NDNoT: design and implementation of an IoT framework over Named Data Networking

Zhehao Wang
University of California, Los Angeles
zhehao@cs.ucla.edu

Jeff Burke
University of California, Los Angeles
jburke@remap.ucla.edu

ABSTRACT

Named Data Networking of Things (NDNoT) framework is a set of libraries that aim to support application development in a home IoT environment, with Figure (setup) being an example.

The design of the framework follows the guidelines proposed in (cite:IoTDI15), specifically providing functionalities that name home devices and their data, discover devices and services, bootstrap devices and define the trust model, and support application level publishing/subscribing. The rest of the abstract introduces Named Data Networking, then describes the design of NDNoT and its considerations for global Internet reachability and constrained devices, and concludes with the description of NDN-Flow, an application developed using this framework.

Named Data Networking (NDN) is a proposed future Internet architecture that shifts the network communication model from host-centric to data-centric. Instead of sending packets between source and destination devices identified by IP addresses, NDN disseminates named data at the network layer, and forwards directly using hierarchical and application-meaningful names. Moreover, NDN adopts content-based security and secures data at the time of its generation.

Naming

In NDNoT, each piece of data, device, and application are named. Namespace design is a priority because names are strongly tied with application semantic, the pattern of data retrieval and trust model definition. In a home IoT environment, applications usually refer to things and their data using application-meaningful names, which could be different from the names of devices under the home context. For example, */MyHome/devices/wii-controller/2* could name a wii game controller device, which provides user input for a game application *game1* in my home, under the name prefix */MyHome/applications/game1/inputs/a*. Separating these two namespaces reflects NDN's concept of naming the data the way the application wants, rather than naming the device that produces it. This is helpful in scenarios such as device replacement, where the replaced and replacement serve the same purpose for the application, and application data retrieval won't be hindered because of the replacement.

In NDNoT, we also assume that each device comes with a manufaturer-configured identity, which is another name under the manufacturer prefix, for example, */Company/Nintendo/wii-controller/serial-1234*. This name serves as a prefix to initialize bootstrapping process for the device, meanwhile it corresponds with a certificate, a piece of named data that allows the user to verify if the device comes from the manufacturer, before introducing it to the home and naming it accordingly.

To summarize, the namespaces are given in Figure (namespace). Suffixes such as "_meta" denotes the metadata related with a device, for example its current status or manufacturer profile. The implementation of NDNoT provides an interface for the user to name their devices, as well as an interface for each device to publish under certain application namespaces.

Bootstrap and trust model

Application data authenticity is a primary concern in NDNoT. In NDN, trust decisions can leverage the structure of names to schematize decision-making on a packet-by-packet basis that does not require channel- or session-based semantics cite: schematized trust.

To provide authenticity NDNoT follows a hierarchical trust model, in which a home gateway is the trust anchor, and devices should be authorized by the home gateway, or intermediate gateways that are authorized by the home gateway. The authorization process, or device bootstrapping, involves having the gateway sign the certificate of the added device, and installing the certificate of the gateway on the added device. Communications in this process are authenticated by a shared secret, which is manually given to both the gateway and the added device (cite: ndn-pi TR).

For a consumer application to be able to verify application data it receives, a trust schema, which defines what the expected relationship between data name and the signing certificate name is, is distributed by the gateway. For each type of application data that a device intends to publish, the device asks for permission from the gateway, who keeps a trust schema for each application running in the home environment. Thus upon receiving application data, the consumer would be able to consult the schema and verify it's signed by an expected device, and if that device's certificate can be traced back to the trust anchor.

The trust relationship is summarized in Figure (trust-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

relationship).

Data confidentiality as a separate concern, could be tackled by name-based access control (cite:nbac-tr).

Device and service discovery

Device discovery in NDNoT is achieved using synchronization (sync, cite:let's chronosync), which is a multi-party communication paradigm that efficiently reconciles collections of named data. Each device, upon being added to the home network, will send out an interest under a broadcast devices namespace carrying an initial digest, in order to learn the names of devices in the home from the data response. Afterwards, the new device adds its own name to the set, and announces a new digest so that receiving devices know to fetch the new device's name, and after learning the name fetch the services it provides by appending a "_meta" component to the interest name.

Application-level pub-sub

Publish-subscribe (pub-sub) is a common communication paradigm for IoT applications. NDN follows a pull-based paradigm: data is delivered based on interests that ask for it, and at network layer persistent subscription with publisher-initiated communication is not supported. NDNoT provides application-layer pub-sub functionality by having the subscriber "refresh" its interest for publisher's data. For example, for a publisher whose data is named with incremental sequence numbers, the consumer can keep issueing interest for the next sequence number it expects to receive, and reissue this interest if it times out.

Global reachability

While the discussion above focuses on a local context, NDNoT can support the incorporation of the local system with the global Internet in multiple ways. One would be defining a "globally reachable" home prefix, such as */edu/ucla/mainhall/room122*, and name things, devices and their data accordingly. Alternatively, forwarding hints and encapsulation, discussed in (cite:SNAMP), can be utilized to provide global Internet incorporation.

Constrained devices

Securing communication for constrained devices arises as a challenge, since they may not have enough computational power or storage to support public key cryptography. To tackle this constraint, NDNoT provides an example which introduces "helpers", a more powerful device, for constrained devices. Constained device shares a secret with its helper by encrypting it with the helper's public key, and uses this shared secret to generate HMAC signatures verifiable by the helper. Helper would keep identity names and key pairs for the devices it helps, and packetizes their messages as NDN data packets, and expose them to the network, so that consumers of this data don't need to differentiate constrained devices from others.

NDN-Flow application

NDNoT framework provides implementation in C++, Python, JavaScript and C#. An application, NDN-Flow, is developed using the framework. NDN-Flow is a home entertainment experience built on top of NDN, in which the player navigates a virtual space by interacting with a webpage on their mobile phones, gyroscopes attached to RFduinos, and have their movement tracked by OpenPTrack (footnote:website), a multi-camera person tracking system running on a PC. In our installation two Raspberry Pis serve as the home gateway, and helper for RFduinos.