Ensemble Jukebox: A Platform for Distributed Entertainnment

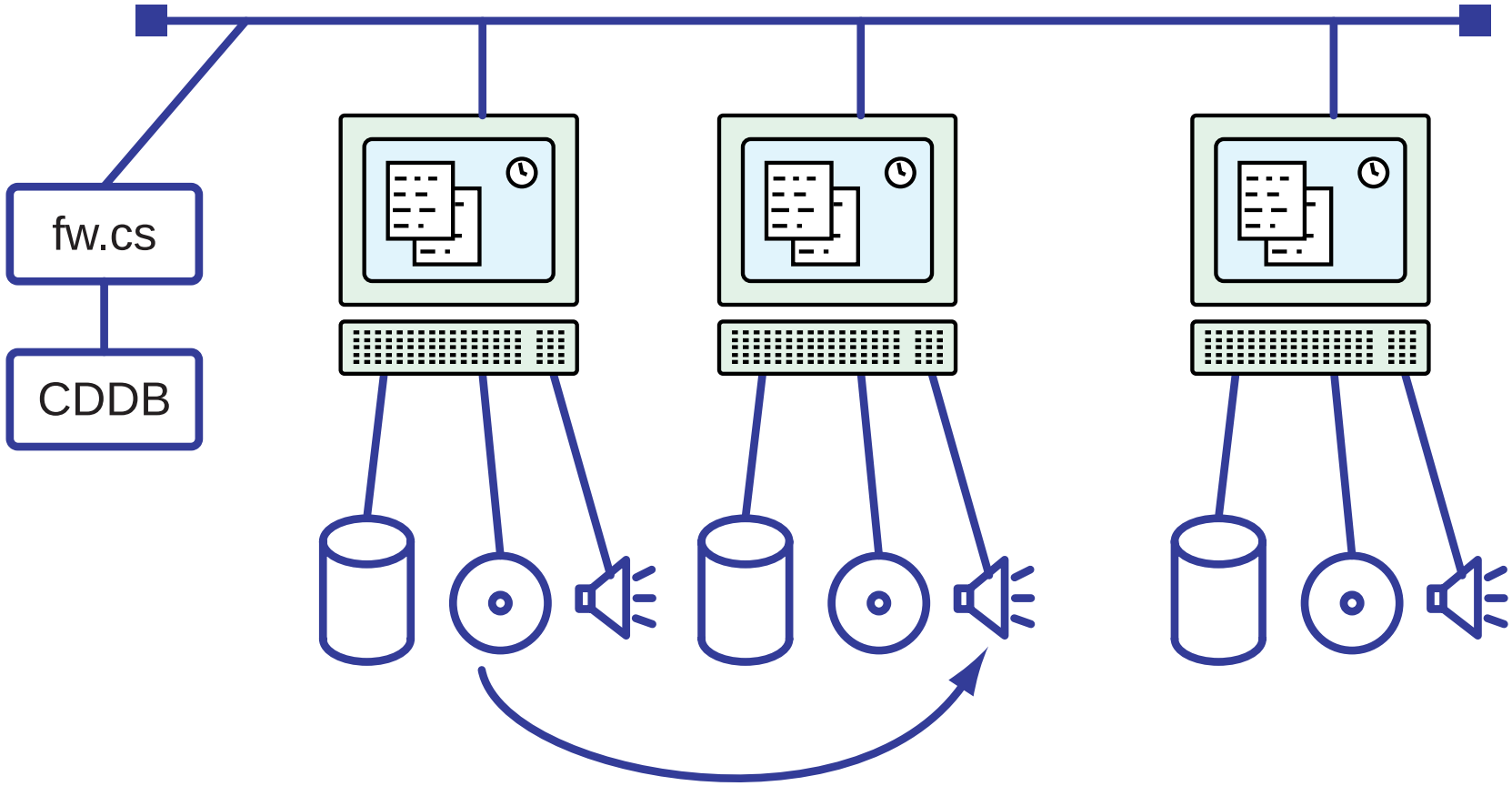
Jason Hickey
Mark Hayden

Outline

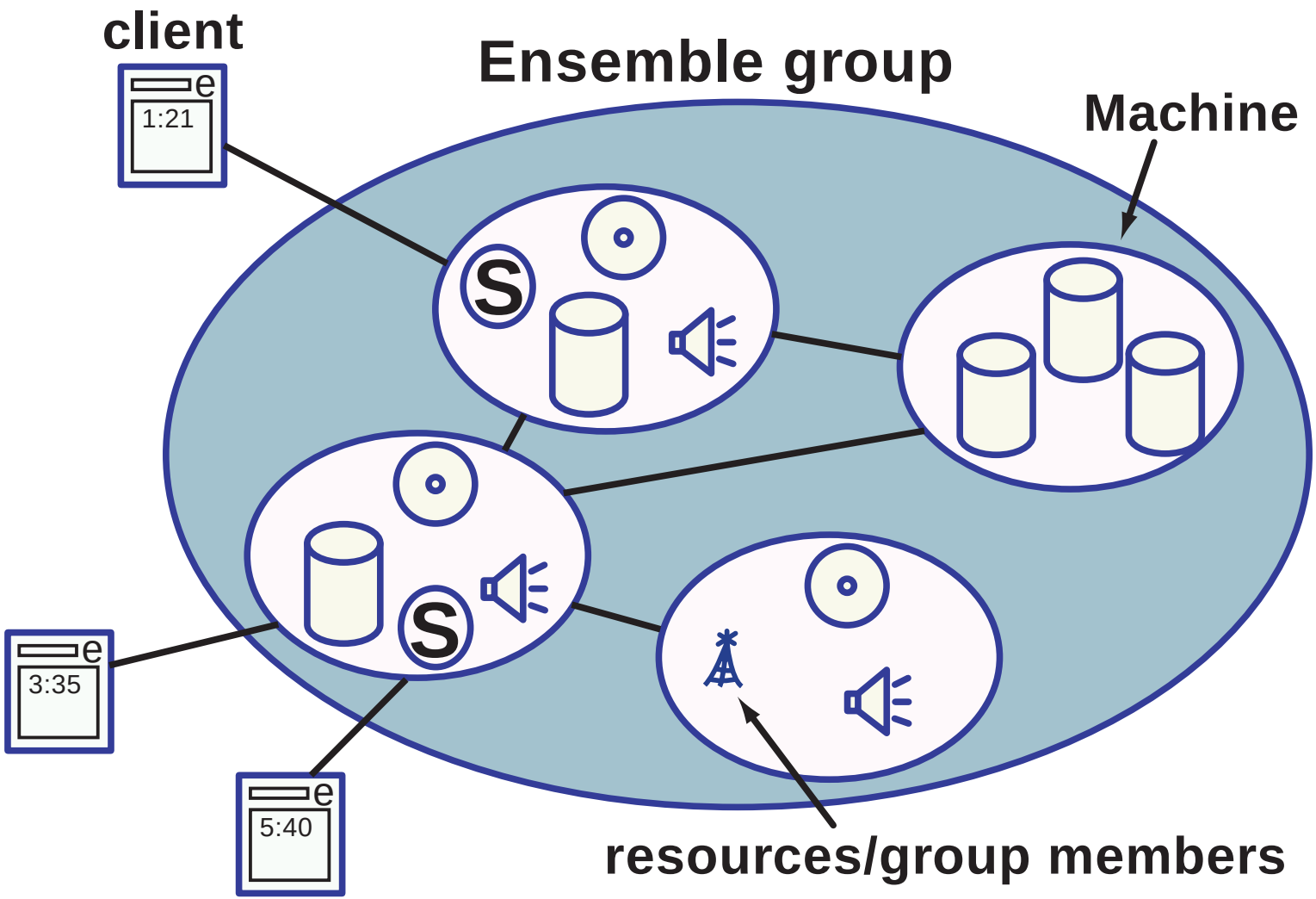
Top down description

- **Jukebox architecture**
- **Group membership promo**
  - abstraction
  - security
  - consistency
- **Group membership wish list**
- **Programming language principles**
- **Open issues**

Physical Architecture



Logical Architecture



Media objects

source

store

sink

stream

Collections: logical documents

- cd or recorded tracks
- span machines

