Summer Dev Plan for EBAMS

Draft 1 – June 24, 2015

Intent of this Presentation

- **Want to write out redmine tasks for collaborative development by June 29 meeting. What needs to be answered before that?**
- **Want to have an alpha version running by end of August. Is that reasonable?**
- Today:
  - Recap Wentao's proposal for streaming computation of aggregates / hierarchical storage.
  - Discuss a "virtualization" of the EBAMS testbed.
  - Figure out how to address unique concerns of EBAMS?
  - Develop plan and responsibilities for proceeding.

Hierarchical storage for BMS

Wentao Shang

Notes in red by JeffB

Recall: BMS namespace

- Root
  - Building (Melnitz, …)
    - Room (studio1, …)
      - Device gateway (panelAA, …)
        - Data type (demand, voltage, current, …)
          - Aggregation (instant, avg, …)
            - Timestamp
- Note that some information is not encoded in the [primary] namespace (e.g., department), though it may be presence in the user namespace for example, or in secondary data namespaces.

Proposed Stream data naming convention

- Raw data
  - /<data-prefix>/RAW/[timestamp]
  - E.g., /UCLA/Bolter/4806/Panel1/Voltage/RAW/7
- Batched raw data
  - /<data-prefix>/RAW/[tstart]/[tstop]
  - E.g., /UCLA/Bolter/4806/Panel1/Voltage/RAW/0/10
- Aggregated data
  - /<data-prefix>/[aggregation]/[tstart]/[tstop]
  - E.g., /UCLA/Bolter/4806/PAnel1/Voltage/AVG/0/10

BMS data storage structure

- Objective: Design a hierarchical storage approach and a stream-based approach to calculating aggregates, distributing processing and taking advantage of local storage.
- Multi-level storage hierarchy
  - Structure defined by BMS namespace
  - Storage can be associated with any name prefix at any level
    - Usually have repo at gateway/room/building levels

