize user intervention needed for setup

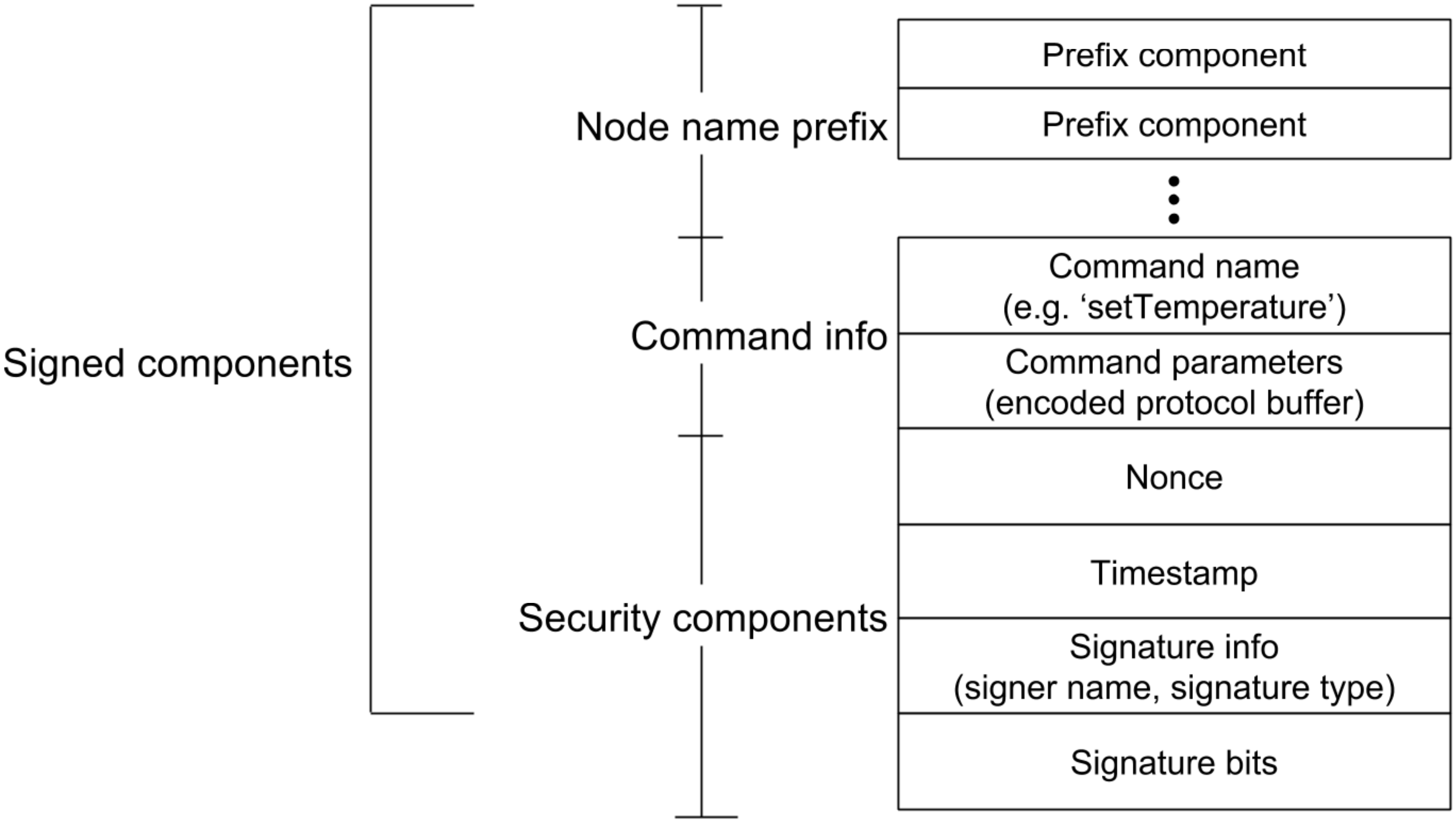
Example Network



Command Interest

- Signed version of interest, to ensure only authorized users issue commands
- Intended for a particular node, e.g.: /home/bedroom/thermostat
- Composed by appending command name, parameters, and signing information
- Command name is usually a verb, e.g. setTemperature
- Parameters are encoded as a single component using Google protocol buffers

Command Interest Name Format



Signed components

Node name prefix

Command info

Security components

Prefix component

Prefix component

Command name
(e.g. ‘setTemperature’)

Command parameters
(encoded protocol buffer)

Nonce

Timestamp

Signature info
(signer name, signature type)

Signature bits

What’s in the toolkit?

