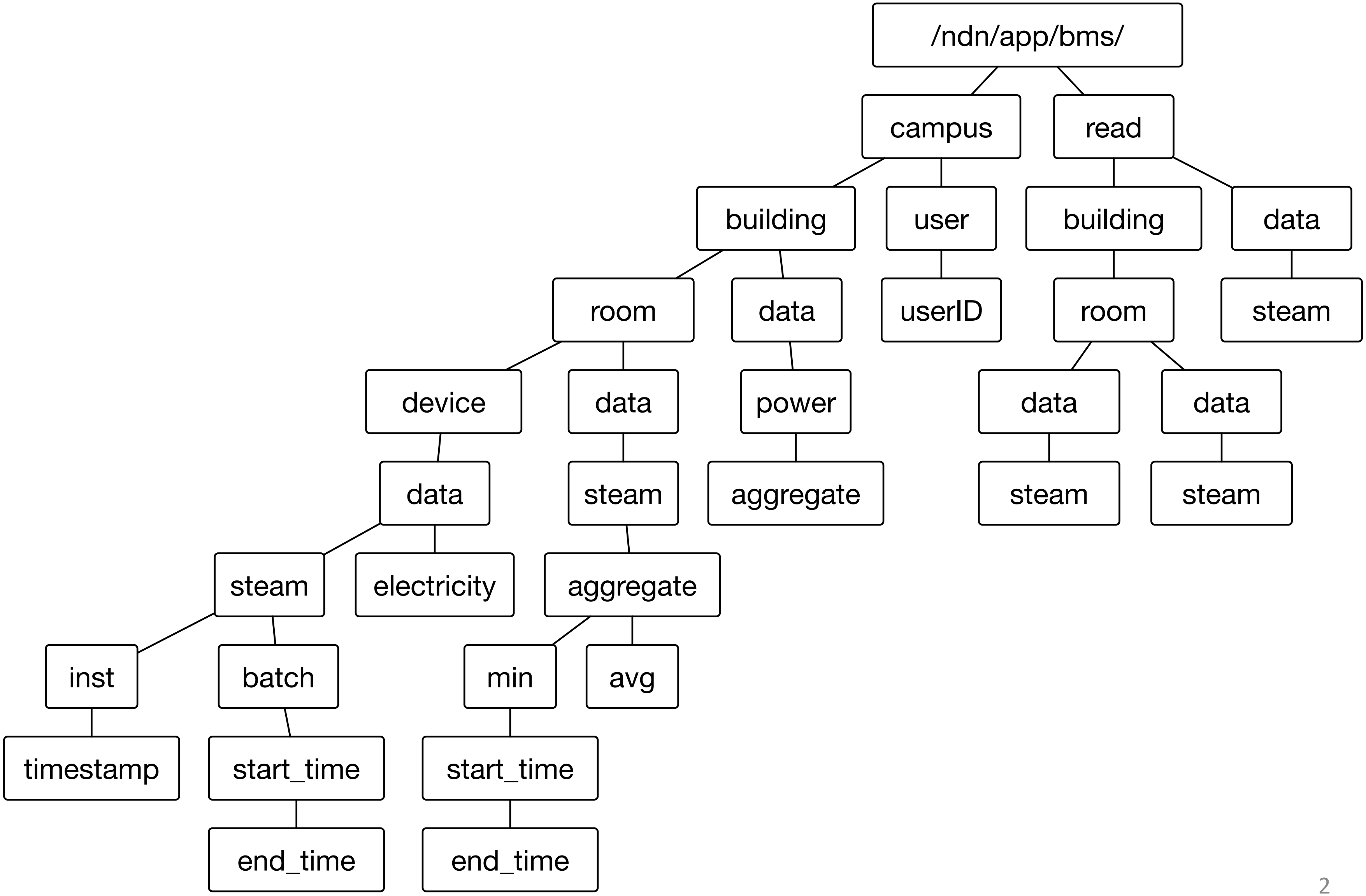
BMS Application Design

Zhehao Wang

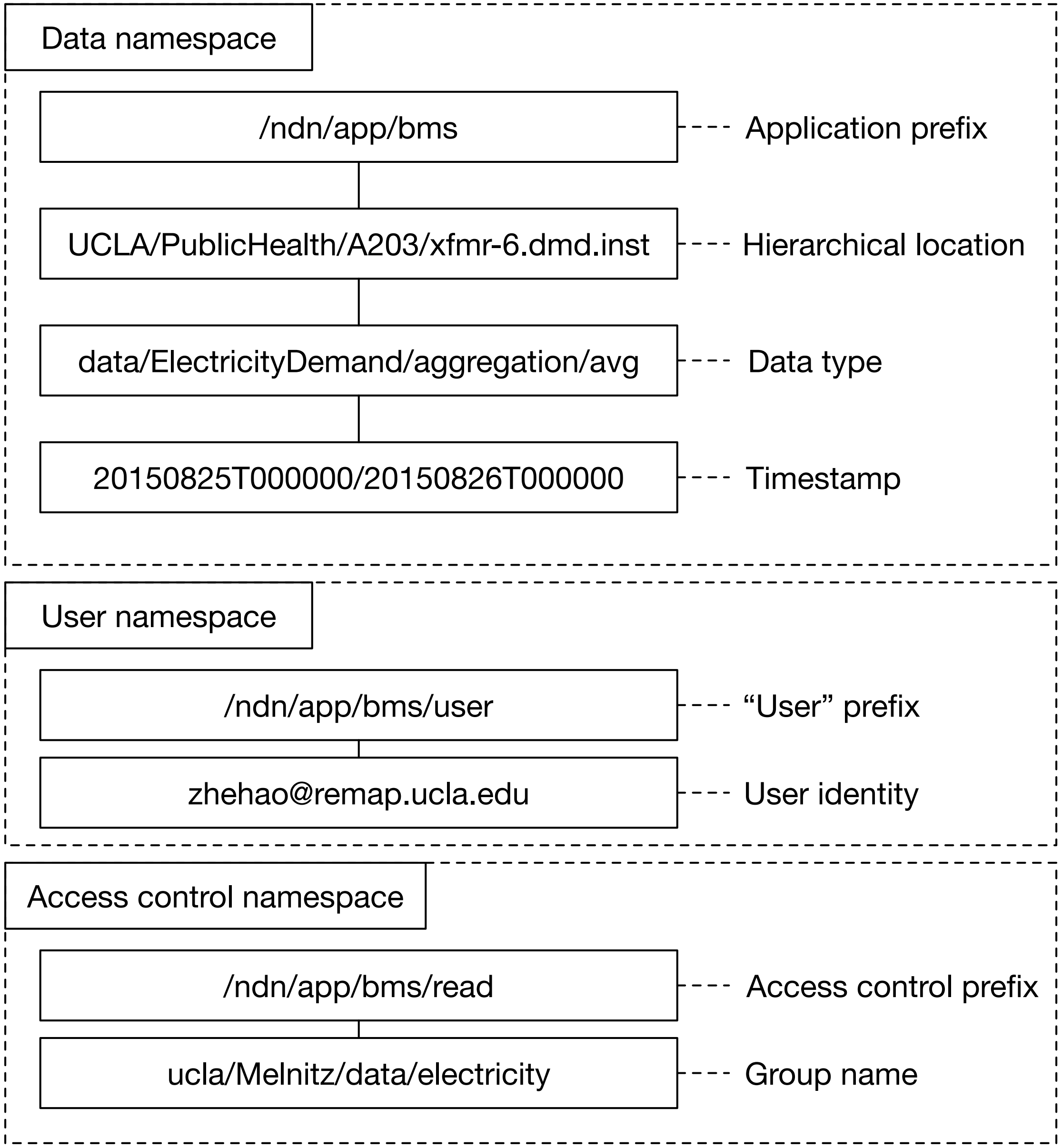
zhehao@remap.ucla.edu

Nov 8, 2015

Namespace


/ndn/app/bms/
campus
read
building
user
building
data
room
data
userID
room
steam
device
data
power
data
data
data
steam
aggregate
steam
steam
steam
electricity
aggregate
inst
batch
min
avg
timestamp
start_time
start_time
end_time
end_time

