COAST: An Architectural Style for Decentralized On-Demand Tailored Services

Michael M. Gorlick, Kyle Strasser and Richard N. Taylor
Institute for Software Research
University of California, Irvine

WICS/ECSA 2012
Helsinki, Finland
24 August 2012
Context: Decentralized Computation

- Distributed computation among multiple spheres of authority
  - Disaster response *(Hurricane Katrina, New Orleans, August 2005)*
    - National, regional, state, local, NGOs, volunteers
  - Large-scale engineering
    - Boeing 787 Dreamliner or Airbus 350 XWB
  - Scientific computing
    - Bioinformatics (computational genomics or proteomics)
  - Weather forecasting
    - Many sensor networks
    - Many models
  - Computational health care
    - Data-intensive personalized medicine *(The Atlantic, July/August 2012)*
  - Logistics
  - Just-in-time manufacturing

Simultaneous increase in both diversity and integration
Decentralized Computation: Many Paths

- Mastery of data exchange, RPC/RMI, and client-side scripting dominates decentralized applications
  - MapReduce, Hadoop, Piccolo (Power & Li, “Piccolo: Building Fast, Distributed Programs with Partitioned Tables,” OSDI, 2010)
  - CORBA (RPC), Java (RMI), Erlang (message-passing)

- Our approach to decentralized computation has evolved
Goals and Means

• Internet-scale decentralized applications
  – Adaptivity
  – Flexibility
  – Agility
  – Safety

• Secure communications and information
• Protect host computing resources
• Defined valued organizational assets
  – Data bases, sensors, algorithms, users

• Means
  – Stylistic rules
  – Bound behavior of mobile code with architecture-centric mechanisms
  • Principle of Least Authority (POLA)
  • Capability-based security
  – Safety through mobile code
Decentralized Computation: A Different Approach

- Exchange **active** computations among peers
  - *Code + run-time state (reified as closures and continuations)*
- Novel security mechanism: Capability URL (**CURL**)
  - *Dictates where computations may go and how they communicate*
  - *Bounds what visiting computations can do*
  - *Limits resource consumption of computations*
  - *Enforces complex constraints*
- Architectural style: COnputAtional State Transfer (**COAST**)
  - *Build capability security into the architectural style*
    - **Functional capability**
      - *What can a visiting computation do?*
    - **Communication capability**
      - *With whom may that computation communicate?*
      - *When may that computation communicate?*
      - *How often may that computation communicate?*

Architectural style can induce application security
COAST Design Intuitions

- **Computations**
  - *Factor your application into many collaborating computations*
  - *Computations are cheap*
  - *Move computations to assets: processors, data, bandwidth, sensors ...*
  - *Computations isolated from one another except by message-passing*

- **CURLs**
  - *Convey the right to communicate*
  - *Can not be guessed or forged and are tamper-proof*
  - *Carry limitations (time-limited offers, single-use, non-delegable, ...)*
  - *Revocable by issuer at any time*
  - *Critical to the COAST security model*

- **Challenge problem**
  - *Soft real-time video distribution*
    - Many cameras to many consumers
    - Video sharing and manipulation
COASTcast: A Real-time Video Distribution Application

Animation #1: Video from camera to display
COASTcast The Movie: Two Separate Video Flows

A

B

previews

current view
COAST: The Architectural Style

- Applications are comprised of *computations* whose sole means of interaction is the *asynchronous messaging* of *closures*, *continuations*, and *binding environments*
- All computations execute within the confines of some execution site

- Computations are named by Capability URLs (CURLs)
  - *Computation* $x$ *may deliver a message to computation* $y$ *only if* $x$ *holds a CURL* $u$ *of* $y$
  - *The interpretation of a message* $m$ *delivered to computation* $y$ *via CURL* $u$ *of* $y$ *is* $u$-*dependent*
COASTcast: A Real-time Video Distribution Application

Animation #2: Sharing Video
COASTcast The Movie: Sharing a Video
COASTcast: Moving a Video Source

Camera Island A
- Encoder 1
- Relay 1
  - Relay 2
  - D2
  - D4
  - Encoder 2
- D3

Display Island A
- Canvas 1
- Canvas 4
- User Interface 1
- Decoder 1
- C1
- U1
- R1
- D1
- Relay 1
- R1
- U1

Camera Island B
- Canvas 3
- Canvas 2
- User Interface 2
- Decoder 2
- C2
- U2
- R2
- Decoder 3
- C3
- U2
- D4
- Relay 2
- R2

Display Island B
- Canvas 1
- User Interface 1
- Decoder 4
- C4
- U1
- R1
- D1
- Relay 1
- R1
- U1

COASTcast: Moving a Video Source

Diagram showing the flow of video sources and interfaces:

- **Camera Island A**: Encoder 2, Relay 2, Decoder 1, R1, C1, U1, Canvas 1, User Interface 1
- **Display Island A**: Canvas 4, User Interface 1
- **Camera Island C**: Encoder 3, Relay 1, Decoder 4, C3, U2, Canvas 3
- **Display Island B**: Canvas 2, User Interface 2
COASTcast The Movie: Change the Video Source
Related Work

- **Capability Security**
  - *Confused Deputy* (Hardy, “The confused deputy: (or why capabilities might have been invented),” SIGOPS Operating Systems Review, 1988)
  - *Revocation & multi-level security* (Miller & Shapiro, Paradigm regained: Abstraction mechanisms for access control, ASIAN’03, 2003)
  - *Object-Capability and Capability Languages* (Miller, Robust composition: Towards a unified approach to access control and concurrency control, PhD thesis, John Hopkins University, 2006)
Future Work/Summary

• Future Work
  – Digital contract negotiation (Alegria Baquero)
  – Collaboration architectures for disaster response (Christoph Dorn)
  – Dynamic software update
  – Electronic health systems (emphasis on security and privacy)
  – Adaptive robotics

• Summary
  – Results suggest COAST is a step forward for decentralized applications
  – Expressive (enough), efficient (enough) and secure (enough) for a variety of domains
  – CURLs essential to robust COAST security
  – Mobile code is manageable given the tools of functional and communication capability
  – Architectural style can make significant contributions to application security