BMS-Repo Design Choices

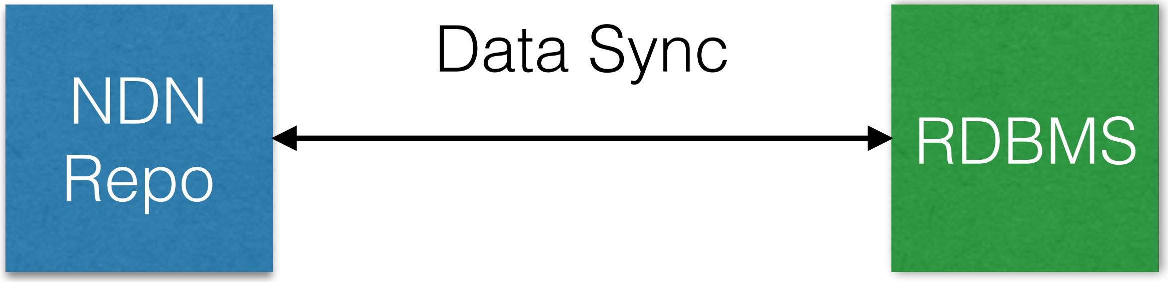
Wentao Shang (UCLA)

Why another repo?

- UCLA Facility Department stores BMS data in relational databases and queries data with SQL
- SQL is very useful for data analysis
- Therefore we want to add SQL support into our NDN-BMS system

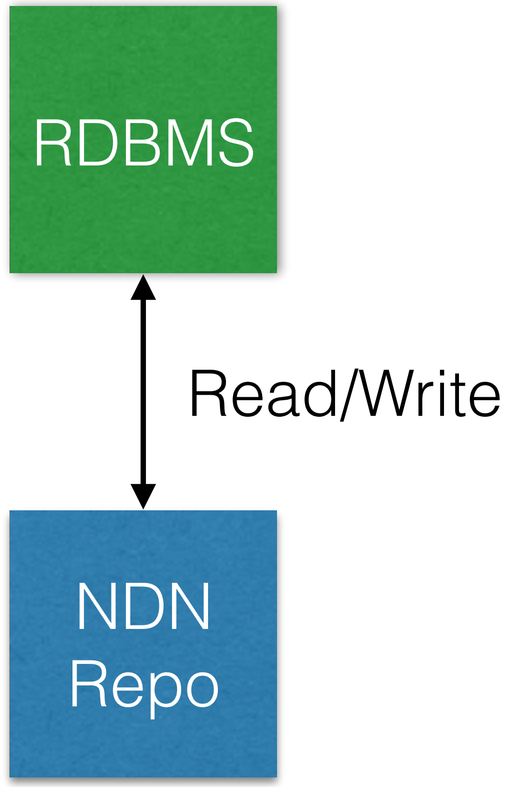
A trivial solution…

- We can run NDN repo and RDBMS in parallel
- Cons: data duplication, which leads to maintenance and synchronization cost