Toolkit Implementation

- Written in Python using PyNDN
- Examples use JSON for data instead of protocol buffers
- Manages NDN certificates so users don’t need to run ndnsec

Toolkit Design

- Assumes that only nodes in the same NDN namespace should be trusted to sign interests or data in that namespace
- Nodes are virtual; each device may run multiple nodes
- Each node may manage sensor, fixture or controller names
- User nodes should be subclasses of the basic node

Toolkit Design

- One master node manages security, is able to list all available commands in network
- Currently sensor and fixture nodes must be manually configured with namespace and master node name
- Security bootstrapping still in development

Toolkit Contents

- IoT Network classes
  - Controller
  - Node
  - Console
- Configuration utility for user nodes + networks
  - Set network, controller and device names
  - List commands with keywords

Toolkit Classes - Controller

- All nodes must connect with the controller and receive network certificates
- Controller also manages a directory of node capabilities
- Capabilities map command names to keywords that can be searched by other nodes

Toolkit Classes - Node

- User customization goes here
- Fixture, sensor or application nodes are all subclasses of this basic node
- User must use configuration utility to name the method associated with each command name
- User method takes the complete interest and returns a data object

Toolkit Classes - Console

- Helps in designing or troubleshooting a network
- Queries the controller for available devices
- Allows signed and unsigned interests to be issued manually
- Response data name and content are displayed to the user