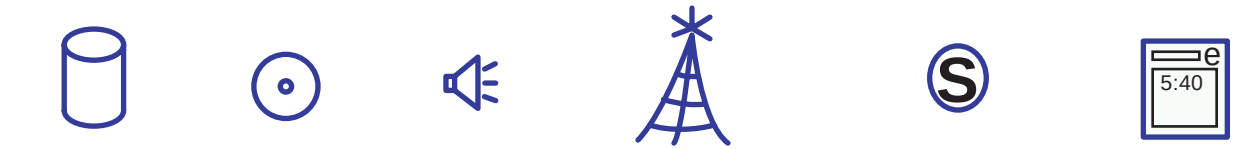
How are all these objects organized?

Group communication systems

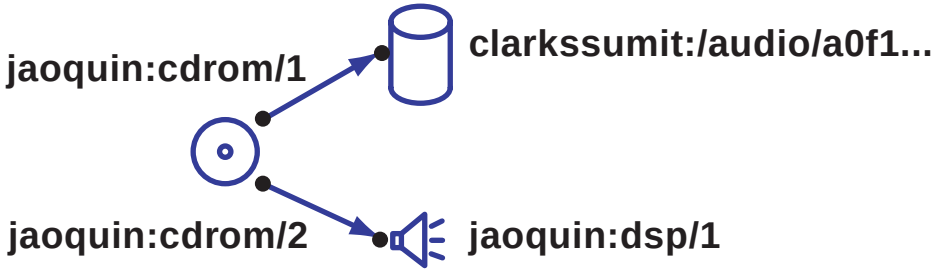
- **Applications are collections of group members**
- **Ensemble *provides*:**
  - communication between members
  - broadcast communication
- **Ensemble *ensures*:**
  - members have identical views of group
  - broadcasts are ordered and atomic

Application design

- Each resource is a group member



- Each member adds its resources to the view
- Resources provide: locks, streams



- Application level flow control (token passing)

