PADRES: A Middleware for the Decentralized Execution of Business Processes

Hans-Arno Jacobsen
Middleware Systems Research Group
University of Toronto

www.msrg.utoronto.ca

www.msrg.utoronto.ca/padres (coming soon)
Acknowledgements

2006-… Supported by:

Research Team:
- Hans-Arno Jacobsen (PI)
- Alex Cheung
- Guoli Li
- Vinod Muthusamy
- Songlin Hu (Visitor)
- Pengcheng Wan
- Alex Wun
- Serge Mankovski (Cybermation Inc.)

Alumni:
- Eli Fidler, Ferdous Jewel, David Matheson,
- Gerald Chan, Matt Medland

2003-2005 Supported by:

- Ontario Centres of Excellence
- Cybermation
- Sun microsystems
- NSERC
- Cybermation
Outline

• A few Examples Motivating Pub/Sub
• Publish/Subscribe Overview
• The PADRES Publish/Subscribe System
• Business Processes in PADRES
• Security in PADRES
The Name

• First generation of students, when I looked away
  – Peng Alex David aRno Eli Serge

• PAAdres is Distributed RESource Scheduling

• Publish/subscribe Applied to Distributed Resource Scheduling
Motivation for
Publish/Subscribe
per se
Your search – Feuerwehrdrehleiteroperationshandbuch – did not match any documents. No pages were found containing "feuerwehrdrehleiteroperationshandbuch".

Suggestions:

- Make sure all words are spelled correctly.
- Try different keywords.
- Try more general keywords.
Amazon to Chapters to you ....

Monday, October 10th in Cyberspace

Thursday, November 15th, in Toronto

Your book “...” is available at .... $10 off
Business Process Execution

Business Process
- Receive
- Assign
- Flow
- Pick
- Invoke
- Scope
- Wait
- Reply

Business Process
- Receive
- Switch
- Scope
- Reply

Client
Agent
Broker
Broker
Broker
Broker
Database
WS
WS

METIS Security Seminar Series, March, 2006
Publish/Subscribe
Publish/Subscribe

Publisher

Publisher

Broker(s)

Notification

Subscriptions

Notification

Subscriptions

Subscriber

Subscriber

Stock markets

NYSE

TSX

NASDAQ

IB

AMGN=58

ORCL=12

HON=24

MSFT=27

JNJ=58

IBM=84

INTC=19

NYSE

NASDAQ

Subscriptions:
IBM > 85
ORCL < 10
JNJ > 60
Publish/Subscribe Benefits

• Decoupling
  – Space (physical distribution)
  – Location (clients do not need references to each other)
  – Time (clients do not need to be up at the same time)
  – Representation (different message formats)

• Other paradigms: request/response, messaging, shared memory, ...
That's Like Data Base Querying 😐 !!

Query and subscription is very similar. Set of tuples and publication is very similar.

However, the two problem statements are inverse.

METIS Security Seminar Series, March, 2006
Publish/Subscribe Matching Problem

• Given a set of subscriptions, $S$, and a publication, $e$, return all $s$ in $S$ matched by $e$.
• $e$ is referred to as event or publication
• Splitting hairs
  – Event is a state transition of interest in the environment
  – Publication is the information about $e$ submitted to the publish/subscribe system
• Simple problem statement, widely applicable, and lots of open questions
Problem Instantiations

- Text / search strings (information filtering)
- Semi-structured data / queries
  - attribute-value pairs / attribute-operator-value-predicates
  - XML, HTML
- Tree-structured data / path expressions
  - XML ./ XPath expressions
- Graph-structured data / graph queries
  - RDF / RDF queries (e.g., SPARQL)
- Regular languages / regular expressions
- Centralized and **distributed** instantiation
- Different matching semantics (e.g., **crisp**, approximate, similar, n-of-m, ...)

Subscriptions
Challenges

• Lot’s of subscriptions
• High publication rate
• High subscription update rate
• Different data formats
The Content-based Model

• Language and Data model
  – Conjunctive Boolean functions over predicates
  – Predicates are attribute-operator-value triples
    • \([\text{class,eq,trigger}]\)
  – Subscriptions are conjunctions of predicates
    • \([\text{class,eq,trigger}],[\text{appl,eq,payroll}],[\text{gid,eq,g001}]\)
  – Publications are sets of attribute-value pairs
    • \([\text{class,trigger}],[\text{appl,printer}],[\text{gid,g007}]\)

• Matching semantic
  – A subscription matches if all its predicates are matched
Distributed Publish/Subscribe