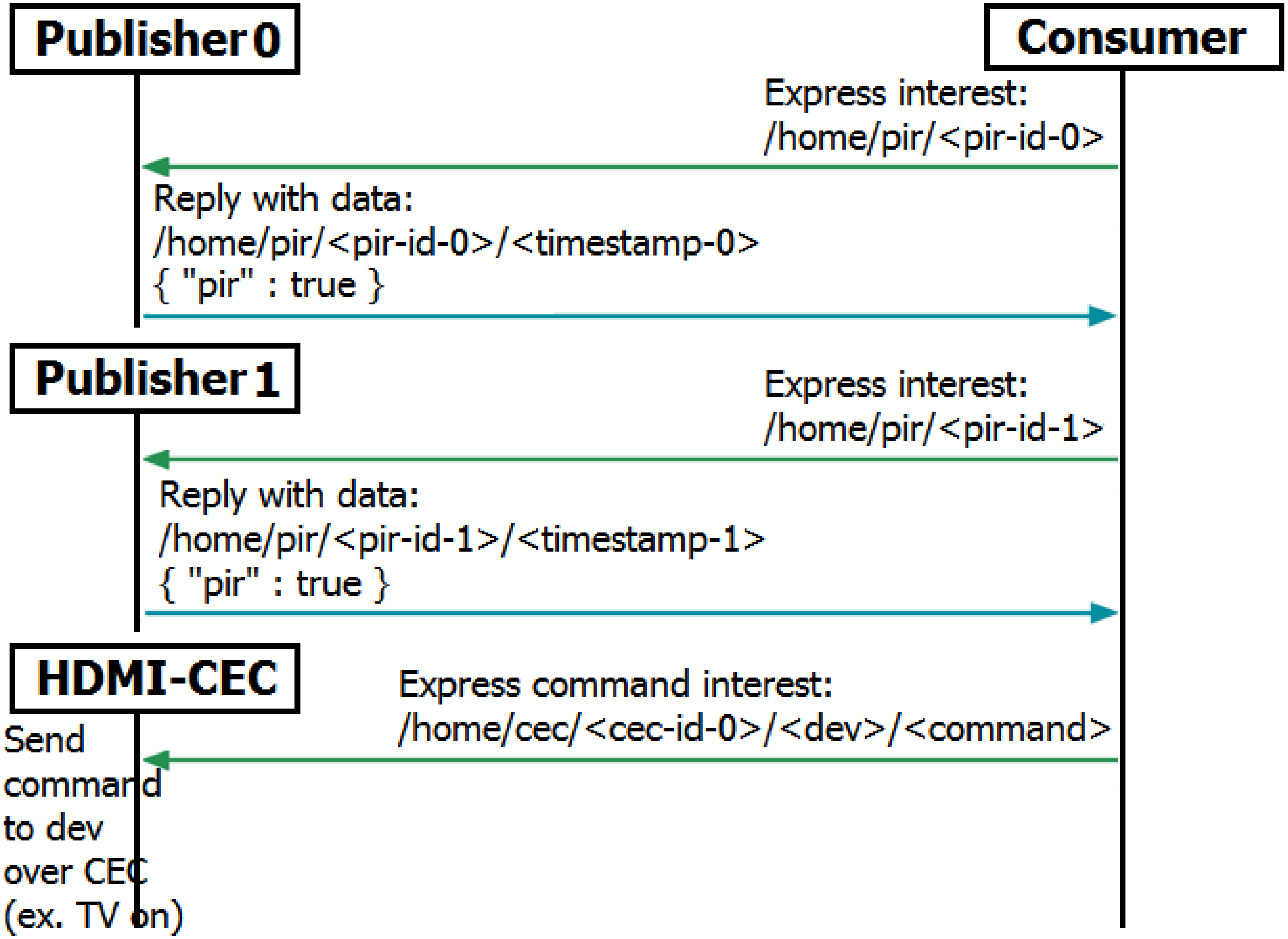
Included NDN Projects

- Libraries and Frameworks:
  - PyNDN
  - ndn-cpp
  - ndn-cxx
  - NFD & NRD
- Tools:
  - ndn-repo-ng
  - ndnsec

Included Examples

- TV control based on occupancy
  - Passive infrared sensor nodes sense occupancy
  - HDMI-CEC television fixture nodes control attached TVs
  - Application node switches TV on when room is occupied or off if it is empty
  - Namespace:
    - Root: /home