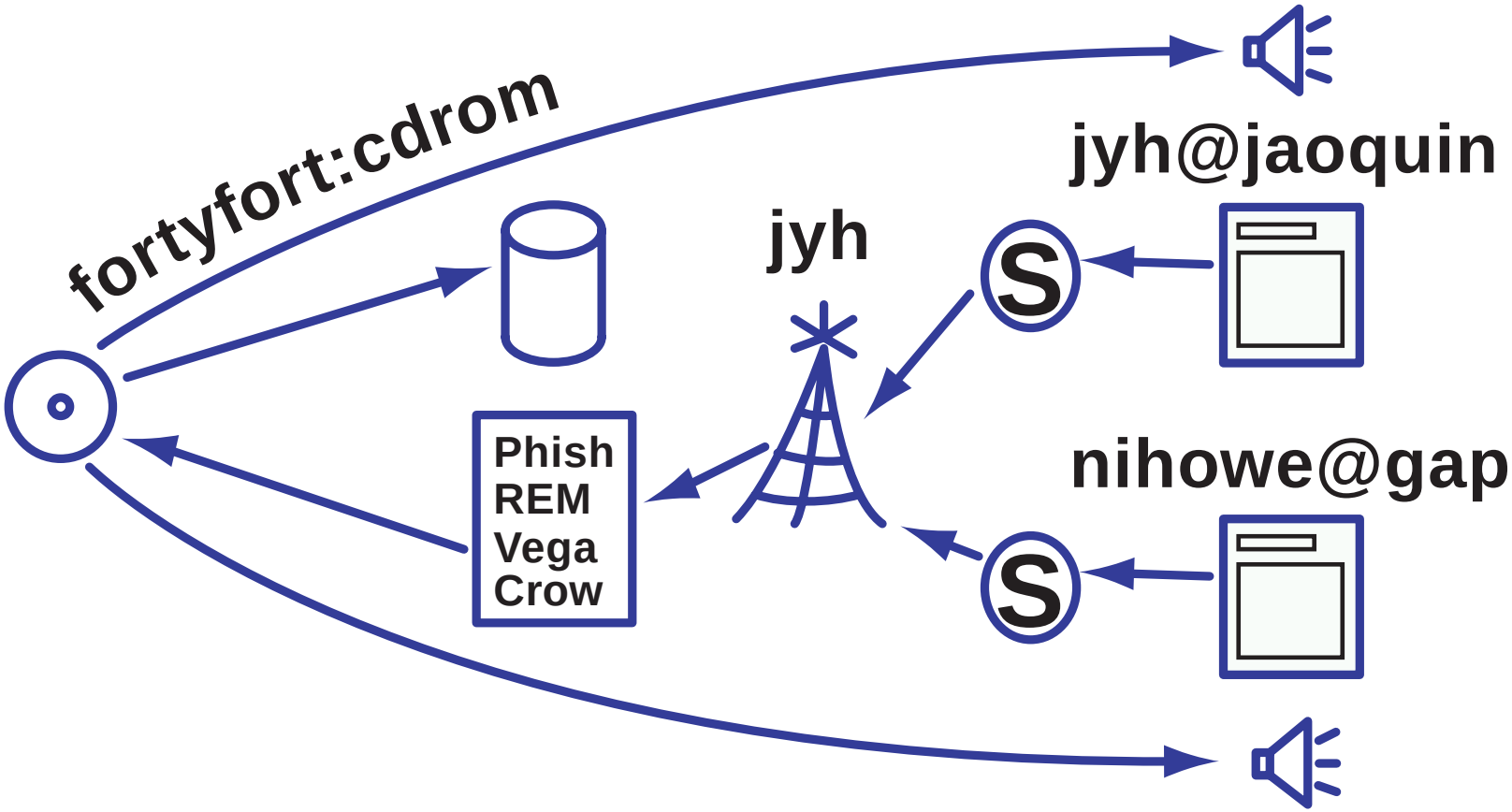
Logical extensions

- **Media are organized as collections**

Phish
REM
Vega
Primus

- **Follows a filesystem model**
  - cp, rm, etc
  - Use windowed navigation (drag & drop, etc)

Radio proxy devices

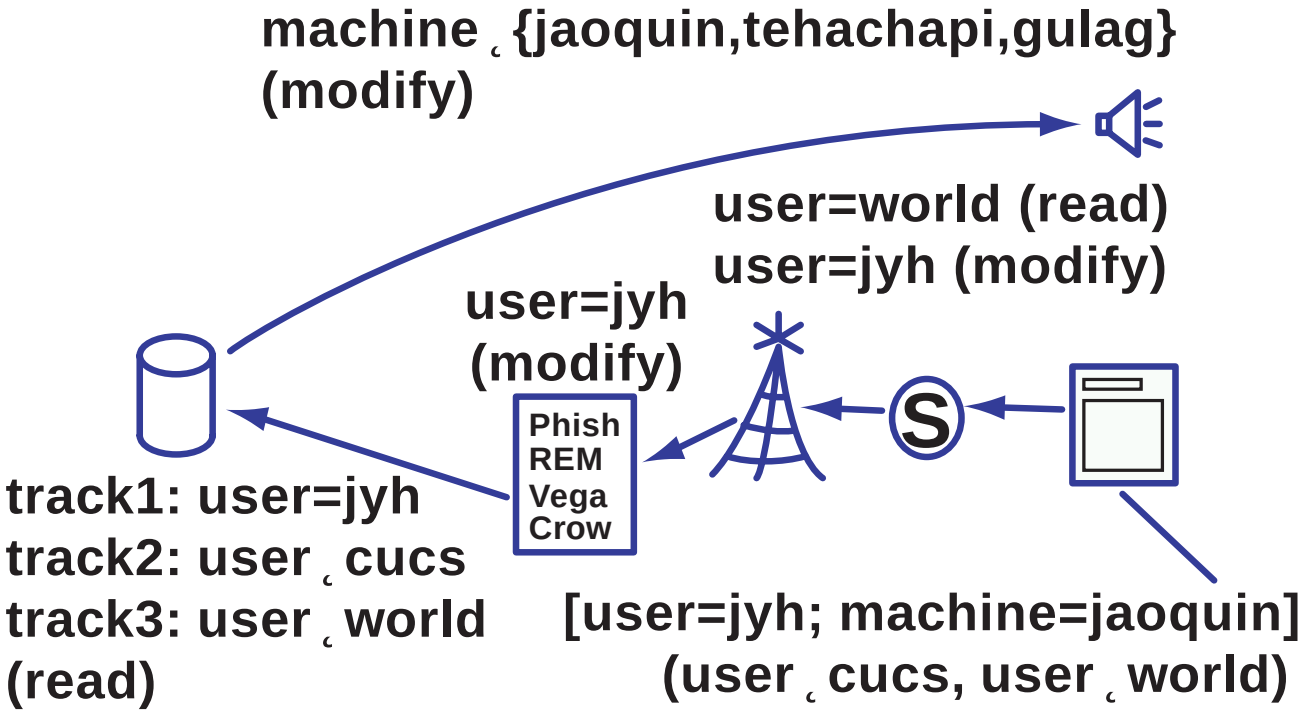


Ensemble support

- **Globally consistent view**
  - Identical views of membership, resources
  - Track database: track → location
  - Atomic broadcast
- **Failures are handled gracefully**
- **Application uses group structure (not physical)**
- **Transparent connection management**

