Exchangeable, Application-Independent Load Balancing for P2P Simulation Frameworks

Daniel Warneke and Ulf Rerrer-Brusch
Complex and Distributed IT-Systems
daniel.warneke@tu-berlin.de

What to expect of this talk...

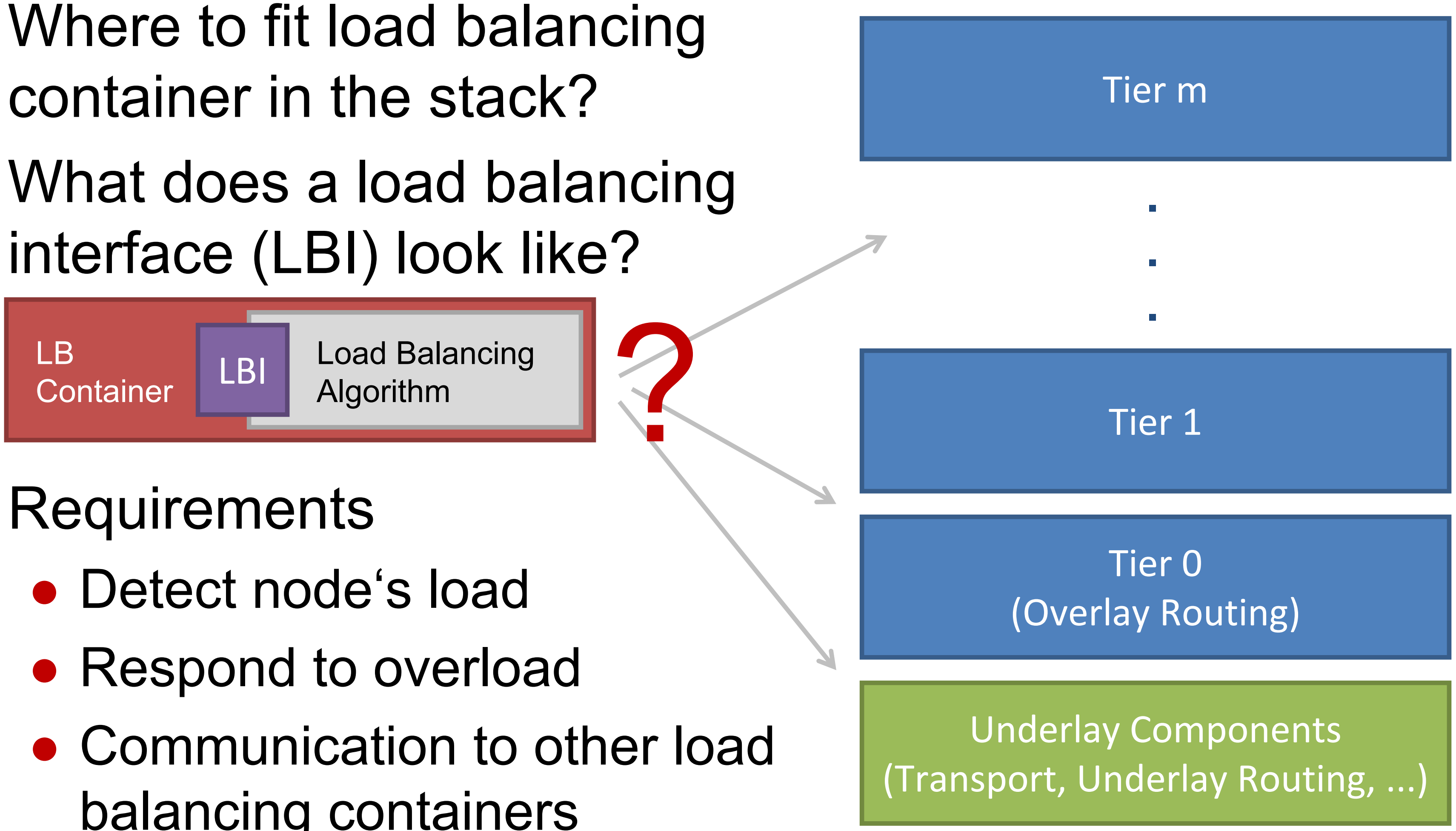
- P2P simulation frameworks are powerful and modular
  - Easy to adapt to own needs
  - More advanced issues can be rapidly approached
- Load balancing is not available as modular block
  - Although often vital, no framework comes with support
  - Application developer has to take care on his own
- How to provide load balancing in a reusable manner?
  - ... open to a variety of concrete load balancing algorithms
  - ... but independent of the P2P application

