THE ADVENTURES OF SELFMAN

Information Society Technologies

KTH KOMPICS

Reactive Component Model for Distributed Computing

Kompics components

» are **reactive** / event-driven programming model

» are **concurrent** / readily exploit **multi-core** architectures

» are **decoupled** by **publish-subscribe ports** and **channels**

» can be **composed** out of **encapsulated** subcomponents

» form **dynamically reconfigurable** architectures

» can form flexible **fault supervision hierarchies**

Experiment profiles

» local / distributed **deployment**: 1 peer / OS process

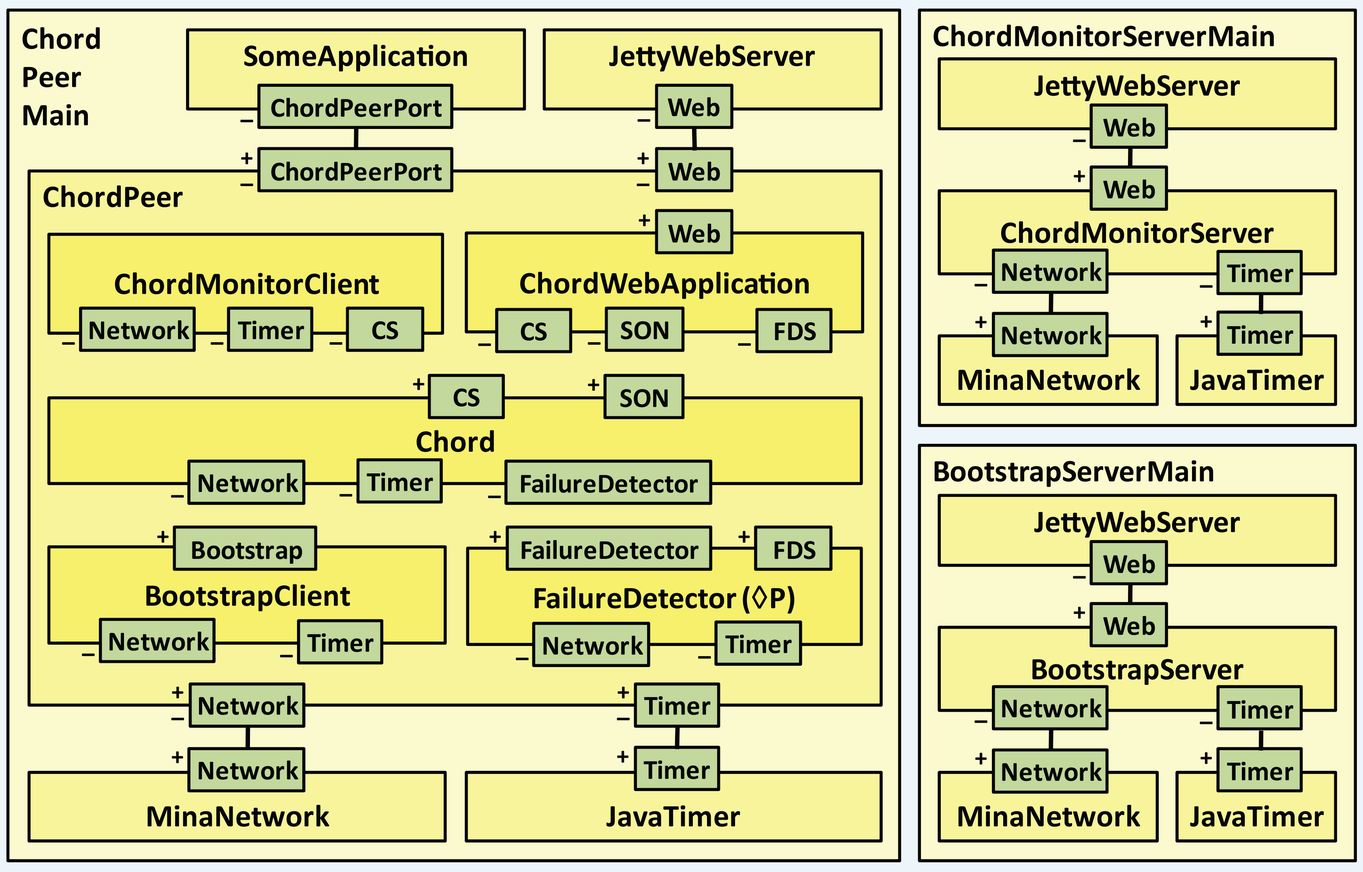
» local / distributed **execution**: multiple peers / OS process

