Building and Evaluating P2P Systems using the Kompics Component Framework

Cosmin Arad  
Royal Institute of Technology (KTH)  
icarad@kth.se  

Jim Dowling, Seif Haridi  
Swedish Institute of Computer Science (SICS)  
{jdowling,seif}@sics.se  

Abstract—We present a framework for building and evaluating P2P systems in simulation, local execution, and distributed deployment. Such uniform system evaluations increase confidence in the obtained results. We briefly introduce the Kompics component model and its P2P framework. We describe the component architecture of a Kompics P2P system and show how to define experiment scenarios for large dynamic systems. The same experiments are conducted in reproducible simulation, in real-time execution on a single machine, and distributed over a local cluster or a wide area network.

This demonstration shows the component oriented design and the evaluation of two P2P systems implemented in Kompics: Chord and Cyclon. We simulate the systems and then we execute them in real time. During real-time execution we monitor the dynamic behavior of the systems and interact with them through their web-based interfaces. We demonstrate how component-oriented design enables seamless switching between alternative protocols.

Keywords—peer-to-peer; evaluation; component framework; design; simulation; experimentation; deployment.

I. INTRODUCTION

Comprehensive evaluation of P2P systems comprises analysis, simulation, and live performance measurements. We present Kompics [1], a model for building reconfigurable distributed systems from event-driven components. Kompics systems can be uniformly evaluated in large-scale reproducible simulation and distributed deployment, using both the same system code and the same experiment scenarios.


II. KOMPICS AND THE P2P COMPONENT FRAMEWORK

Kompics is a component model targeted at building distributed systems by composing protocols programmed as event-driven components. Kompics components are reactive state machines that are executed concurrently by a set of workers. Components communicate by passing data-carrying typed events through typed bidirectional ports connected by channels. Ports are event-based component interfaces. A port type represents a service or a protocol abstraction. It specifies the types of events sent through the port in each direction. A component either provides (+) or requires (−) a port. Components may encapsulate subcomponents.

The Kompics runtime supports pluggable component schedulers. The default scheduler is multi-threaded and executes components in parallel on multi-core machines. We use a single-threaded scheduler for reproducible simulation.

We developed a set of utility components and methodology for building and evaluating P2P systems. Service abstractions for network and timers can be provided by different component implementations. The framework contains reusable components that provide bootstrap and failure detection services. System-specific components are developed for global system monitoring and web-based interaction. We highlight the elements of the P2P framework in the architecture of our Chord implementation illustrated in Figure 1.

III. DEFINING AN EXPERIMENT SCENARIO

We designed a Java domain-specific language (DSL) for expressing experiment scenarios for P2P systems. We call a stochastic process, a finite random sequence of events, with a specified inter-arrival time distribution. Here is an example scenario composed of 3 stochastic processes:
1000 peers join in a space of $0.2^{16}$. The inter-arrival time between 2 consecutive joins is exponentially distributed with a mean of 2s. A churn process starts 2s after every 500ms on average (exp), a new peer joins or an existing peer fails. In parallel with the churn process, 5000 lookups are initiated uniformly around the ring ($0.2^{16}$) for keys in the first ring quadrant ($0..2^{16}$). The experiment terminates 1s after lookups are done.

IV. EXPERIMENT PROFILES

We can reuse the same experiment scenario to drive simulation or local real-time execution experiments, as well as remote experiments where the system nodes are distributed over the machines of a cluster (possibly running ModelNet [7]) or a testbed like PlanetLab [8] or Emulab [9].

During simulation and local execution (see Figure 2) we model the network at the message-level. In simulation, we execute the same system code built for deployment. Calls for the current system time are trapped and the current simulation time is returned. Simulation enables deterministic replay, debugging, reproducible results, and large-scale experiments without loss of accuracy.

We developed an infrastructure for deploying and executing distributed experiments. Experiment scenarios are locally interpreted by a Master component which coordinates a set of remote Slaves. Each Slave resides on a machine available for the experiment and it manages a set of system peers.

V. DEMONSTRATION OVERVIEW

This demonstration consists of evaluations of two P2P systems developed in Kompics: Chord [10] and Cyclon [11]. Each system is first evaluated in a reproducible simulation experiment. We reuse the same experiment scenario to execute the systems in real time. We observe the dynamic behavior of the systems though the web interface of the monitoring server, which aggregates the global system state periodically. We also inspect the local state of a few system nodes though their web interfaces and we interact with Chord by manually issuing lookups from different nodes.

We reuse the same scenario definition to drive a distributed experiment where nodes are deployed remotely on some cluster machines or on PlanetLab [8]. We repeat some of the previous system interactions. This illustrates the uniform experience of evaluating real systems across simulation, local execution, and distributed deployment.

We use a BitTorrent [12] system developed in Kompics, in a simulation experiment, to demonstrate a realistic bandwidth emulation model. Finally, we return to local execution to experiment live with different scenario definitions.

VI. SUMMARY

We briefly introduced the Kompics component model and we described the component architecture of the Chord overlay developed using the Kompics P2P framework. We showed how to define experiment scenarios for large and dynamic systems and how the same experiments are conducted in reproducible simulation, in real-time execution on a single machine, and distributed over a local cluster or a wide area network.

The source code used for this demonstration, including the Kompics runtime, the P2P framework, experiment scenarios, and implementations of Chord, Cyclon, and BitTorrent, is available online at http://kompics.sics.se.

REFERENCES