Outline

- Challenges
- Design approaches
- Interface definition
- Implementation
- Evaluation

Challenges

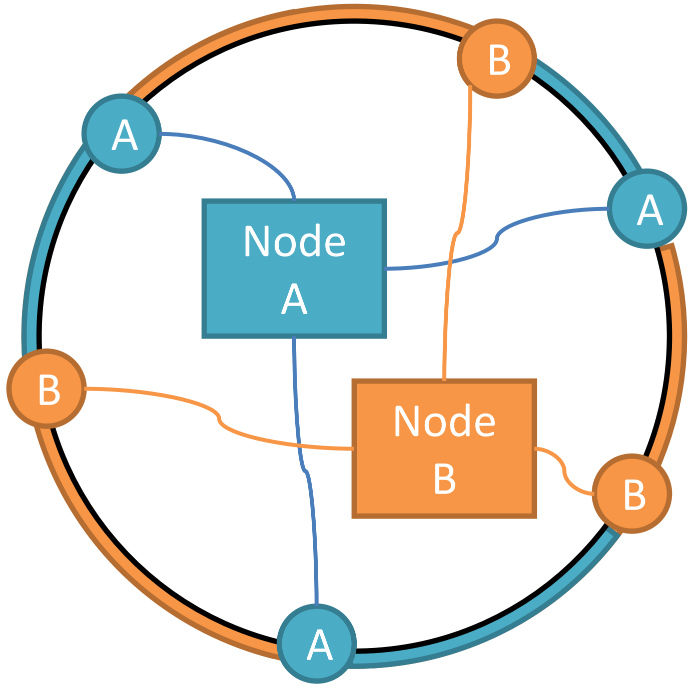
- Where to fit load balancing container in the stack?
- What does a load balancing interface (LBI) look like?

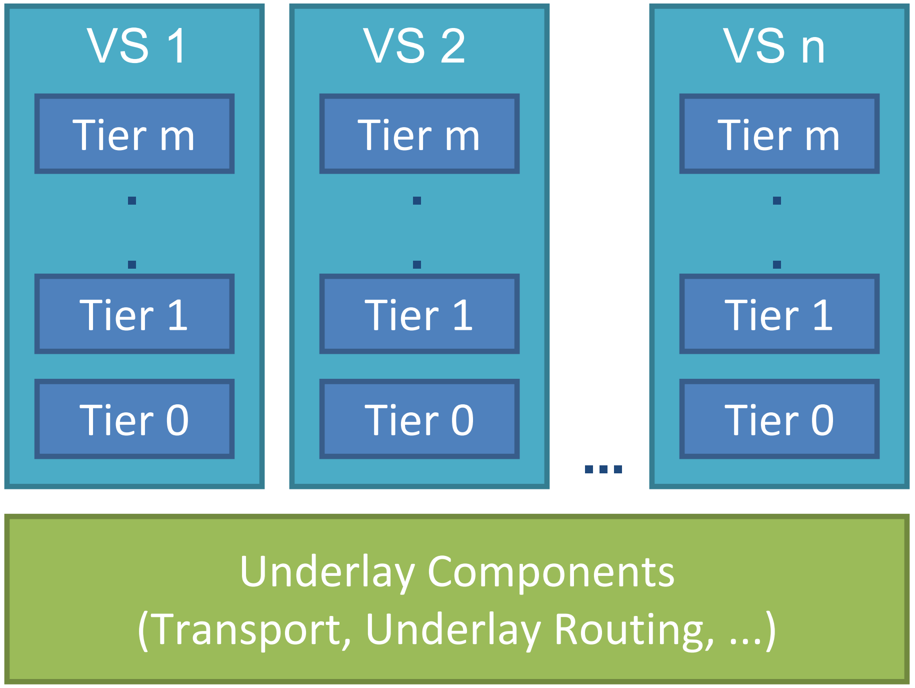


- Requirements
  - Detect node‘s load
  - Respond to overload
  - Communication to other load balancing containers

