Proposed Design Choices for IoT over ICN

draft-lindgren-icnrg-designchoices-00

presented by Olov Schelén

IRTF ICNRG

@ IETF 94, Yokohama
Objectives

• To support efficient and scalable IoT over existing ICN proposals with small changes to the ICN concepts

• To provide some design proposals (with trade-offs) to allow for effective, efficient, scalable and secure handling of IoT data in an ICN network
Design choices for IoT
Summary (then following slides add details on naming)

• Immutable atomic data units
  – Eliminating cache inconsistencies
  – Consequently dynamic data is modeled as a stream of immutable data units
  – Old and new values can be separated by name
• Naming
  – Sequence number as optional part of name is beneficial
  – Time is important for sensor data. A method to map actual time to a sequence number is needed
  – Since new names are created frequently, deducible names are beneficial, avoiding advanced name “lookup”
• Capability / data property advertisements
  – Tell how data is structured, named, and published (e.g. valid points in time)
  – Advertisements can be disseminated with ICN (low frequency expected)
• Object security
  – Handled by the IoT framework, no need to modify typical ICN standards
Naming in Streams of immutable data

• Sensor reading names include sequence number

• How do clients find the “current/latest” seq nr?
  – Can be guided by advertised capabilities
  – Can use metadata (from request of wildcard seq nr)
  – Can make use of subscription service, or long-lived (pending) requests/interests
    • if available – requires state in the network!
    • works also for upcoming real-time data!
  – Can probe the sequence space (next slide)
Probing for the latest value

• Need a starting sequence number:
  • sensor capability advertisement, application knowledge, request without sequence number

• Search (binary) or probe the sequence number space from the starting point
  – Assuming the producer will provide a negative response to non-existing data

• Challenges:
  – gaps (if common) would be a problem
  – Requests for non-existing data may build request state in the network
  – Non-existing data can suddenly start to exist (only publisher knows)
Probing and latency

• Can probe arbitrarily close to time of availability
  – provided fixed frequency
  – potential issue with requesting non-existing data
• If a request is held until the value is available
  – Such a pending request acts as a one-time subscription
  – minimises latency
• To be safe on real-time communication, ICN used in concert with existing internet protocols (at least initially)
  – Fall-back for real-time actuation/control of specific IoT devices w/o ICN
Importance of time

• Time is almost always important for IoT data
  – Should be possible to request data from a particular time

• Approach:
  – Use sequence numbers as part of the name
  – Let the application provide mapping between sequence numbers and time
    • Can be part of capability advertisement, metadata, or part of the actual IoT data
  – No need to extend the ICN service
  – Works well for periodic data

• Challenge for receiving triggered data, alarms and other spurious data
  – Basically the same problem as for actuation (next slide)
Handling actuators: 2 methods

• Method 1: Actuator state is represented by stream of immutable named data objects
  – Actuator regularly has to request its new state
  – Someone has to publish that state
  – Clean mapping to ICN's object and interaction model
  – Trade off between actuation latency and efficiency (seldom & urgent)

• Method 2: Actuation invoked as a side-effect of receiving a particular request
  – State might be encoded in the name requested or supplied as additional information
  – Actuator might respond with its state in the data returned
  – Issues with security (authenticity and access control)

• Models could be combined
  – Request (model 2) that triggers the actuator to request its new state (model 1)
ccn-lite implementation of CCNx 1.0

• Bare bones implementation
  – Type Length Value (TLV) packet format
  – CCNx 1.0 forwarder
  – Exact matching of interest and content names
  – InterestReturn message implemented as NACK
Metrics

• Number of Interest messages sent
• Number of Content Objects messages sent
• Average Content Store (CS) occupancy
• Maximum/Average Pending Interest Table (PIT) entries
• Latency/Hop count
Evaluation Framework
ccn-lite on Docker containers

• Docker containers provide network isolation between ccn-lite-relays and allow network level emulation
• Static mesh topology of ccn-lite-relays
• Randomly selected content sources and consumers
• Content published as a stream of immutable objects with increasing sequence numbers
• Periodic and triggered content publishing models
• Logs to evaluate metrics