Salamander: A Push-based Distribution Substrate for Internet Applications

G. Robert Malan, Farnam Jahanian and Sushila Subramanian

University of Michigan

Presented by Alexander Vasserman
Outline

• Introduction
• Application Domains
• Architecture
  – Channel Subscription Services
  – Application Level Quality of Service
  – Lightweight Temporal Database
• Performance Evaluation
• Conclusion
Introduction

• Platform for push-based applications
• Persistent data queries
  – Match both existent and new data
• Attribute-based data space
  – Opaque data objects with text attribute lists
• Virtual distribution channels
  – Application specific distribution topology for publish/subscribe clients
Design Features

• Application specific plug-in modules
  – data modification and delivery decisions
  – provide mechanism for application QoS policies
  – handle large variation in available resources

• Two-tier cache

• Support heterogeneous set of clients
  – Perl, C, Java interfaces
Application Types

• Groupware
  – human-to-human communication
  – shared software and workspaces
  – synchronized access to data sources
  – remote access and control of instruments

• Webcasting
Main challenge: scalable dissemination
The Upper Atmospheric Research Collaboratory

• Goals
  – collaboratory environment for geographically dispersed research communities
  – real time experiments at remote facilities

• Tools
  – shared synchronized displays
  – multiparty chat boxes
  – shared annotation database
  – distributed text editor
Internet Performance Measure and Analysis

• Goals
  – Real-time measurement, analysis, dissemination, and visualization of performance and routing

• Tools:
  – ASExplorer
    • explore routing topology, growth and instabilities
  – NetNow
    • measure network loss and latency
Channel Subscription Service

• Anonymous push – dissemination
  – Publish opaque data objects with text attribute lists
  – Subscribe using attribute expressions as persistent queries
  – Attributes used to look up destination

• Negotiated push – feedback
  – Lookup endpoint identifier using attribute list
  – Unicast messages
  – Request data, turn data flows on and off, etc.
Application Level Quality of Service

• Application-specific plug-ins at each node
  – QoS parameters defined by user
  – Fine-grained drop policies
  – Data degradation and conversion/compression
  – Multiple format data for different clients

• UARC:
  – bottleneck at client CPU's, not bandwidth
Lightweight Temporal Database

- Mix between temporal and relational DB
- Store virtual channel's data based on time
  - write once updates
- Retrieve based on time ranges and attributes
- Memory cache for in-transit data
Performance Evaluation

- Exponential latency in # receivers?
- Savings from common data?
Conclusions

• Introduce push based architecture for web dissemination and collaboration
  – channel subscription abstraction
  – combination of temporal and relational db
  – application specific plug-ins at all nodes
  – anonymous and negotiated push