Approach: Virtual Servers

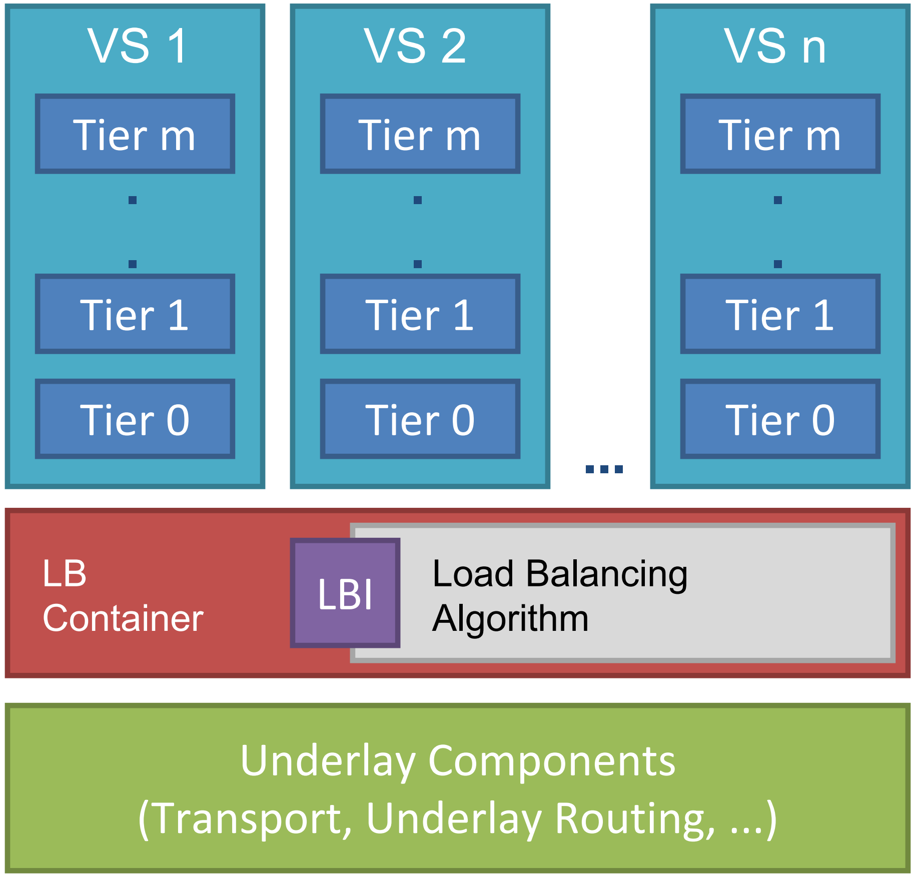
- Idea: Multiple overlay stack instances on one node
  - Each instance represents autonomous peer
  - Concept allows only little assumptions about internals





