Using a Global Event Model in Distributed Resource Management Algorithms

PhD Thesis Research Proposal Summary

Adam Rifkin, `adam.at.`

In this thesis, we will examine event-oriented algorithms for distributed resource management, and compare our solutions to existing approaches. Since we can demonstrate the isomorphism between events, messages, and remote procedure calls, we expect our results to be widely applicable.

RESEARCH AGENDA

Local event models lack several mechanisms useful to developers when components can be globally distributed and dynamically reconfigurable: facilities for handling component and communication channel faults and for dealing with exceptional behavior such as excessive communication latencies; filtering facilities based on predicates that can include soft real-time constraints (because decentralized clockscannot guarantee global accuracy); and composition facilities for using technologies such as multicast and techniques such as event notifier aggregation. We intend to analyze and to understand the event-oriented paradigm in general, and its composing, filtering, and scaling properties in particular.

There is a tension between scalability and availability of the components participating in event-oriented algorithms. In some cases, resource consumers poll providers with announcements in some order until the request is satisfied. In other cases, resource consumers merely announce a call for service, and collaborating providers take responsibility for listening and assembling a basket of resources; or, middlemen take responsibility for listening and assembling a basket of resources, by collating resource offers from competing providers. In all cases, widening the announcement scope increases the pool of available providers, at the expense of increased wide-area message traffic.

Our overall algorithm design goal is to strike a balance between scalability and availability of the providers, requestors, and resources; to that end, we will employ techniques such as aggregating, hierarchical structuring, and scoping. We will compare and contrast our solutions with traditional methods for data consistency (for example, two-phase commit from the database and the distributed shared memory communities) and for hot-spot reduction (for example, soft state from the multicast community and caching from the Web client and randomized algorithm communities).

Several applications can benefit from our analysis, including transaction services, automatic intermediation, and metacomputing. We will design algorithms that exploit event-oriented development, and reason about them using probabilistic models and temporal logic. Then we will implement and evaluate selected algorithms, comparing them with traditional approaches based on performance metrics such as: average response time to receive resources, resilience to faults, bandwidth utilization, and real-time guarantees.

RESEARCH SCHEDULE

- Identify an interesting set of applications representing the problem space
- Perform a requirements analysis and define proper performance metrics
- Formalize these requirements and propose a hypotheses of expected findings
- Propose an experimental design, including how to collect and compare data
- Solve these problems using existing tools, and develop event-oriented alternatives that are demonstrate composability (by clean architecture and component reuse) and high-confidence (by correctness proofs)
- Implement selected algorithms and evaluate them based on metrics