Namespace – examples



Data namespace

/ndn/app/bms --- Application prefix

UCLA/PublicHealth/A203/xfmr-6.dmd.inst --- Hierarchical location

data/ElectricityDemand/aggregation/avg --- Data type

20150825T000000/20150826T000000 --- Timestamp

User namespace

/ndn/app/bms/user --- “User” prefix

zhehao@remap.ucla.edu --- User identity

Access control namespace

/ndn/app/bms/read --- Access control prefix

ucla/Melnitz/data/electricity --- Group name

Name components

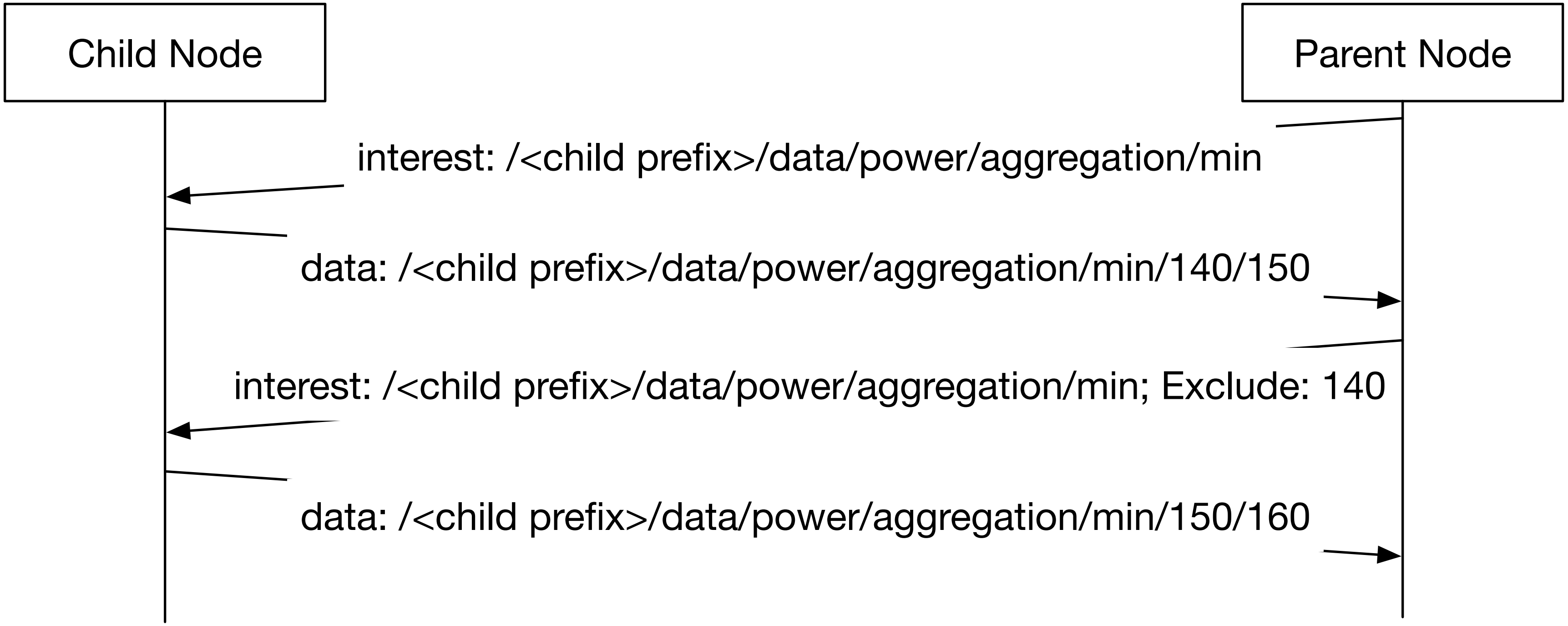
- Physical location branch contains the encrypted data gathered or aggregated at each level
  - In the example, “xfmr-6.dmd.inst” is referring to a specific sensor that publishes “electricity demand” data. The data type is reflected in the name components (in this case, “electricity – aggregated average”).
- “User” branch keeps the list of user identities
- “Read” branch keeps the list of access control groups