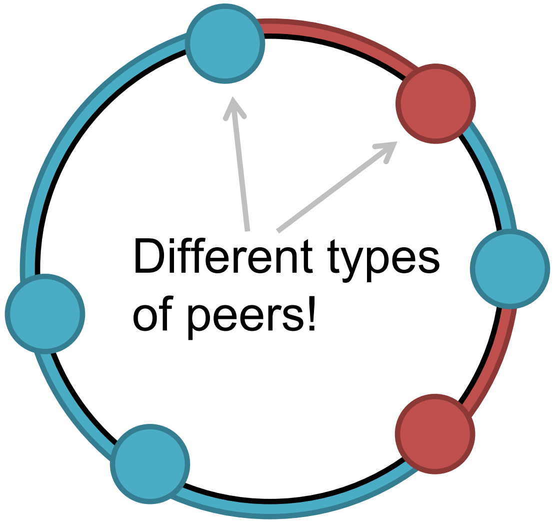
Load balancing with VSs

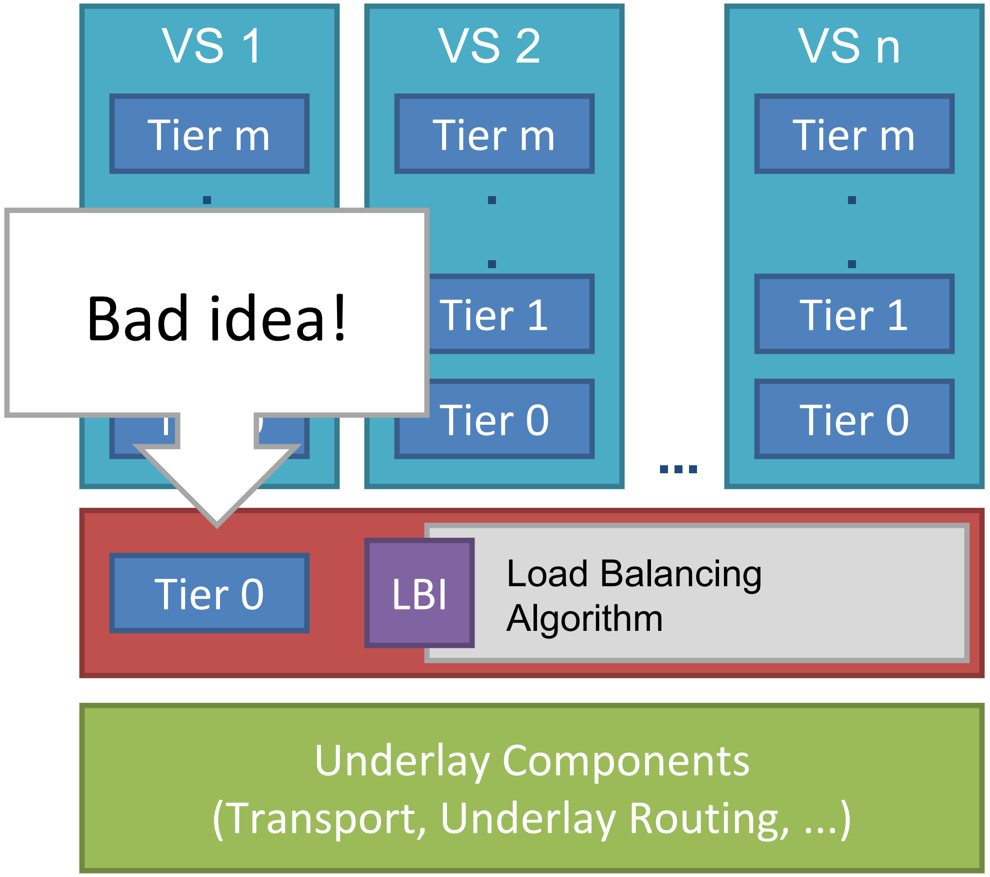
- Requirements
  - Detect node‘s load ✓
  - Respond to overload ✓
- Communication to other LB containers?
- Many LB schemes use overlay network to arrange rendezvous
  - Not possible yet ☹



Overlay access to LB container?

- Run Tier 0 (Overlay Routing) in LB container?





- Better approach:
  - Use Tier 0 of one VS
  - Use bootstrap node if node has no VS

Load balancing interface

Control/list virtual servers

-key startVirtualServer(key id, double delay)

-boolean stopVirtualServer(key id, double delay)

-double getLoad(key vsid)

-key [] getVirtualServers()