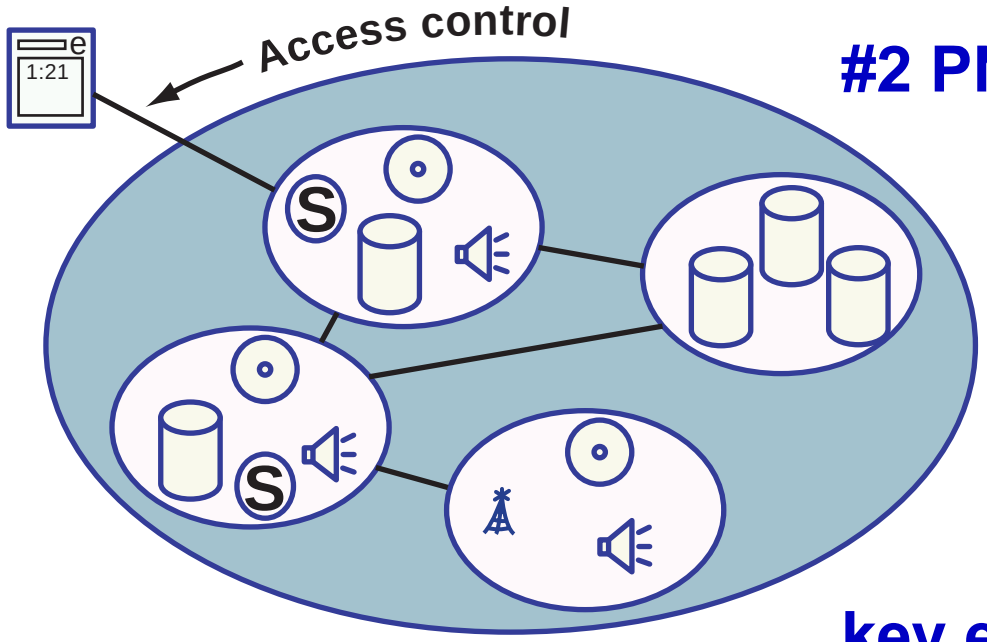
Security

- Cd media is copyright protected
- Users may publish using radios
- Application: lock-based, certificate passing:



Ensemble security

Course-grain network level security



#1 VPN, secure OS

#2 PN, secure protocols

key exchanging

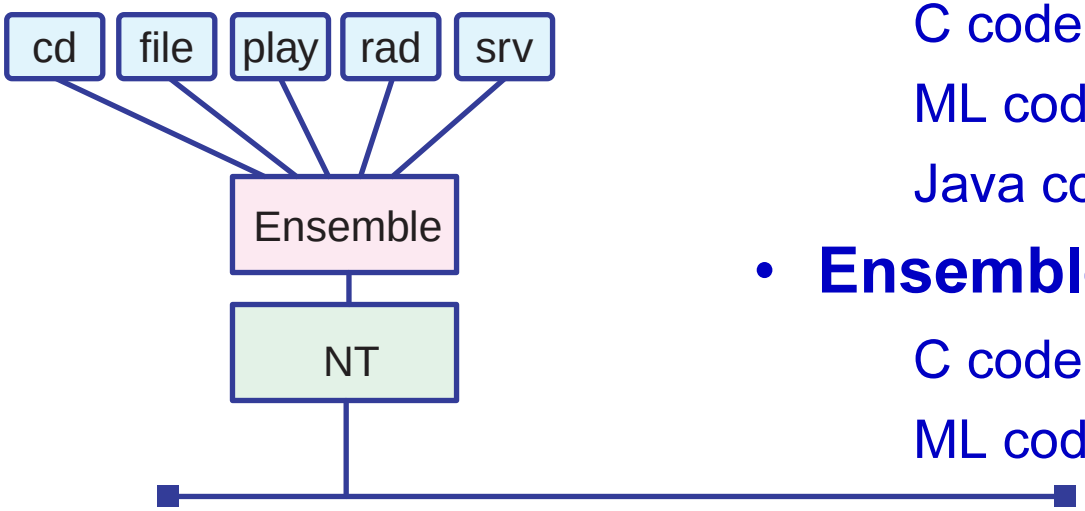
bootstrapping problem

Security issues

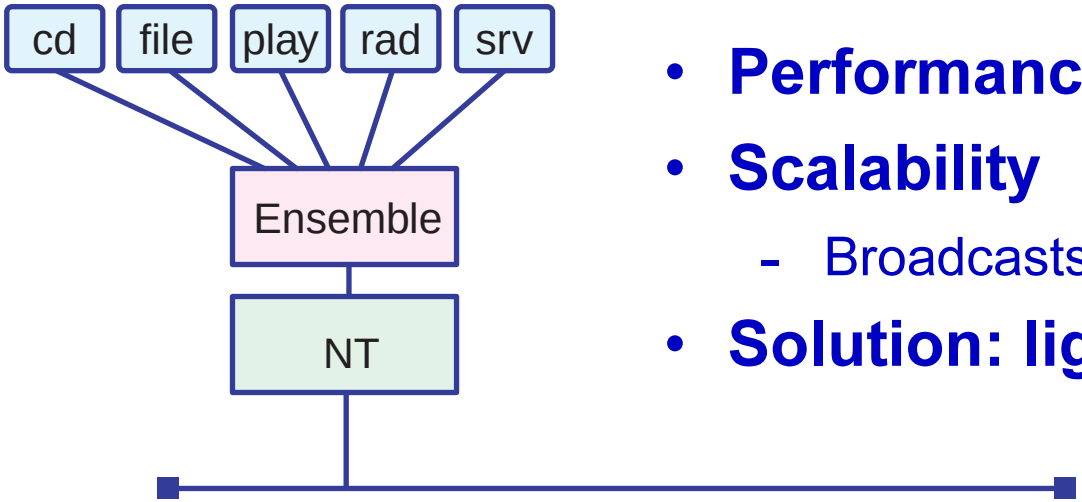
- ***Mechanism* is implemented**
- ***Policy* is simplistic**
  - Who owns copyright?
- **Administration**
  - Current admin is not group-based
  - Hot swapping
- **Need general security framework**

Programming language issues

- **Design is highly modular**
- **Include only needed functionality**
- **Code is functorized**
  - **highly heterogeneous**
    - C code: 17K lines
    - ML code: 46K lines
    - Java code: 14K lines
  - **Ensemble**
    - C code: 13K lines
    - ML code: 58K lines