Hierarchical storage

- Raw data collected, kept, and batched only at the leaf nodes.
- Leaf nodes publish aggregated (min, sum, avg, etc) data at fixed time window.
- Non-leaf nodes fetch the aggregated data from all of its children, and aggregate the data after all children respond.
- Non-leaf nodes can publish aggregates with the same time window T; or n * T, n = 1, 2, 3...

Moving data across levels

- Long lived interests for fixed (configured) time window



Child Node

Parent Node

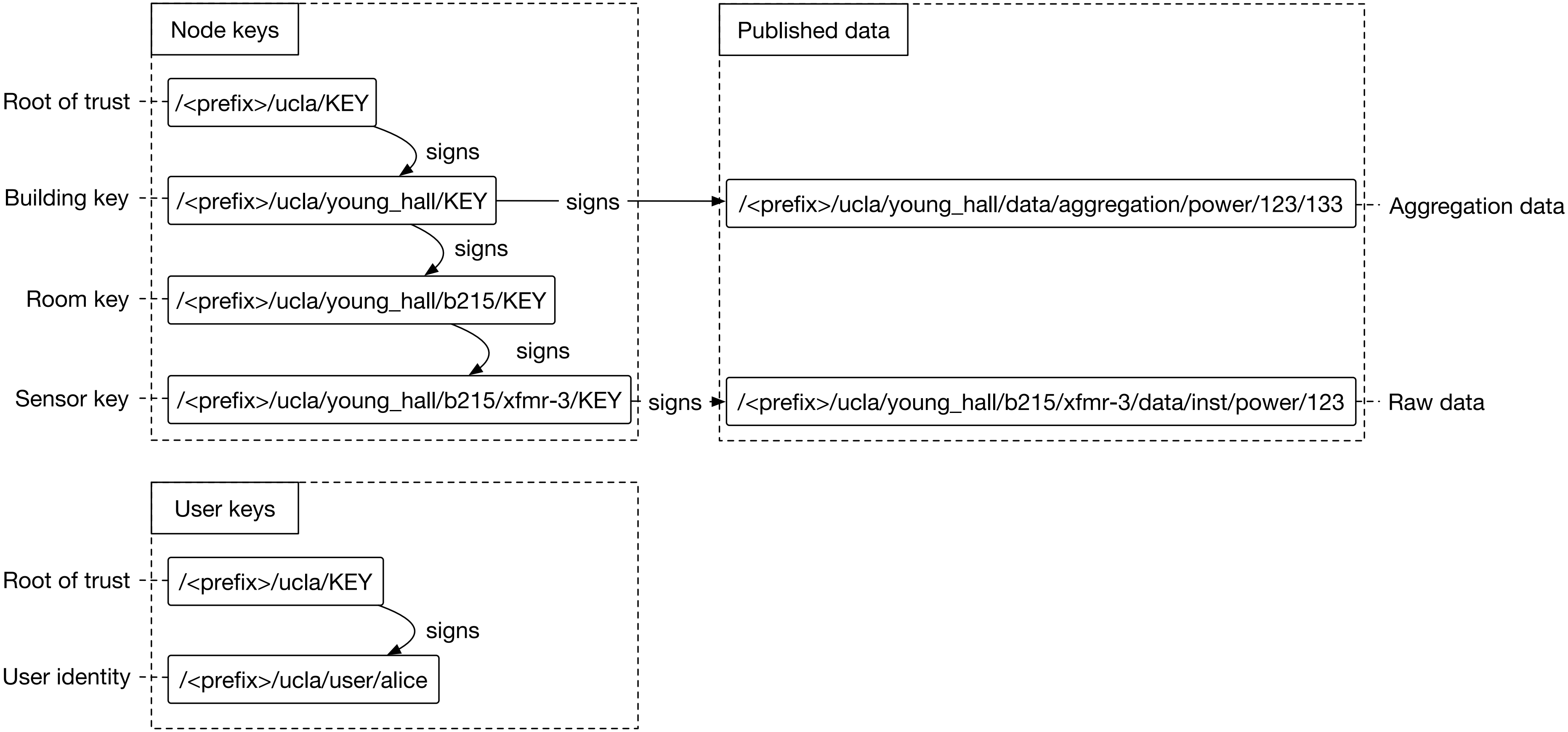
interest: /<child prefix>/data/power/aggregation/min