- A.k.a. **content-based routing**
- All interactions are based on publish and subscribe
- **No address information is exposed or available to clients & brokers**
Publish/Subscribe in Industry

- Standards
  - CORBA Event Service
  - CORBA Notification Service
  - OMG Data Dissemination Service
  - Java Messaging Service
  - WS Eventing
  - WS Notification (Draft)

- Emerging technologies
  - RSS aggregators
    - PubSub.com, FeedTree
  - Real-time data dissemination
    - TIBCO, RTI Inc., Mantara Software
  - Application integration
    - Softwired
  - Hardware-based brokers
    - Sarvega (Intel), Solace Systems, DataPower (IBM)
Publish/Subscribe in Academia

• Research projects
  – Gryphon (IBM)
  – Hermes (Cambridge)
  – SIENA (Boulder)
  – REBECA (Darmstadt)
  – ToPSS (UofT)
  – PADRES (UofT)

• Classification of Pub/Sub
  – Channel
  – Topic
  – Content
  – Subject space

Channel-based
nytimes.com RSS

Content-based
nytimes.com RSS
type = editorial
author = Safire

nytimes.com
Pub/Sub Research Directions

- Matching algorithms
  - Language expressiveness, scalability, speed
- Routing protocols
  - Network architectures, scalability
- Higher level abstractions
  - Workflow execution
  - Monitoring

<table>
<thead>
<tr>
<th>TopSS</th>
<th>A-TopSS</th>
<th>CS-TopSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(matching)</td>
<td>(approximate)</td>
<td>(composite subs)</td>
</tr>
<tr>
<td>S-TopSS</td>
<td>L-TopSS</td>
<td>Rb-TopSS</td>
</tr>
<tr>
<td>(semantic)</td>
<td>(location-based)</td>
<td>(rule-based)</td>
</tr>
<tr>
<td>X-TopSS</td>
<td>persistent-ToPSS</td>
<td></td>
</tr>
<tr>
<td>(XML matching)</td>
<td>(subject spaces)</td>
<td></td>
</tr>
<tr>
<td>M-TopSS</td>
<td>P2P-TopSS</td>
<td>LB-TopSS</td>
</tr>
<tr>
<td>(mobile)</td>
<td>(peer-to-peer)</td>
<td>(load balancing)</td>
</tr>
<tr>
<td>Federated-ToPSS</td>
<td>Ad hoc-ToPSS</td>
<td></td>
</tr>
<tr>
<td>(federation of TopSS brokers)</td>
<td>(ad hoc networking)</td>
<td></td>
</tr>
<tr>
<td>Historic-ToPSS</td>
<td>FT-ToPSS</td>
<td></td>
</tr>
<tr>
<td>(historic data)</td>
<td>(fault tolerance)</td>
<td></td>
</tr>
<tr>
<td>JS-TopSS</td>
<td>BPEL-ToPSS</td>
<td></td>
</tr>
<tr>
<td>(job scheduling)</td>
<td>(BPEL execution)</td>
<td></td>
</tr>
</tbody>
</table>
Applications Enabled by Pub/Sub

- Selective information dissemination
- Location-based services
- Personalization
- Alerting services
- Application integration
- Job scheduling
- Monitoring, surveillance, and control
- Network and distributed system management
- Workforce management
- (Scientific) workload management
- Business activity monitoring
- Business process management, monitoring, and execution
Modeling the Motivating Examples

• Google example
  – Search string is the subscription
  – Pages continuously indexed by Google are publications
  – A match identifies new information found
  – Requires high-performance centralized matching engine

• Amazon / Chapters
  – Book looked-up on Amazon is the subscription
  – Current location coordinates and items on sale at store are the publications
  – A match identifies that the user is close to a book store that has the sought book on sale
  – Requires a distributed publish/subscribe infrastructure

• Business process execution (see later)
The PADRES System
PADRES Project Overview

- Collaborative R&D project (2003-2005)
- Part of ToPSS Family (Toronto Publish/Subscribe System)
- A publish/subscribe system built on a peer-to-peer overlay network model (i.e., at application-level)
- The goal is to advance publish/subscribe research as well as provide a flexible messaging substrate for decentralized workflow management
- Investigate the hybridisation of publish/subscribe and query-based data access in distributed environments
- Experiment with emerging applications, such as business activity monitoring and business process execution (i.e., workflow management)
- In addition to standard publish/subscribe semantics, PADRES supports a large range of original ideas
PADRES Architecture Overview

• PADRES consists of 2 major components
  – Brokers
    • Forward messages using overlay network
    • Provide bindings as client connection points
  – Clients
    • Publish, Subscribe, Advertise
Broker Architecture