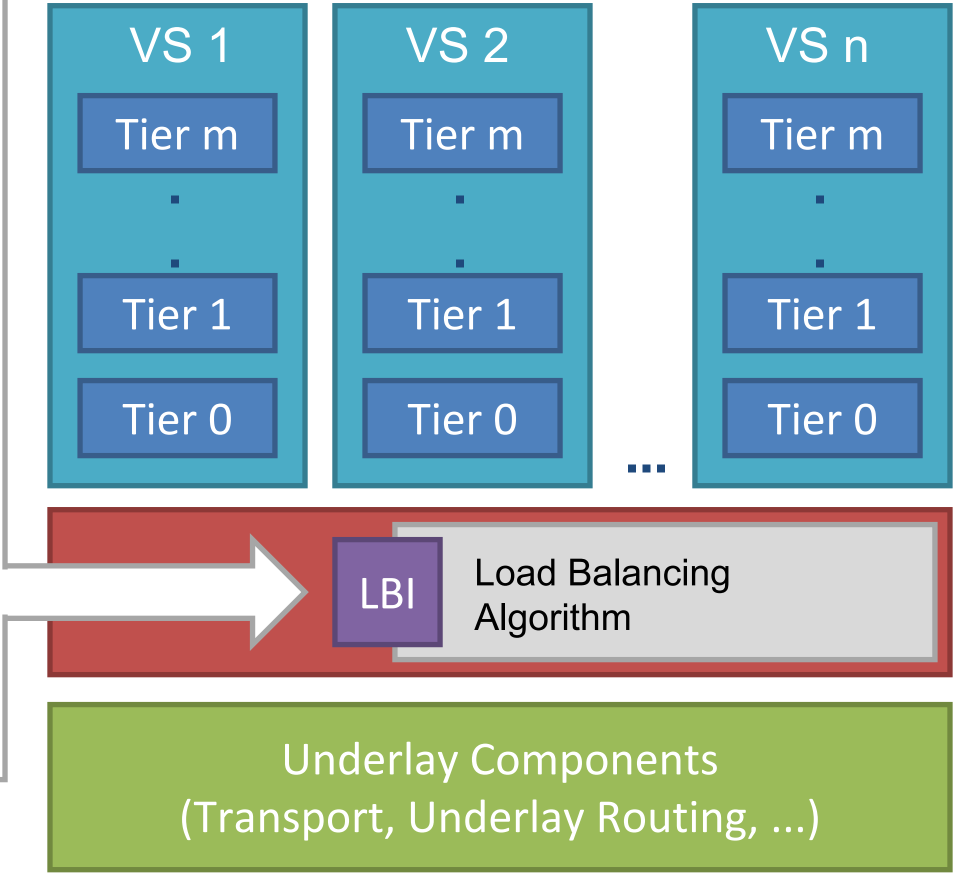
Communication LB containers

-void routeToID(key destid, message msg, key vsid)

-void routeToAddress(transportaddress dest, message msg)

-void receive(key destid, message msg) (CB)