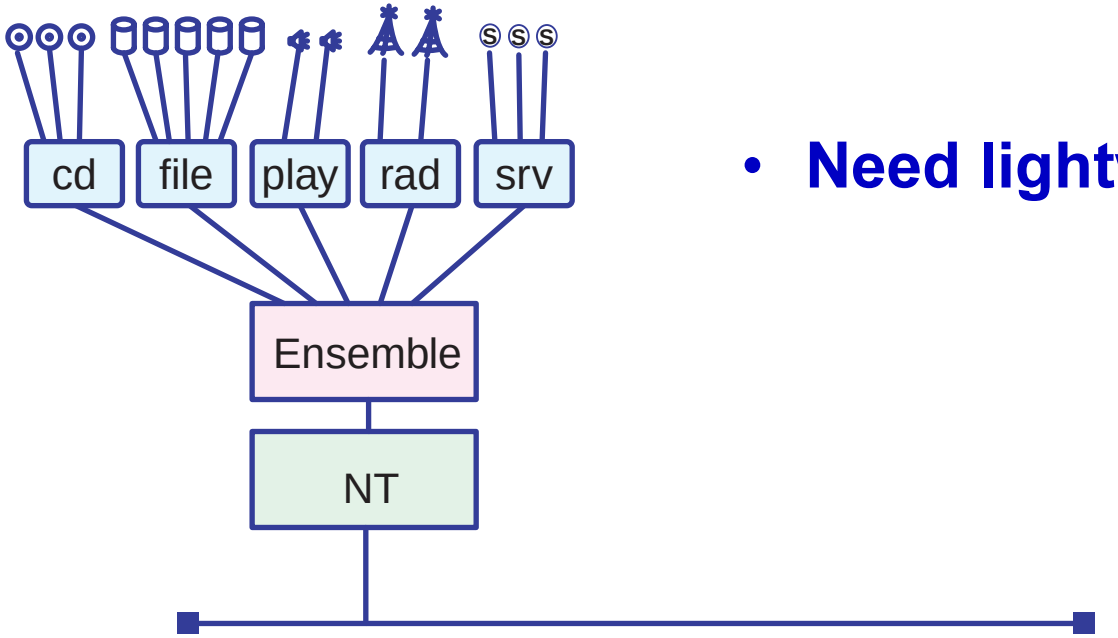


Modularity



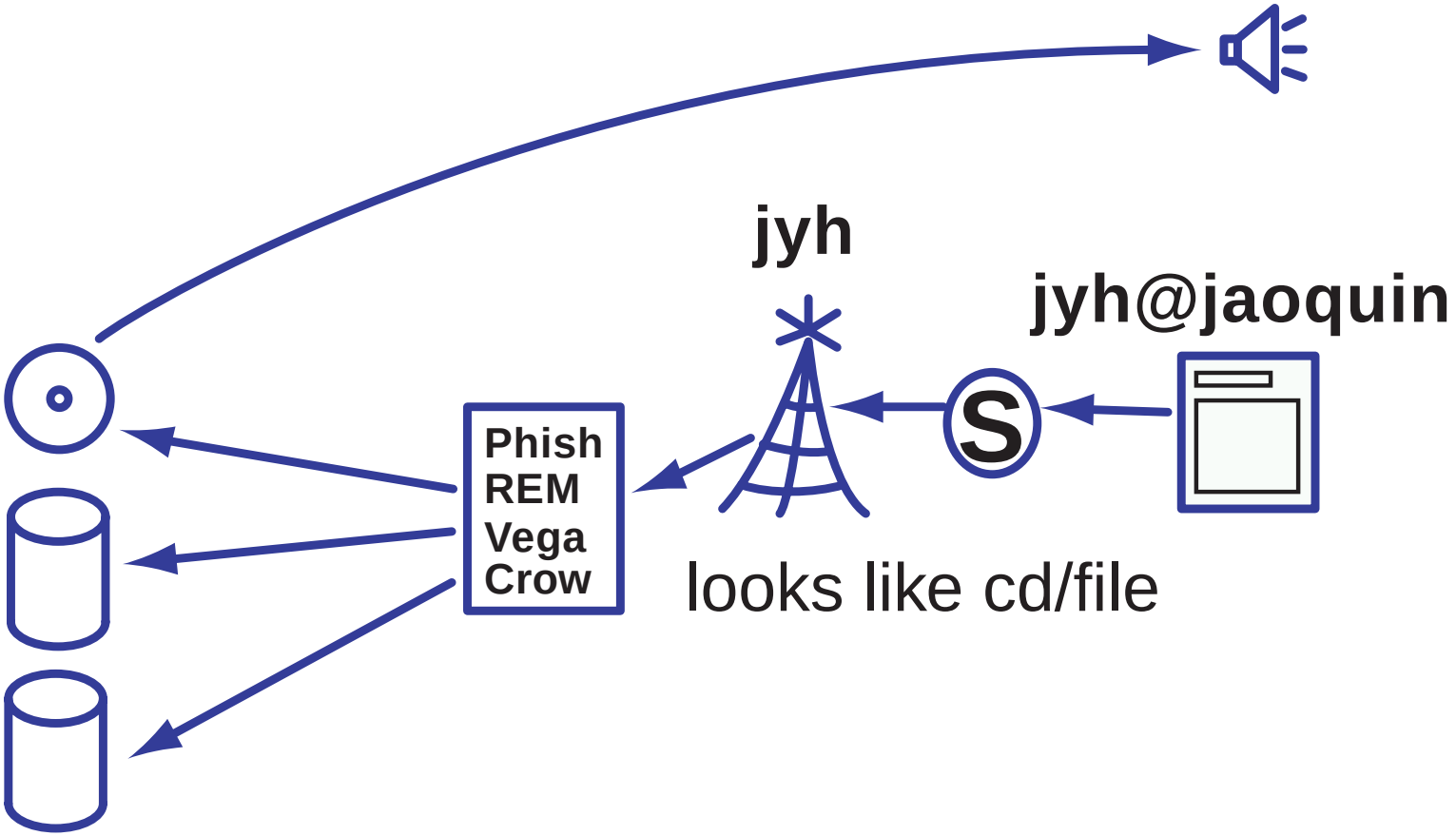
- **Performance penalty**
- **Scalability**
  - Broadcasts, view changes
- **Solution: lightweight endpoints**