Second design: RDBMS over NDN

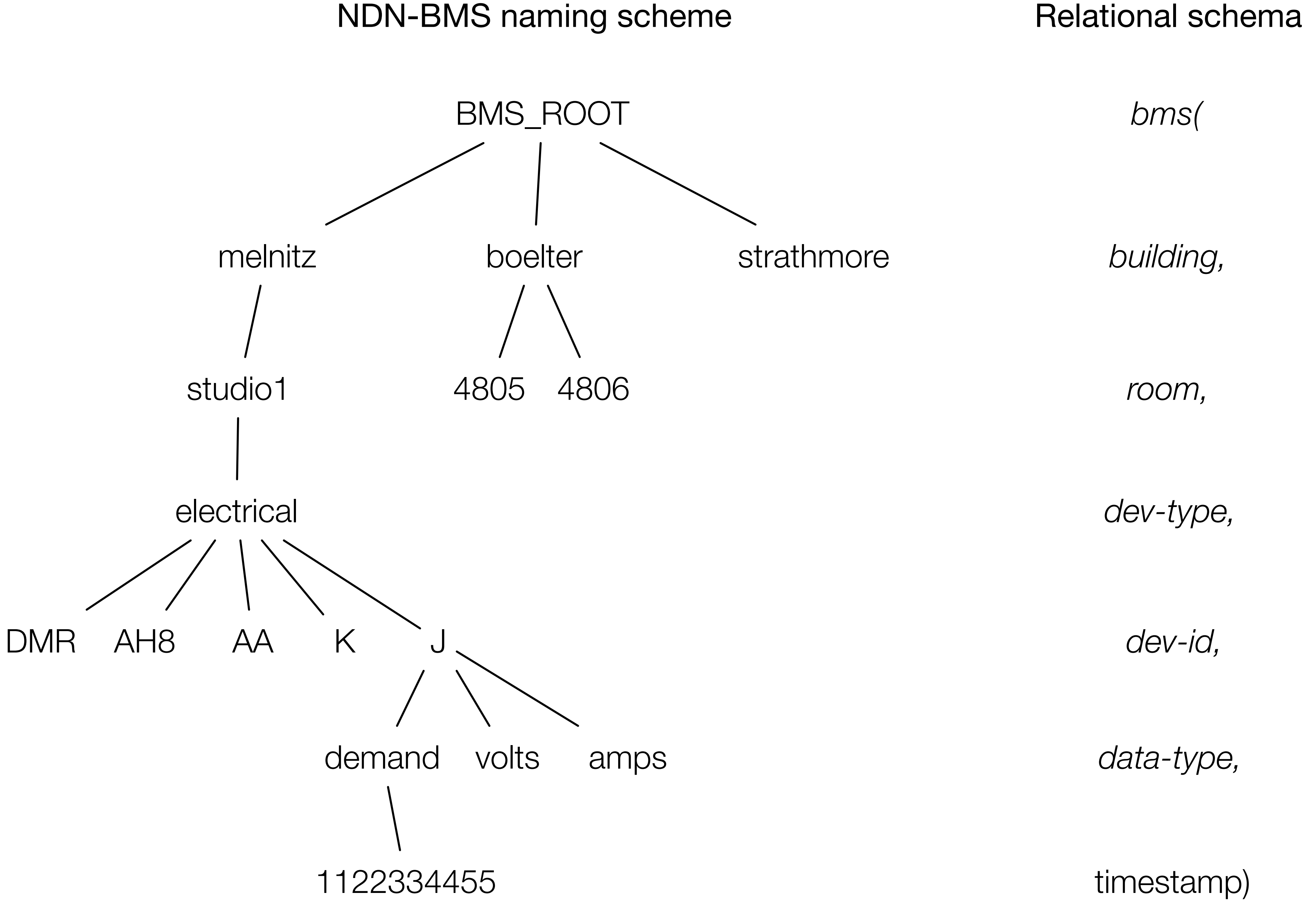
- Option 1: use NDN repo as underlying data store (like a file system)
    - Similar to “Hive/Spark over HDFS”
    - Cons: lose the power of encoding application semantics into NDN naming



Second design: RDBMS over NDN

- Option 2: decompose SQL queries into NDN Interests
    - “Ideal” solution, but hard to achieve
    - Still looking into it…
- Fundamental challenge is the difference in the querying power

Implicit schema in NDN-BMS data naming



NDN Interest vs. SQL query

- Observation: *if we map NDN naming to relational schema, NDN Interest is equivalent to the σ operator in Relational Algebra (i.e., the WHERE clause in SQL)*

