What can Databases do for Overlays?

Designing Overlay-Software is tedious work
- Designing topologies for specific requirements
- Sockets, serialization, message processing, ...
- Handling failure of networks and nodes, maintenance, ...
- Distributed debugging, performance optimization, ...
- Avoiding code dependency on incompatible frameworks
- Typically 10,000-30,000 lines of Java/C/C++

... and doing all of this by hand?

⇒ Overlay-Software needs Abstractions ⇐

Local Decisions need Local Data Management

Constraints
- local topology rules specify and constrain properties of neighbouring nodes

Triggered Maintenance
- topology maintenance is the perpetual process of repairing and augmenting the local view according to the rules

Ranking
- topology adaptation is the selection amongst the edges permitted by the rules

Quality-of-Service
- topology selection is the choice between the available topologies based on specific application requirements

The Node Views Abstraction

Topology Design – easy as SLOSL

```
CREATE VIEW chord_fingertable
AS SELECT node.id, node.ring_dist, bucket_dist=node.ring_dist-2'
RANKED lowest(nodes+i, node.maxc_latency/node.ring_dist)
FROM node db
WITH log_k = 160, nodes = 1
WHERE node.supports_chord = true AND node.alive = true
HAVING node.ring_dist in (2^i:2^(i+1))
FOREACH i IN [0,log_k]
```

Databases as standard components in Overlay Software
- Key component to store topology data (data about nodes)
- OverML+SLOSL: abstract specification languages vs. 10,000 lines of source code
- Model-View-Controller pattern decouples simple, event triggered components
- Generic exchange of specifications and view data in programmable networks
- Integrative platform for testing, comparing and deploying different overlays

Node Views + SLOSL Overlay Workbench

flexible, integrative, high-level Overlay Design – at your service!