data: /<child prefix>/data/power/aggregation/min/140/150

interest: /<child prefix>/data/power/aggregation/min; Exclude: 140

data: /<child prefix>/data/power/aggregation/min/150/160

Signing/Verification



Node keys

Root of trust

/<prefix>/ucla/KEY

signs

Building key

/<prefix>/ucla/young_hall/KEY

signs

signs

Room key

/<prefix>/ucla/young_hall/b215/KEY

signs

Sensor key

/<prefix>/ucla/young_hall/b215/xfmr-3/KEY

signs

Published data

/<prefix>/ucla/young_hall/data/aggregation/power/123/133

Aggregation data

/<prefix>/ucla/young_hall/b215/xfmr-3/data/inst/power/123

Raw data

User keys

Root of trust

/<prefix>/ucla/KEY

signs

User identity

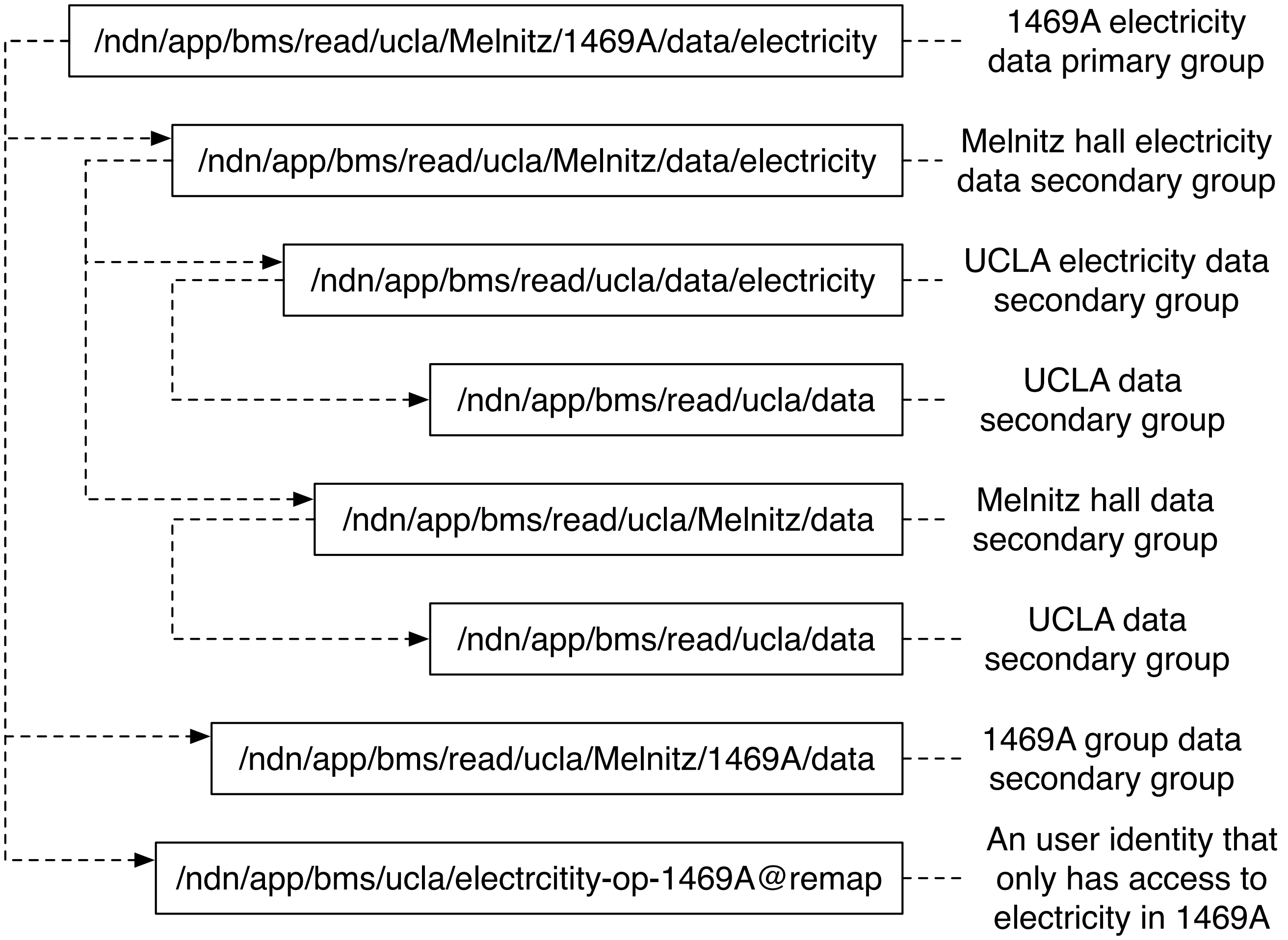
/<prefix>/ucla/user/alice

Group based access control

- Example: access control at room and data type level
- Each producer at room level or higher has a group for every type of data that it produces. Example group names:
  - "/ndn/app/bms/ucla/melnitz/1469A/data/": users in this group have access to all the data in Melnitz 1469A
  - "/ndn/app/bms/ucla/melnitz/data/electricity": users in this group have access to Melnitz Hall's electricity data

Primary and secondary groups

- Primary/secondary group relationship for “electricity data in Melnitz 1469A”



/ndn/app/bms/read/ucla/Melnitz/1469A/data/electricity	1469A electricity data primary group
/ndn/app/bms/read/ucla/Melnitz/data/electricity	Melnitz hall electricity data secondary group
/ndn/app/bms/read/ucla/data/electricity	UCLA electricity data secondary group
/ndn/app/bms/read/ucla/data	UCLA data secondary group
/ndn/app/bms/read/ucla/Melnitz/data	Melnitz hall data secondary group
/ndn/app/bms/read/ucla/data	UCLA data secondary group
/ndn/app/bms/read/ucla/Melnitz/1469A/data	1469A group data secondary group
/ndn/app/bms/ucla/electrcitity-op-1469A@remap	An user identity that only has access to electricity in 1469A