» local **simulation**: multiple peers / OS process

» the deployment code is executable in **simulation** mode
- *using a deterministic single-threaded component scheduler*
- *replay debugging, reproducible results, large experiments*

» the same experiment scenario can be used for local simulation, local execution, or distributed execution

Chord deployment architecture



Peer-to-Peer framework

» reusable **components** and **patterns**
- *failure detection, bootstrap, monitoring*
- *communication, web-based interaction*

» implemented overlay **systems**
- *Chord, Kademlia, Cyclon, T-Man, BitTorrent*

» P2P **experiment scenario** definition **DSL**
- *specify & compose "stochastic processes"*
- *churn, system-specific actions, termination*

» generic **P2P simulator** / orchestrator
- *coupled with system-specific simulators*

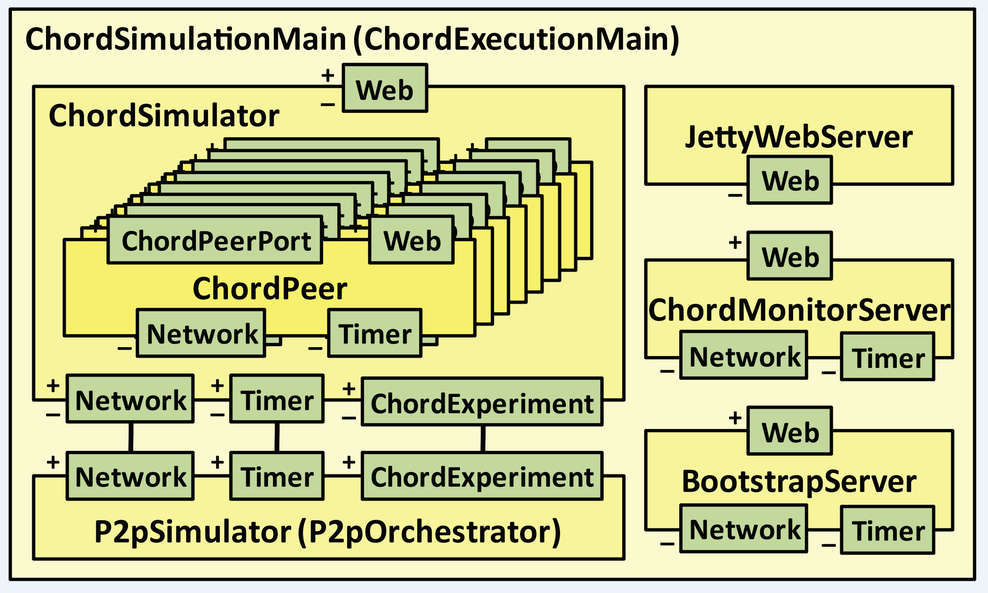
» reusable **latency** and **bandwidth** models

» Java implementation

Chord experiment scenario

```
StochasticProcess boot = new StochasticProcess() {{
  eventInterArrivalTime(exponential(2000));// ~2s
  raise(1000, chordJoin, uniform(16)); }}; // 1000 joins
StochasticProcess churn = new StochasticProcess() {{
  eventInterArrivalTime(exponential(500)); // ~500ms
  raise(500, chordJoin, uniform(16));      // 500 joins
  raise(500, chordFail, uniform(16)); }};  // 500 failures
StochasticProcess lookups = new StochasticProcess() {{
  eventInterArrivalTime(normal(50));       // ~50ms
  raise(5000, chordLookup, uniform(16), uniform(14)); }};
boot.start();                                  // start
churn.startAfterTerminationOf(2000, boot); // sequential
lookups.startAfterStartOf(3000, churn);    // in parallel
terminateAfterTerminationOf(1000, lookups);// terminate
```

Chord simulation architecture



Documentation, examples, and source code at http://kompics.sics.se/

KTH VETENSKAP OCH KONST

Distributed Computer Systems Group
Unit for Software and Computer Systems
Information and Communication Technology

Cosmin Arad (cosmin@sics.se)
Jim Dowling (jdowling@sics.se)
Seif Haridi (seif@sics.se)

Computer Systems Laboratory SICS