Submodularity



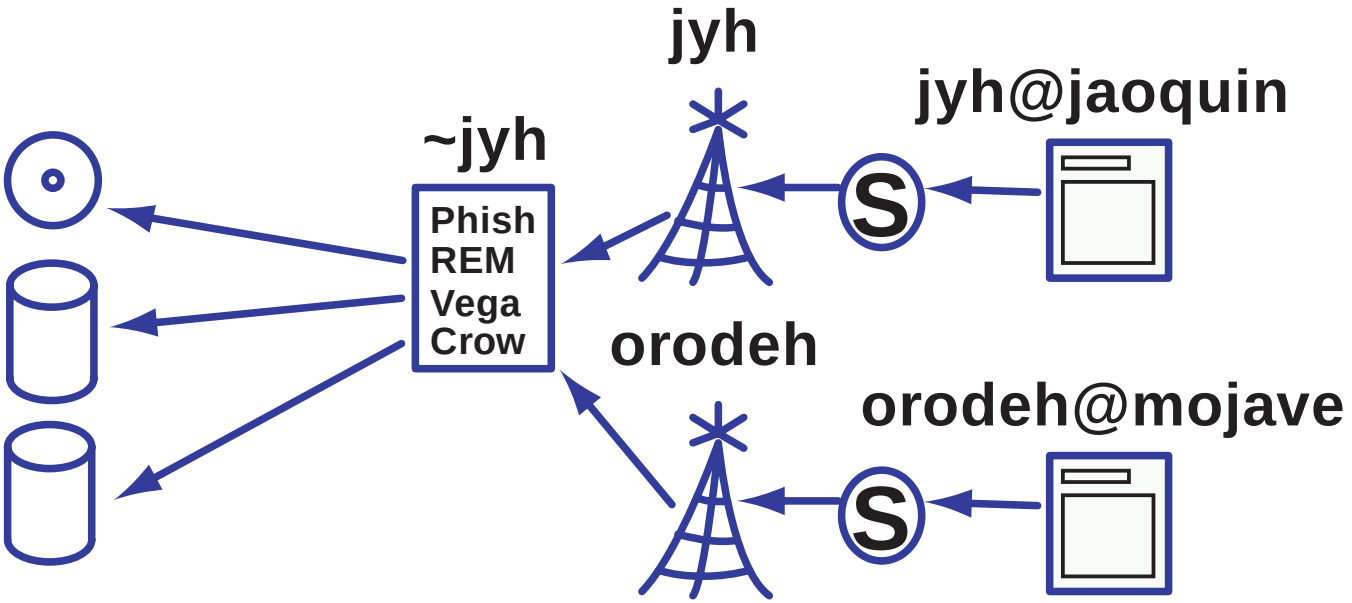
- **Need lightweight groups**

Proxy devices

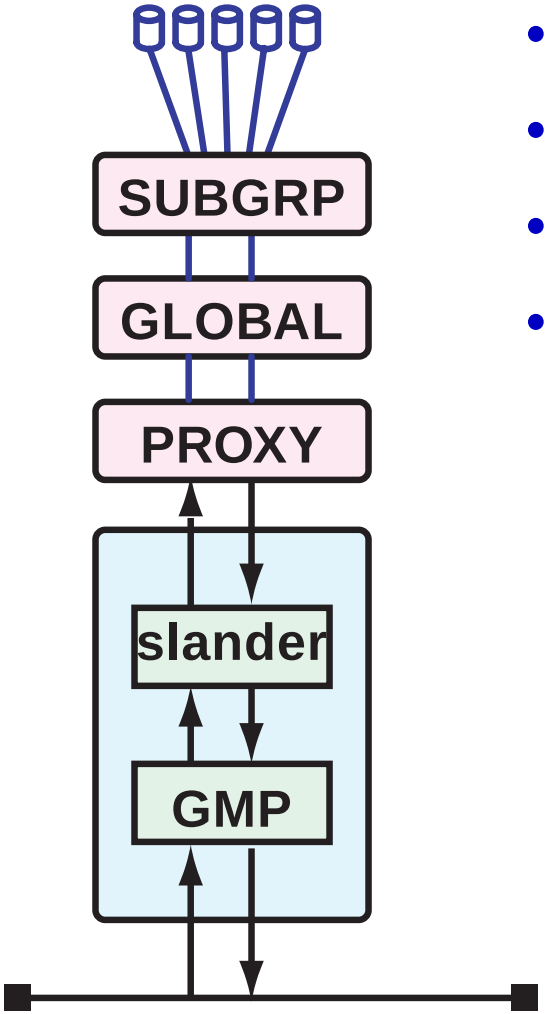


Global databases

- Global list of resources
- Common filesystem semantics



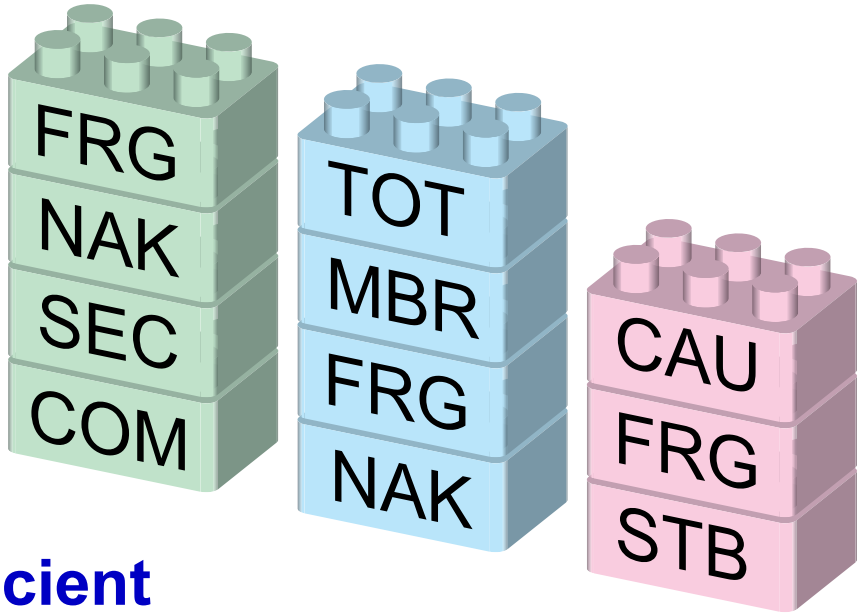
Application layers?



- General, common functionality
- Global consistency
- Proxy
- Process-local lightweight groups

More programming language issues

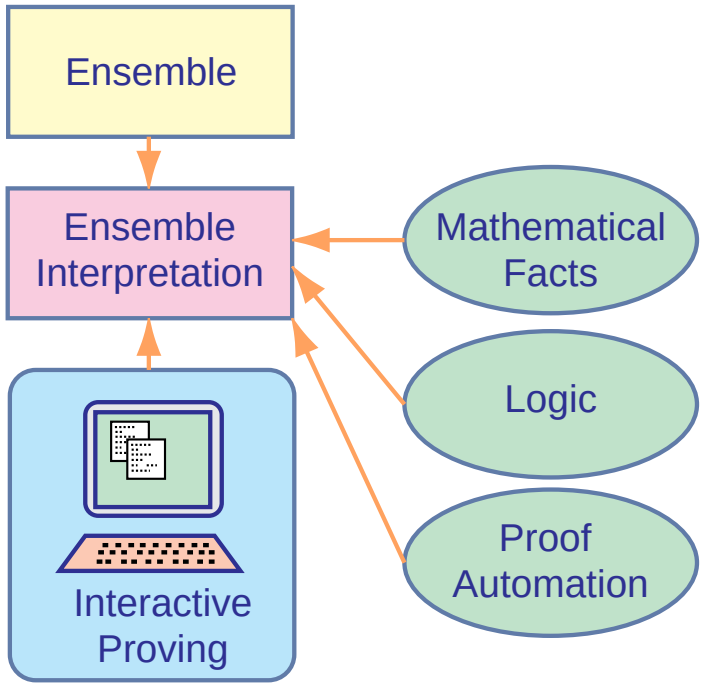
- Protocols are difficult to get correct
- Need assistance maintaining invariants
- Help with optimization



- Layered protocols are inefficient

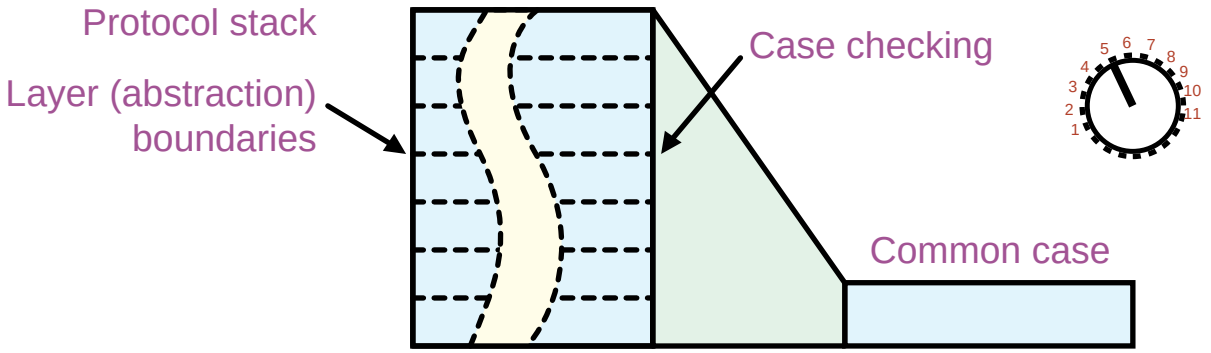
Nuprl

- **Deductive proof system**
  - Interactive theorem proving
  - Heuristics, decision procedures, tactics
- **Type theory**
  - Functional programming language
  - Expressive type system
  - Refinement of core ML