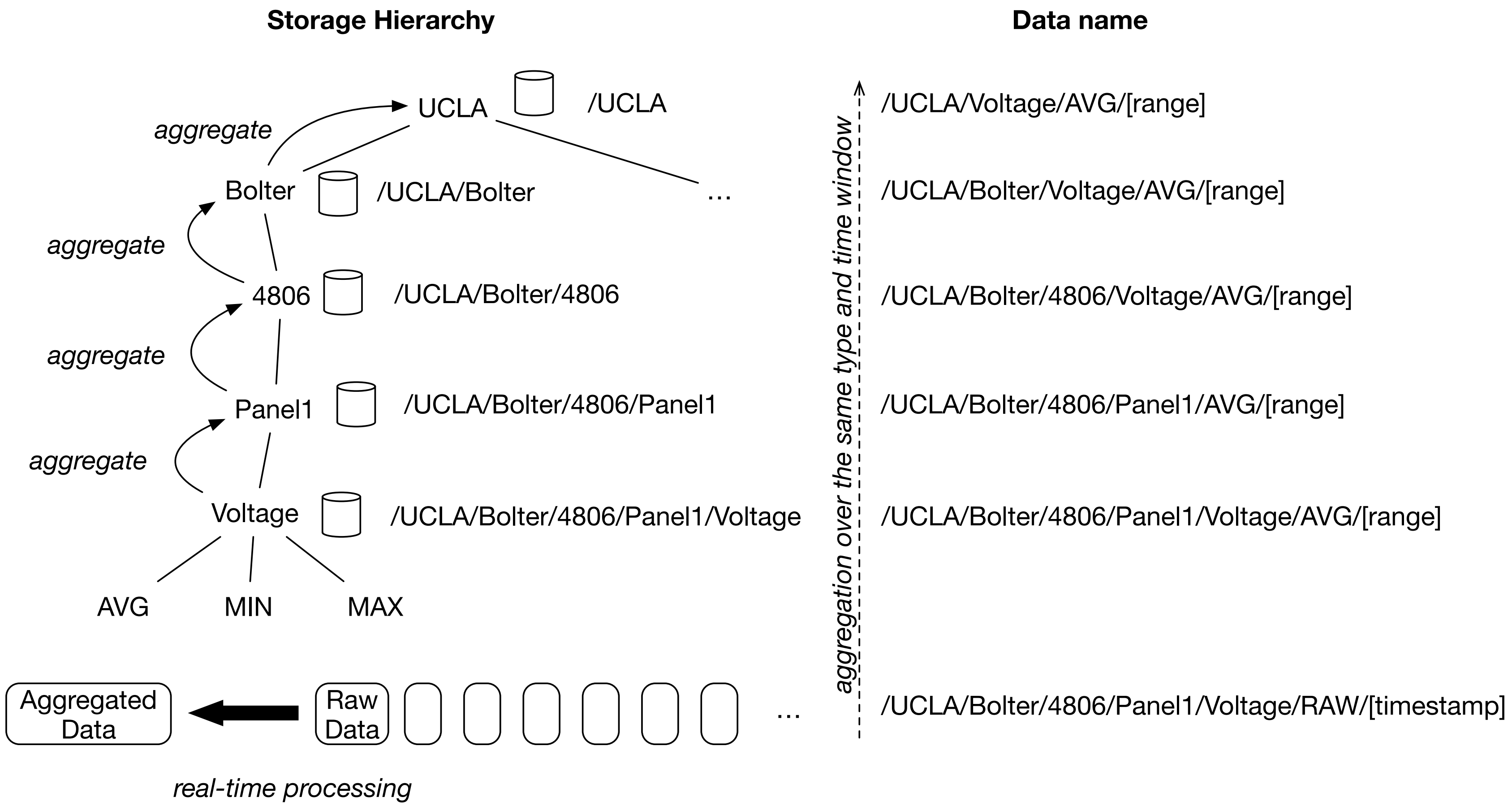
Data placement

- Choice 1: full replication
  - Each level replicates ALL data
  - Problem: data size grows exponentially as the level goes up
- Choice 2: aggregation
  - Aggregate the data (and therefore reduce data size) as we move up the hierarchy
  - This can be used in combination with local storage that keeps *all* raw samples for a limited amount of time, allowing them to be retrieved in case of a fault or alarm. So, we assume that in the aggregate case, full detail for the level in the hierarchy is kept, just not necessarily permanently.

How to aggregate

- Observation: aggregation is ~~only~~ typically most useful within the same BMS data type
- Approach: aggregate data with the same type as we move up the hierarchy
  - /UCLA/Bolter/4806/Panel1/Power/AVG
  - /UCLA/Bolter/4806/Power/AVG
  - /UCLA/Bolter/Power/AVG
  - /UCLA/Power/AVG
  - /Power/AVG

Illustrated


Storage Hierarchy
Data name
UCLA
/UCLA
aggregate
Bolter
/UCLA/Bolter
...
aggregate
4806
/UCLA/Bolter/4806
aggregate
Panel1
/UCLA/Bolter/4806/Panel1
aggregate
Voltage
/UCLA/Bolter/4806/Panel1/Voltage
AVG
MIN
MAX
Aggregated Data
Raw Data
...
real-time processing
aggregation over the same type and time window
/UCLA/Voltage/AVG/[range]
/UCLA/Bolter/Voltage/AVG/[range]
/UCLA/Bolter/4806/Voltage/AVG/[range]
/UCLA/Bolter/4806/Panel1/AVG/[range]
/UCLA/Bolter/4806/Panel1/Voltage/AVG/[range]
/UCLA/Bolter/4806/Panel1/Voltage/RAW/[timestamp]

Note: This example should use “power”, as campus-wide voltage averages are uninteresting and potentially not correct given variation in panel voltages.

Time window synchronization

- Aggregation is meaningful only when all data fall into the same time window
  - Assume the clocks are loosely synchronized
  - That is, the delta between clocks is no larger than the sampling interval
- Simplest solution: everyone computes aggregation on the fixed time window
  - Otherwise upper level storage need to keep some “recent” RAW data, which is what we try to avoid
  - Actually, upper level storage probably keeps some RAW data (at least subsampled).

How to move data across levels

- A classic NDN problem
  - Interest notification (1.5 RTT)
  - Long-lived interest (1 RTT)
    - ChronoSync essentially uses long-lived interest
- Proposed design: pair-wise sync between parent and children
  - Children's aggregated output is synced with the parent (as the input for parent's aggregation)
  - Probably want to allow sync with other parents to allow cross-tree averages?

End of
Hierarchical storage for BMS

Wentao Shang

Proposal for Summer work

- Wentao at internship.
- Proposal:
  - Create a “virtualized” EBAMS testbed with Docker instances corresponding to NDN nodes in an EBAMS network. Run these on one or more nodes connected to the NDN testbed.
  - Try Wentao’s streaming aggregation approach, and also prepare for other experimentation.
  - Perhaps four levels of hierarchy: Panel/DAQ, Building, Department (other namespace), Campus Warehouse.
  - Implement lowest level (panel/DAQ) to acquire data from existing EBAMS gateway corresponding to its data type.
  - Web-browser based dashboard allowing both retrieval of aggregates and “drill-down” into local storage by selecting names corresponding to “RAW” data.

Proposed responsibilities

- Dashboard UX and NDN-JS implementation: Dustin and Zhehao.
- Trust model, encryption-based access control design and security library implementation: Yingdi with others from IRL.
- Security function support in PyNDN and NDN-JS: JeffT
- Docker instance creation and virtualized testbed deployment: *who? How well is docker already supported, if at all?*
- Sensor gateway node, access to EBAMS data: Zhehao, JeffB
- Aggregation node design and implementation, incl local storage with time limits: *who?*