    - Infrared sensors: /home/pir/<GPIO pin>
    - HDMI CEC node: /home/cec/
    - Consumer: /home/consumer/

TV Control Network Flow



Publisher 0

Consumer

Express interest:
/home/pir/<pir-id-0>

Reply with data:
/home/pir/<pir-id-0>/<timestamp-0>
{ "pir" : true }

Publisher 1

Express interest:
/home/pir/<pir-id-1>

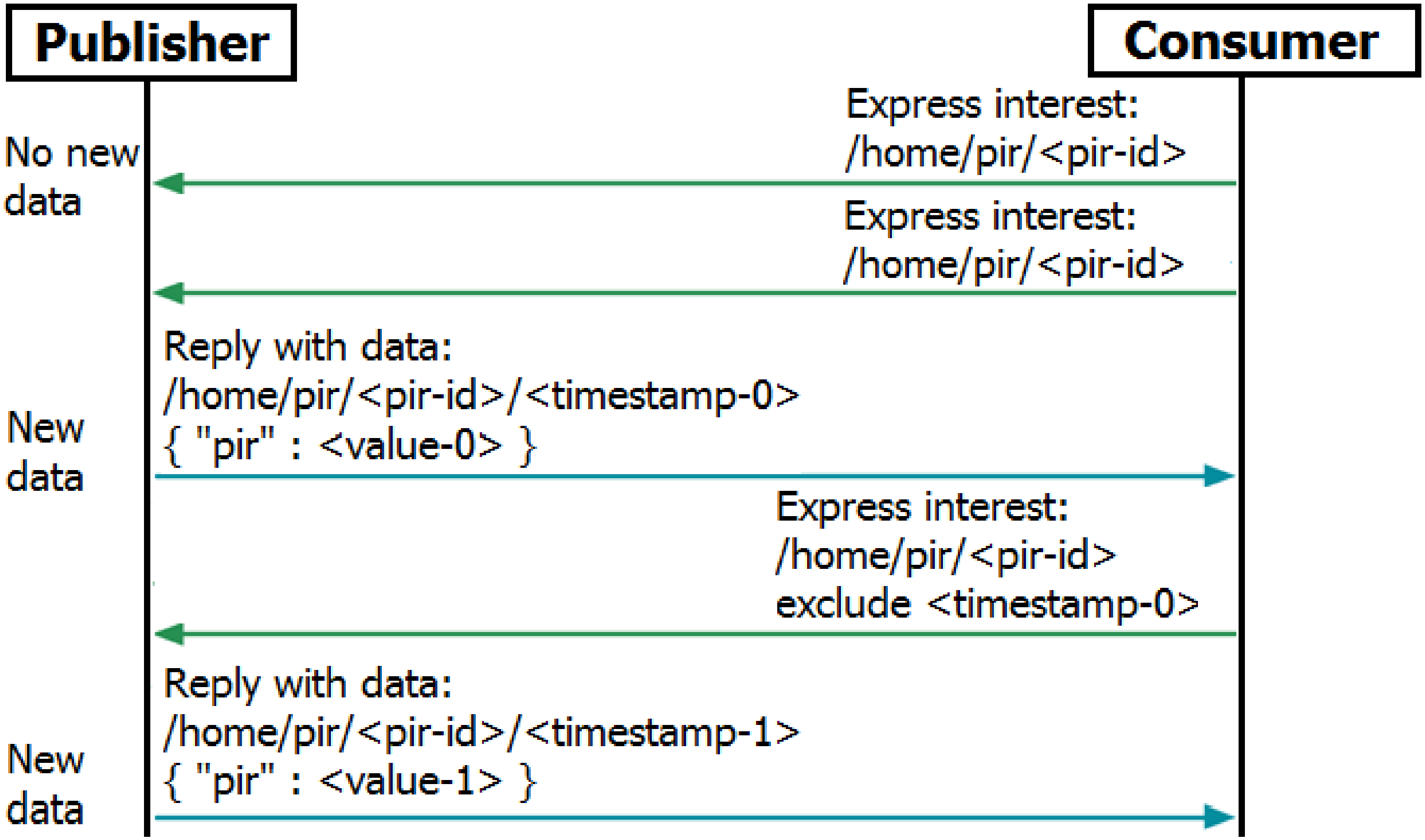
Reply with data:
/home/pir/<pir-id-1>/<timestamp-1>
{ "pir" : true }