Fastpath optimization

- **Inlining (beta-reduction)**
- **redundant code elimination**

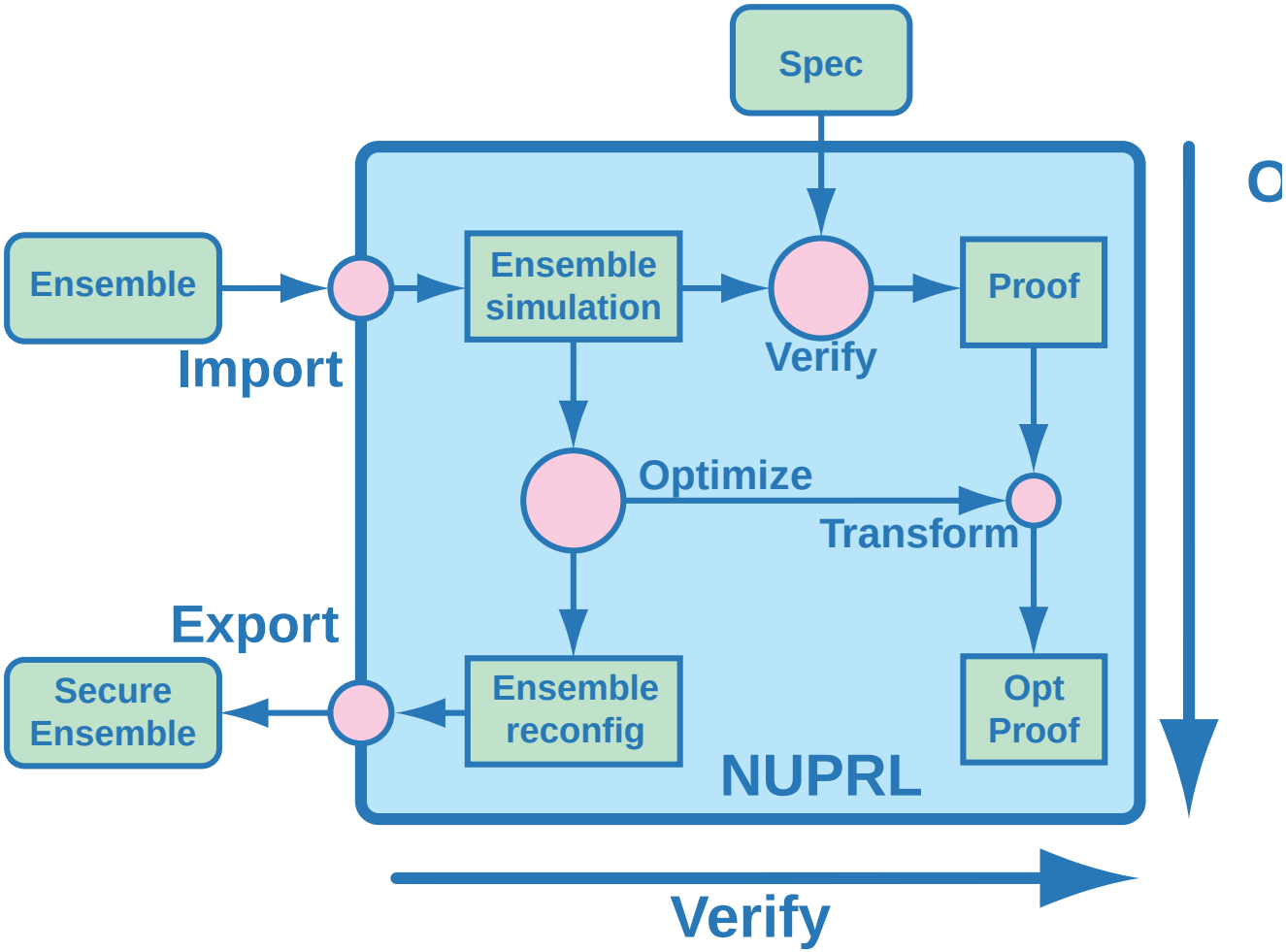


module type *LayerSig* = •••
module *FIFO* : *LayerSig*
module *STABLE* : *LayerSig*
module *GROUP* : *LayerSig*
•••

FIFO o *STABLE* o *GROUP* o ••• = *Common*

Nuprl programming environment

Secure program transformation



Nuprl-Light: logical programming environment

- **Formal interpretation of modules**
- **Add formal types to module system**
  - val sort : $\forall$T: Type. $\forall$l: T list. { l' : T list | Sorted(l, l') }
- **Invariant expressions:**
  - val cdroms: cdrom list
  - axiom cd_exists: $\forall$cd: cdrom. cd $\in$ cdroms $\Rightarrow$ Online(cd)
- **Optimization**
  - Function inlining
  - Program transformation
  - Tactic-based domain knowledge

Summary

- **EJB is a testbed for exploring**
  - Group communications for multimedia
  - Security in distributed systems
  - Principles of programming
    - Verification
    - Optimization
    - Rich, formal programming environments
- **EJB is a demo for generating interest in common PL/distributed systems**
  - Problems are solvable
  - Results are visible