Figure 1: TODO: A depiction of the assumed discovery process for devices in the network.

1 Introduction

2 Oceanography Scenario Description and Motivation

Our protocol is peer-to-peer, where peer-to-peer here means communication between a pair of nodes with a direct physical link (acoustic, RF, fibre, what have you).

2.1 Data Naming Conventions

The data naming convention in the oceanography network is designed to be intuitive, hierarchically structured, and adaptable. It is based on a URI-like format with query parameters, ensuring both readability and precision.

The convention follows a path-query format. The path typically contains the broad location and data type (e.g., “/pacific-ocean/temperature"), while the query parameters provide specific details (e.g., “near_latitude=53.34"). This dual-layer structure ensures both a high-level categorization and fine-grained control. In this format data names are allowed to be somewhat ambiguous such that multiple different specific data payloads might satisfy a single data name.

Location Identifier: Specifies the oceanic region (e.g., Indian Ocean, Atlantic Ocean).

Data Type: Indicates the type of data (e.g., temperature, pH level).

Query Parameters: Include specifics like geographic coordinates, depth, time constraints, and tolerances.

An example of a valid data name in our scenario is:

‘‘`/atlantic-ocean/water-ph?near_latitude=53.34&near_longitude=68.34&near_depth=30`’’.

3 ICN/NDN Protocol

Here we describe our ICN/NDN protocol which we name G17ICN. We designate this as version 1.0.

3.1 Concise Description of the Protocol

The protocol uses HTTP as the serialization and request/response format, with a JWT as the request body. There are two types of ICN packets in our protocol, interest and satisfy packets. Interest and satisfy packets are encoded as JWTs and delivered in the body of a HTTP POST request.

The JWTs are signed using PKC, the default algorithm in our emulation is RSA 256 with key size 512, but this is configurable. Both interest and satisfy packets are signed.

Timestamps in the protocol in are seconds since the epoch, either integer or decimal.

3.1.1 Routing

Each device acts as a router to facilitate communication between other devices. In designing a routing algorithm there are several things we wish to optimize/minimize.

1. **Packet Latency:** Minimizing the time it takes for a packet to travel from the source to the destination.
2. **Round Trip Latency:** Minimizing the it takes to receive an appropriate satisfy packet after issuing an interest packet.
3. **Bandwidth Utilization:** Maximizing the effective use of the available bandwidth in the network.
4. **Packet Loss:** Minimizing the number of packets that are lost during transmission.
5. **Throughput:** Maximizing the number of successful message deliveries over a communication channel.
6. **Routing Overhead:** Minimizing the amount of additional network traffic caused by routing protocols.
7. **Scalability:** Ensuring the routing algorithm performs well as the network size grows.
8. **Energy Efficiency:** Minimizing energy consumption, particularly in wireless and mobile networks.
9. **Security:** Ensuring that the routing protocol is secure against various types of attacks.

while satisfying all requests with min Satisfy packets are sent lazily, i.e. only after receiving a corresponding interest packet.

```
{
        "type": "interest",
        "created_at": "1700471392",
        "data_name": "/temperature?after=170047332&near_latitude=53.82&near_longitude=4.46&
near_depth=100m"
}
```

Figure 2: An example of the JSON data for an interest packet that will be encoded as a JWT.

3.1.2 Optimisation Headers

Our protocol uses HTTP headers to provide hints to neighbouring nodes for routing optimisation. The 'x-g17icn-version' will facilitate compatibility with future version of the protocol. The 'x-g17icn-hopcount' header indicates the distance in hops that a packet has traveled from its origin. A series of headers 'x-g17icn-router-0', 'x-g17icn-router-1', etc. allow each router to reconstruct the traceroute of the packet so far. This can also be used for a router to gleem an approximate picture of the network topology. The 'x-g17icn-router-X' headers contain three pieces of information; 1. the time at which the router first received the packet, 2. the key name of the router, 3. a numeric indicator of the router's congestion at that time.

3.2 Motivation and Design Considerations

HTTP is a lightweight textual format for separating out message headers and message content. It has ubiquotous library support in almost all environments. We implemented our emulation of this protocol in python using the 'asyncio' asynchronous runtime, which entailed running multiple http servers in a single process *and* having those servers communicate with other. The existence of HTTP libraries designed to run in the 'asyncio' environment facilitated rapid prototyping of our emulator. Running multiple servers in a single asynchronous process can be tricky in terms of avoiding deadlocks, but the libraries we used, 'Quart' for servers, and 'httpx' for clients, could communicate without deadlocks.

3.3 Security

RSA 256 is used for signing (and optionally encrypting) data packets. Each device has a public/private key pair. The name of a device is the SHA 256 hash of that device's public key, which we call the "key_name". All interest and satisfy packets include the data name of the creator of the packet and are signed by the creator of the packet using the private key such that any node can verify the authenticity of a packet. The node uses it's private key to create a cryptographic signature of the interest/satisfy packet, and encodes the packet and signature as a JWT.

While this protocol provides message authenticity it does not establish a trust model. For version 1.0 of G17ICN we assume that a list of trusted devices (mapping key names to public keys) has been distributed out-of-band. Each device holds the list of trusted keys and discards any message not signed by a device from this list.

4 Emulation

We have implementented an *emulation* of our protocol, not a deployable implementation of the protocol, for the reason that an emulation facilitates more thorough evaluation and debugging. In the scenario we have described it is assumed that devices would communicate with each other primarily over acoustic links but we have not emulated characteristics of the physical layer. Rather, our focus has been on emulating diverse and dynamic network topologies.

In the emulation each device finds for itself a unique IP interface (host/port pair).