HDMI-CEC

Express command interest:
/home/cec/<cec-id-0>/<dev>/<command>

Send command to dev over CEC (ex. TV on)

Polling Sensor



Publisher

Consumer

No new data

Express interest:
/home/pir/<pir-id>

Express interest:
/home/pir/<pir-id>

New data

Reply with data:
/home/pir/<pir-id>/<timestamp-0>
{ "pir" : <value-0> }

Express interest:
/home/pir/<pir-id>
exclude <timestamp-0>

New data

Reply with data:
/home/pir/<pir-id>/<timestamp-1>
{ "pir" : <value-1> }

Included Examples

- LED lights under user controller
  - LED nodes control LEDs attached to GPIO pins
  - Application node takes user input and issues commands to LED nodes
  - Namespace:
    - Root: /home
    - Single LED node: /home/led/
    - Multiple LED node: /home/led-multi/<pin number>
    - Application: /home/viewer/

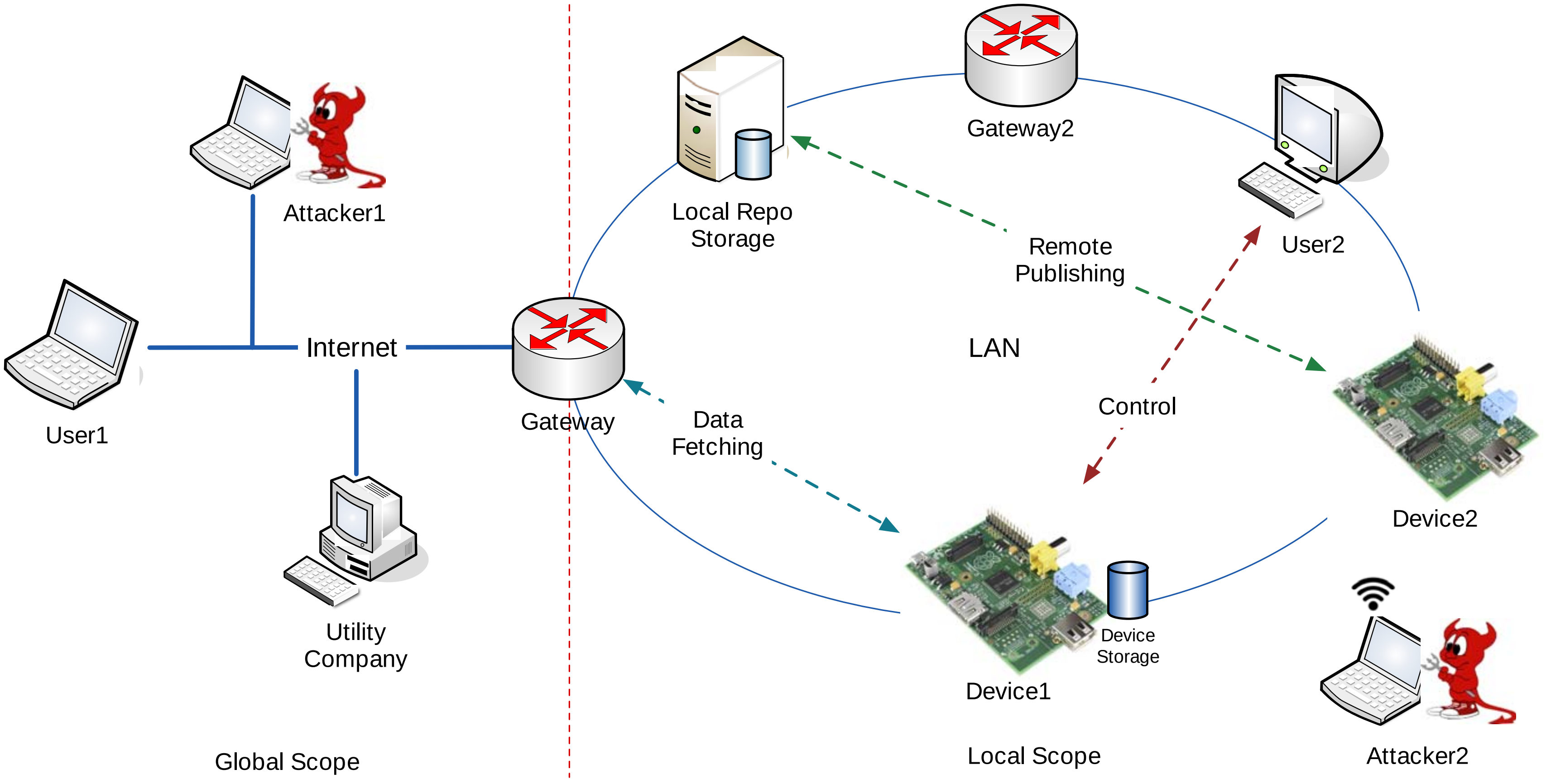
Included Examples

- Content cache
  - Publisher node measures CPU and memory usages, number of users and uptime
  - Publisher node can publish multiple prefixes
  - No application provided – users can use console class to request and inspect data
  - Namespace:
    - Root: /home
    - Publisher: /home/repoman/
    - Publisher prefix list: /home/repoman/listAvailablePrefixes

Current Examples

- Bus stop bench sculpture
  - 1 sensor node – publishing next bus information
  - 1 fixture node – controlling colors on light strip
  - 1 controller node – maintains certificates, lists devices, issues commands to light fixutre based on next bus ETA and occupancy
  - Namespace:
    - Root: /ndn/ucla.edu/sculptures/ai-bus
    - Lights: /ndn/ucla.edu/sculptures/ai-bus/lights
    - Controller: /ndn/ucla.edu/sculptures/ai-bus/controller
    - Publisher: /ndn/ucla.edu/apps/transportation/bus