- SQL is way more powerful (RA-complete)…

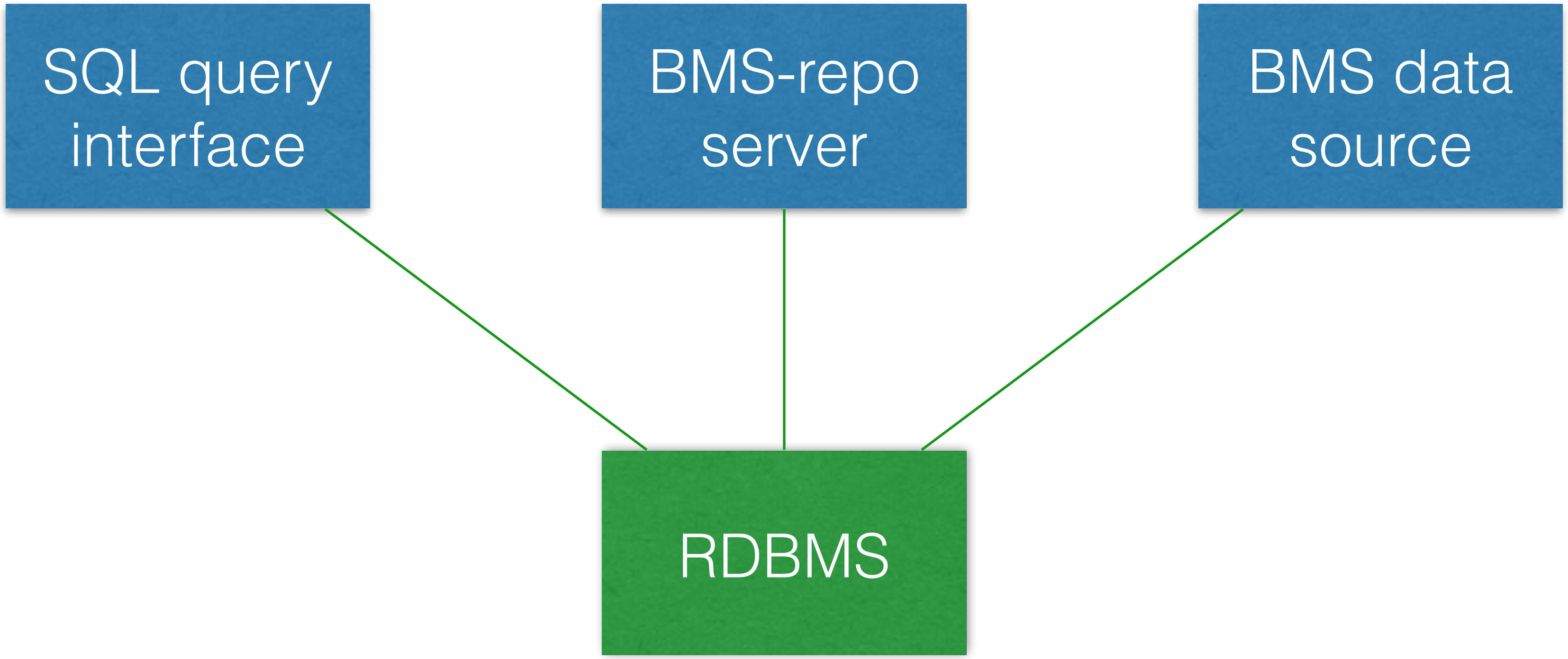
- … which means it is possible to express NDN Interest using SQL query

Example

- Interest: /<prefix>/melnitz/studio1/electrical/AA/voltage, Exclude=(ANY, T1), ChildSelector=0

- SQL query: SELECT * FROM bms WHERE building = ‘melnitz’ AND room = ‘studio1’ AND devtype = ‘electrical’ AND devid = ‘AA’ AND devtype = ‘voltage’ AND (NOT timestamp <= ‘T1’) ORDER BY timestamp ASC LIMIT 1;

Third design: NDN-over-RDBMS



Challenges

- Efficiency: can be improved by pre-packaging (and pre-signing) each single data point
- Scalability:
  - Use application-level data sharding
  - Or use better database…

Thanks!

- Suggestions on the SQL query decomposition algorithm are highly welcome! :-D