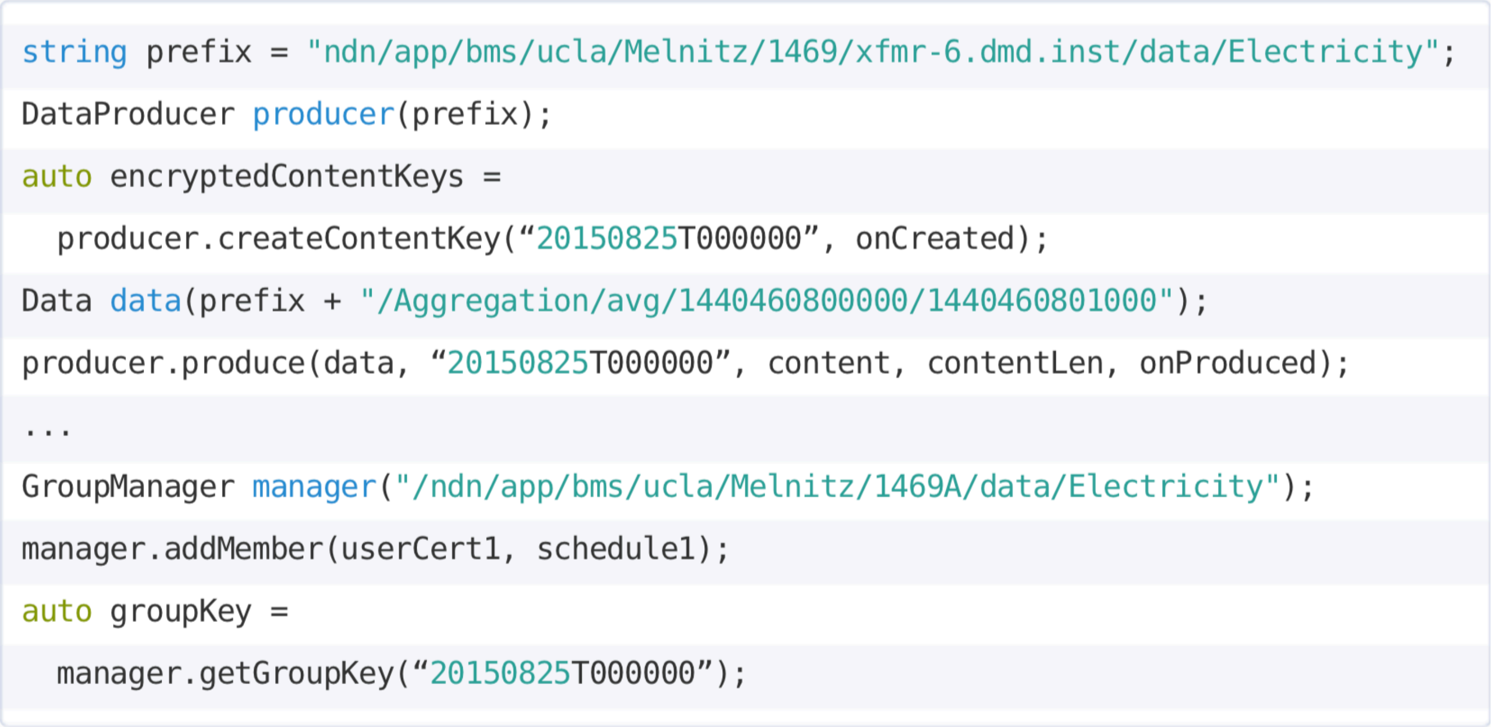
Access control – example API calls

- For example, content name

“/ndn/app/bms/ucla/Melnitz/1469A/xfmr-6.dmd.inst/data/Electricity/Aggregation/avg/1440460800000/1440460801000”

- Primary group name

“/ndn/app/bms/ucla/Melnitz/1469A/data/Electricity”