Deployment Scenario



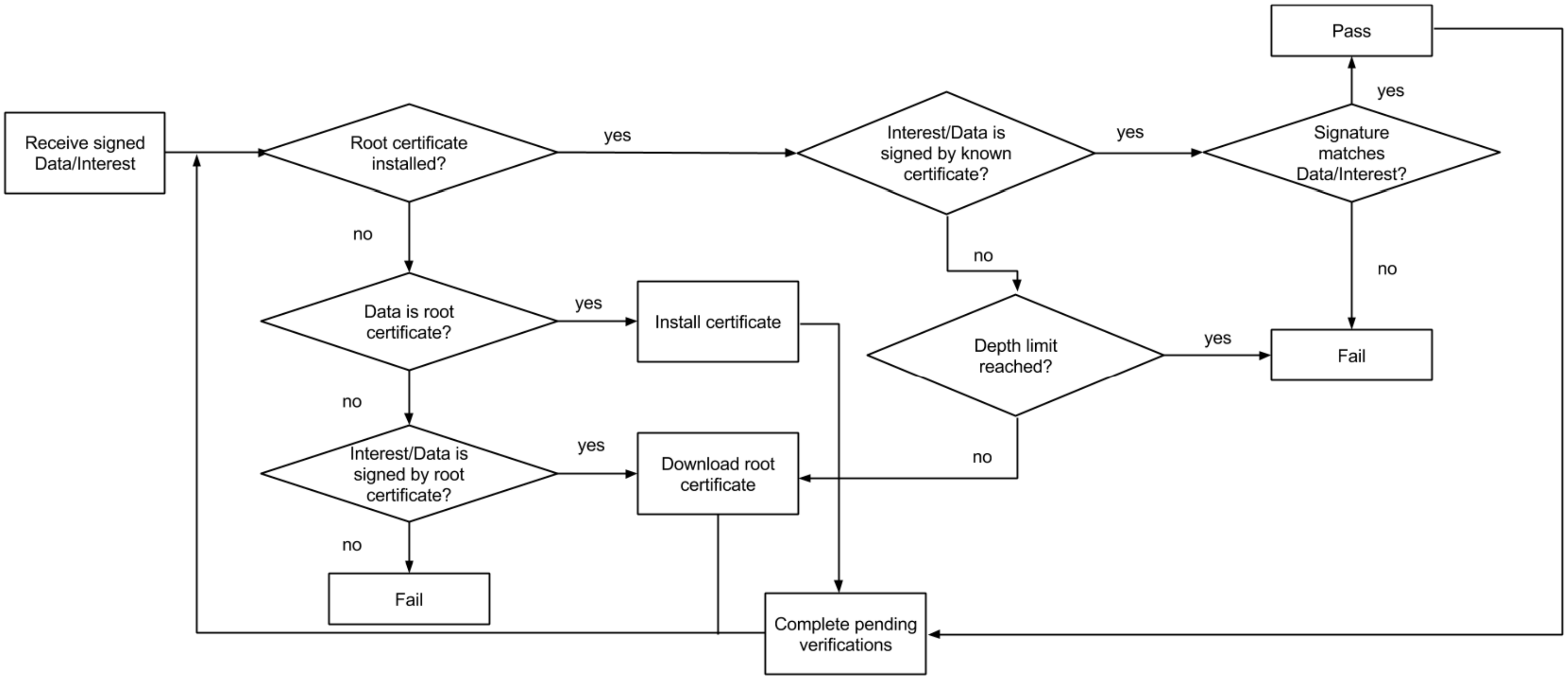
Security/Trust Model

- Currently, devices must be set up with their namespace as well as the name of the master node (gateway)
- Before they can issue or respond to interests, devices must send a certificate signing request to master node
- Example namespace: /home/fred/
  - Master node name: /home/fred/controller
  - Device name: /home/fred/bedroom/light1

Security/Trust Model

- Each command interest or data packet includes the network name of the certificate used to sign it
- In order to be valid, the certificate:
  - Must have a name within the home network, e.g. /home/fred/KEY/bedroom/light1/ksk-3838/ID-CERT
  - Must be itself signed by the master node or another node in the home network
  - If not signed by the master node, the certificate chain must lead to the master node in a small number of steps

Trust Verification Flow


Receive signed Data/Interest
Root certificate installed?
yes
no
Interest/Data is signed by known certificate?
yes
Signature matches Data/Interest?
yes
Pass
no
Data is root certificate?
yes
Install certificate
no
Interest/Data is signed by root certificate?
yes
Download root certificate
no
Fail
no
Depth limit reached?
yes
Fail
no
Complete pending verifications

Available IoT Frameworks

Name	NDN RPi Toolkit	Nest API	Thread	Apple Homekit	AllJoyn
Device Discovery	✔	✔	✔	✔	✔
Bootstrapping	✔*	✗	✔	✔	✔
Security	✔	✔	✗	✗	✗
Caching	✔	✗	✔	✔	✔
Low Power	?	✔	✔	✔	✔
Free	✔	✗	✔	✗	✔
Supported Languages	Python, C++, JS	JS	?	Obj-C, C++	C++, Java, C#, JS, Obj-C

✔	Yes/Included
✗	No/Absent
?	Unknown
✔*	In development

Get the Source

- Source is available at

  https://github.com/remap/ndn-pi