CB = callback function

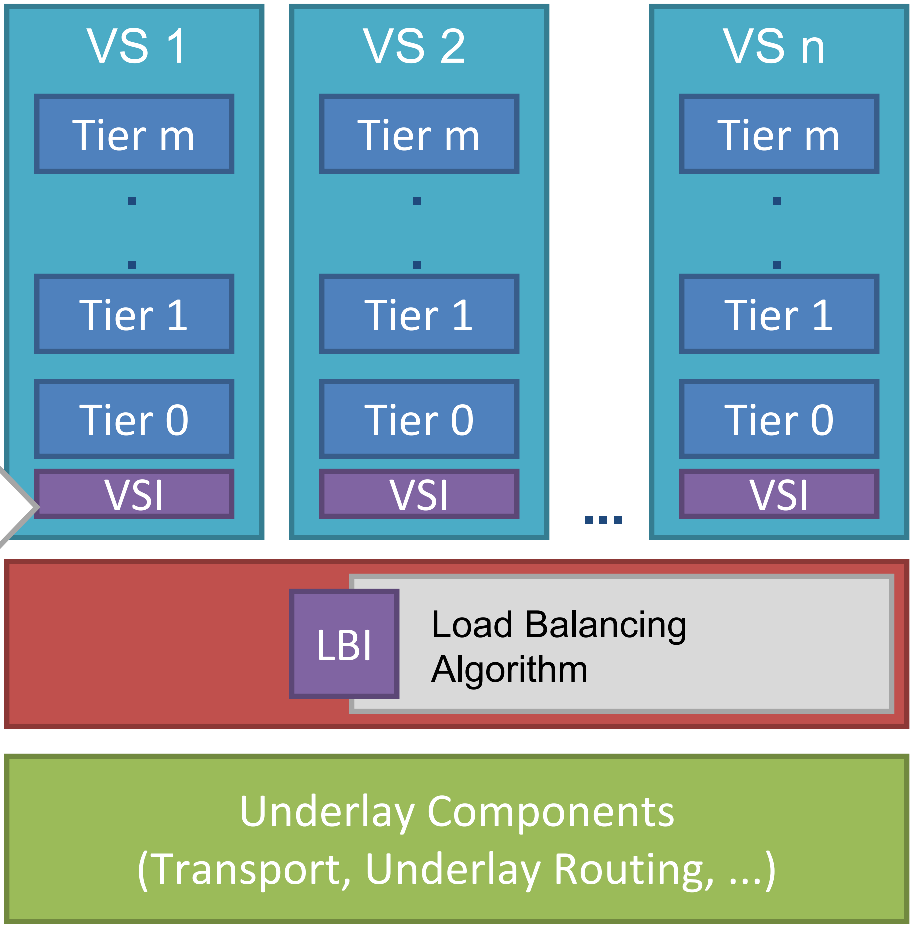


Virtual server interface

- Optional interface in VSs
- Required to
  - handle graceful shutdowns
  - provide application-specific load information

Application-specific load
-double getLoad() (CB)

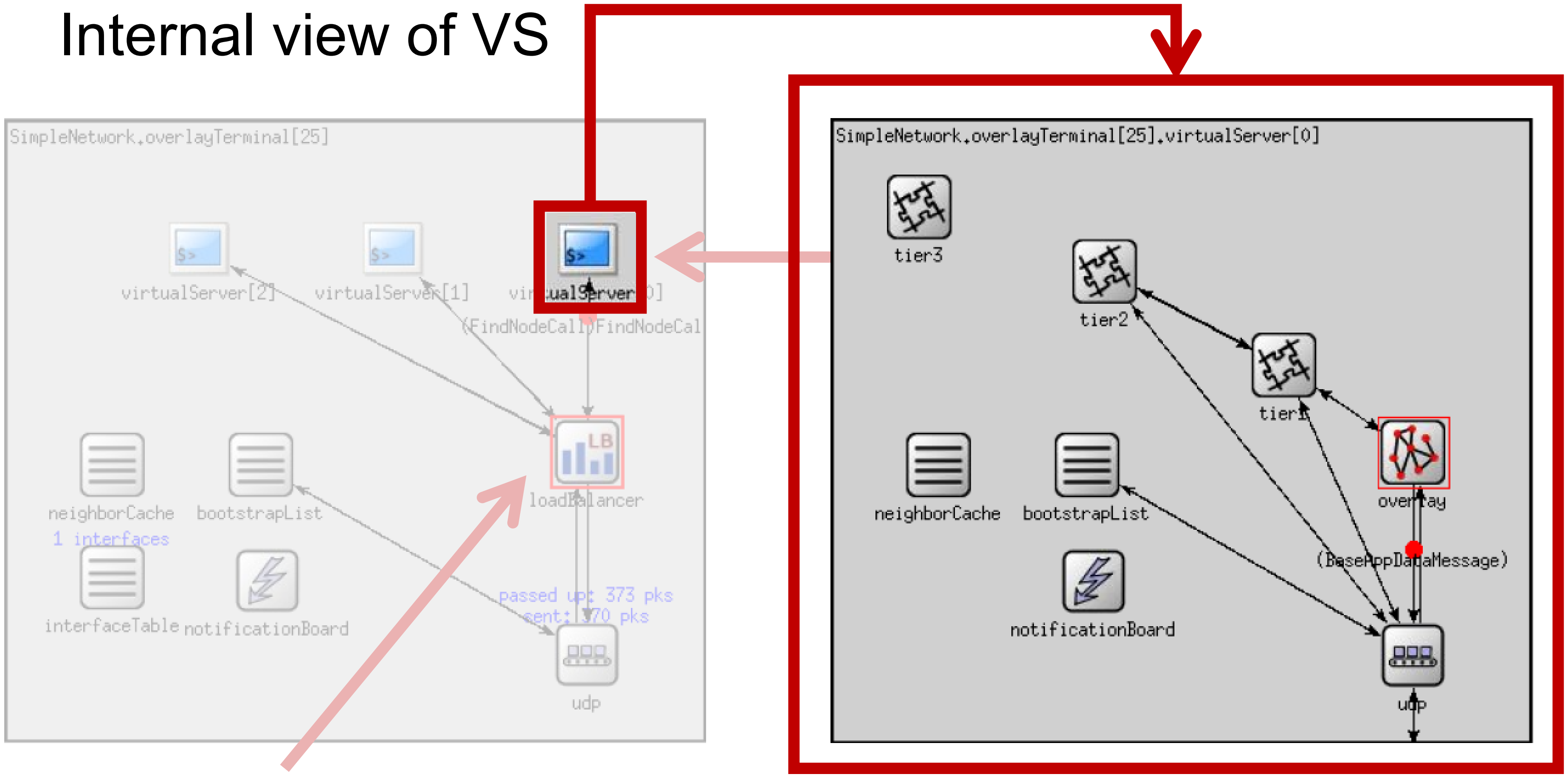
Graceful shutdown
-void gracefulShutdown(double delay) (CB)