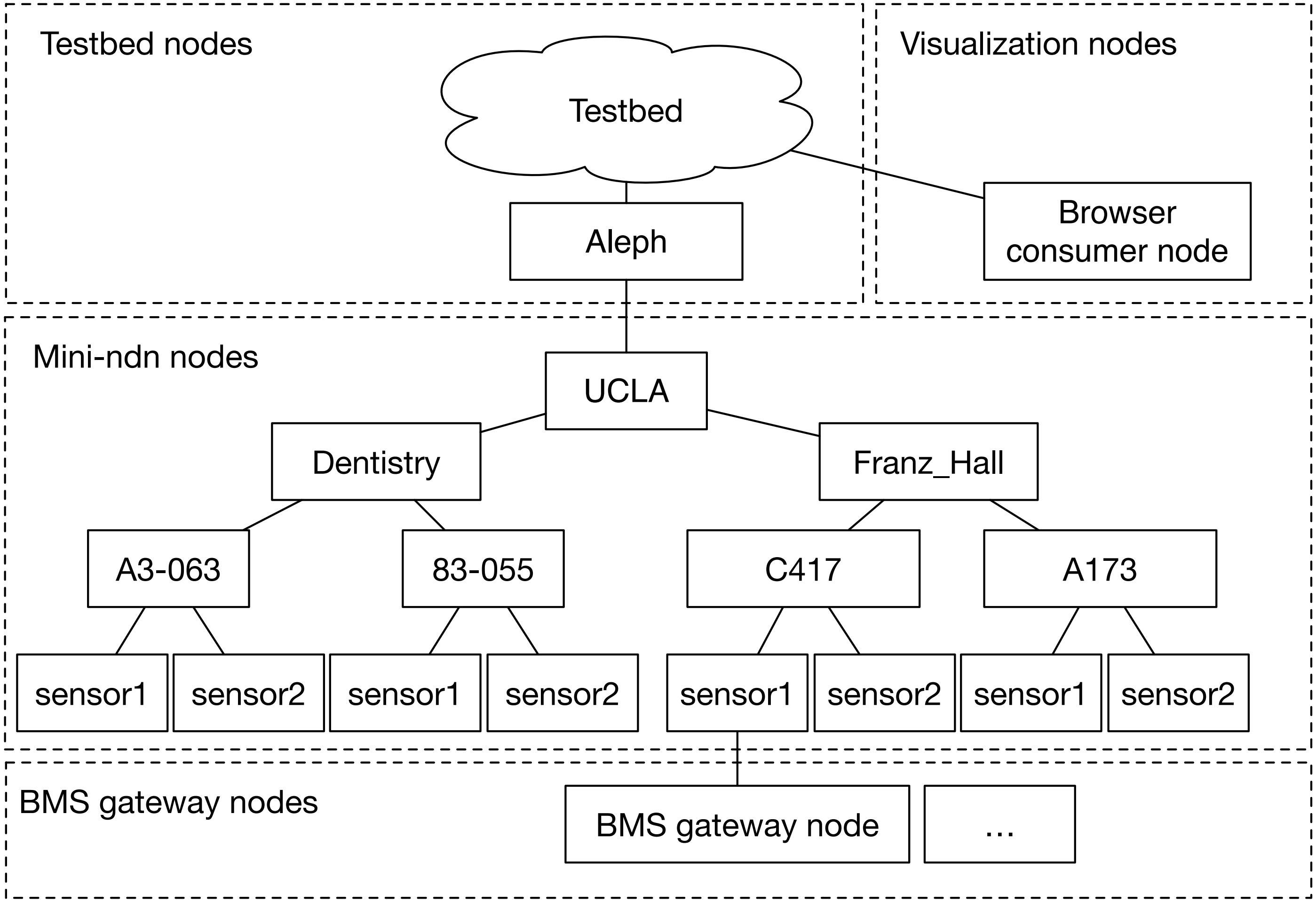


```
string prefix = "ndn/app/bms/ucla/Melnitz/1469/xfmr-6.dmd.inst/data/Electricity";
DataProducer producer(prefix);
auto encryptedContentKeys =
  producer.createContentKey(“20150825T000000”, onCreated);
Data data(prefix + "/Aggregation/avg/1440460800000/1440460801000");
producer.produce(data, “20150825T000000”, content, contentLen, onProduced);
...
GroupManager manager("/ndn/app/bms/ucla/Melnitz/1469A/data/Electricity");
manager.addMember(userCert1, schedule1);
auto groupKey =
  manager.getGroupKey(“20150825T000000”);
```

Group based access control – cont

- Adding identities to a group
  - Upon receiving "group add" command data (signed by an authorized manager)
  - Group generates missing E/D-Keys based on the schedule indicated in the command data

Implementation components



(Each line is a face from parent node to child node, with child's prefix registered)

Implementation components - cont

- BMS nodes
  - PyNDN applications running on mini-ndn
  - Three levels of BMS nodes (building, room, device); each node has its own nfd and BMS node processes; routes are statically configured
  - Panel-level nodes collect data from sensor gateway nodes
- Browser interface for data visualization: NDN-JS

Current progress

- Current implementation has
  - Gateway publisher (working on real time data)
  - BMS nodes running in mini-ndn
- Todo
  - Group based access control
  - Resume connection to real data

Thanks

Design Addons

- Parent nodes could keep subsampled raw data. (Express interest with certain timestamp range excluded)
- Parent nodes could keep a full copy of recent data.
- If needed, parent nodes could be configured to calculate aggregates from specific nodes in its subtree, and publish the data under

/<prefix>/data/power/aggregation/<encoded_list_of_node_names>/min/0/10

- Command data verification (node commands, group commands): verify that a command comes from an identity authorized to give this command