CB = callback function

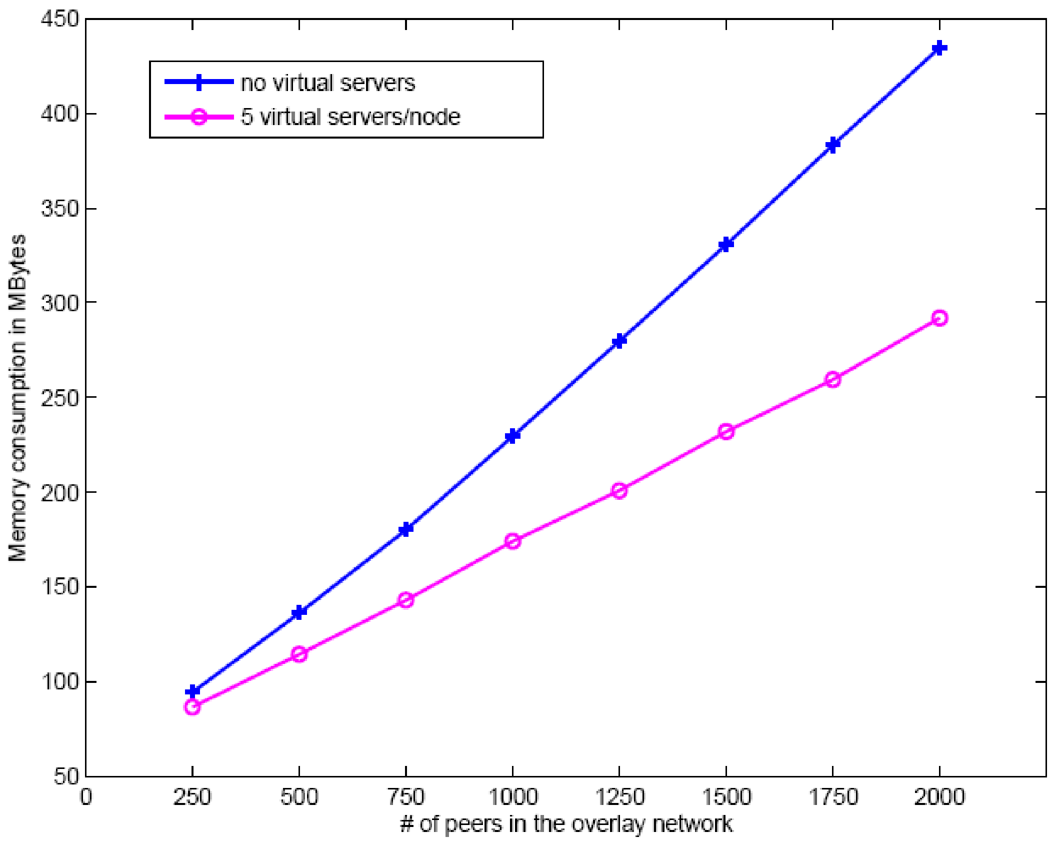
Implementation in OverSim

Internal view of VS



Load balancing container

Evaluation



- VS causes overhead of 28 KB per terminal
- Can be compensated in INET model
  - Several VSs share the same underlay components
  - Result: More peers with less memory consumption

Conclusion

- Load balancing with VSs can run in modular container
  - Algorithms in container can be exchanged
  - Only little or no dependencies to application
- Proof-of-concept implementation for OverSim
  - Backward-compatible to existing applications
  - Reasonable memory/execution overhead
- Step to foster modularization in simulation frameworks
  - Crucial for building more complex applications in the future