- BrokerCore
  - InputQueue
  - OutputQueues
  - Matching Engine
    - Publication / Subscription Routing Table
    - JESS
  - Controller
    - Lifecycle Manager
    - Overlay Manager
  - QueueHandler
  - RMITransport Handler
  - Broker_Control Message
  - QueueHandler
  - QueueHandler
  - ClientRMI
  - JMS
  - DB
  - BrokerRMI
Publish and Subscribe Cycle

• Publishers come alive and submit
  – *Advertisements* as indication of the type of information they may provide in the future
    • Advs. are flooded
    • Advs. are an optimization (not absolutely required)

• Subscribers submit
  – *Subscriptions* as indication of interest specifications
    • Subscriptions propagate towards source of advs. to establish **publication-routing paths** in the network

• Publisher publish concrete information
  – Publications propagate along **publication-routing path** toward interested subscribers

• # Advs < # Subs < # Pubs
Content-based Routing: Advertising

Distributed Overlay Broker Network

Publisher

Subscriber

PADRES

Broker

Broker

Broker

Broker

Advertisement
Content-based Routing: Subscribing

Broker Network

Distributed Overlay

Publisher

Broker

Broker

Broker

PADRES

Subscriber

Subscription

Publisher

Broker

Broker

Broker

Broker

METIS Security Seminar Series, March, 2006
Content-based Routing: Publishing

Distributed Overlay Broker Network

Publication

Publisher

Broker

Broker

PADRES

Broker

Broker

Publisher

Subscriber

Subscriber
Unique PADRES Features

- Rule-based matching engine for routing decisions (Rete-based matcher)
- Historic data access in publish/subscribe layer
- Composite subscription and composite event detection
- Meta events and subscriptions
- Failure detection in publish/subscribe layer
- Load-balancing (high-volume subs. & pubs.)
- Workflow management application support
Composite Subscription

• Composite subscription consists of atomic subscriptions
  – Provide a higher level view for subscribers (e.g., for event aggregation)
  – Here used to express flow dependencies

• Subscription language features
  – Operators (AND, OR) and variables ($x$)

• Seamlessly supported by Rete for centralized matching but require extensions to content-based routing for distributed matching
Composite Subscription Example

Composite event is the constellation of events being detected by the composite subscription.

CS=\{(S1 \text{ OR } S2) \text{ AND } (S3 \text{ OR } S4) \text{ AND } S5\}
Composite Subscription Routing

CS = \{(S1 \text{ AND } S2) \text{ AND } S3\}

CS' = \{S1 \text{ AND } S2\}

Publishers

Subscribers

Distributed Overlay Broker Network

CS

S

P

S

S1

S2

S3

P1

P2

P3

B1

B2

B3

B4

B5

B6

B7

B8
Composite Event Detection

CS = \{\{S_1 \text{ AND } S_2\} \text{ AND } S_3\}

CS' = \{S_1 \text{ AND } S_2\}

P Publishers
S Subscribers

Distributed Overlay Broker Network

CS, CS'

P1, P2, P3, P12, P123

S1, S2, S3, S
Decentralized Business Process Execution in PADRES

We use the terms business process and workflow synonymously.
Decentralized Workflow Execution

- Natural composition of distributed resources
- A centralized execution may constitute a bottleneck
- A centralized execution may introduce unnecessary message load
- Avoid single point of failure
Workflow Descriptions

- Workflows are described using XML-based languages, such as BPEL
- Activities defined in a workflow depend on each other
- A workflow instance is generated by a trigger
- Failure handing is part of the workflow description
Workflow Transformation

- The first job(s) subscribe to trigger messages
  
  Job A: \{[\text{class,}=,\text{trigger}], [\text{workflow,}=,\text{payroll}], [\text{instanceID,}=,$x$]\}

- Job dependencies are modeled by subscriptions

  Job D: 
  \{
  \{[\text{class,}=,\text{job_info}], [\text{workflow,}=,\text{payroll}], [\text{instanceID,}=,$x$],[\text{job,}=,B],[\text{status,}=,\text{succ}]\}
  \AND
  \{[\text{class,}=,\text{job_info}], [\text{workflow,}=,\text{payroll}], [\text{instanceID,}=,$x$],[\text{job,}=,C],[\text{status,}=,\text{succ}]\}
  \}

  Job E: \{[\text{class,}=,\text{job_info}], [\text{workflow,}=,\text{payroll}], [\text{instanceID,}=,$x$],[\text{job,}=,A],[\text{status,}=,\text{fail}]\}

- A workflow instance is generated by a trigger publication

  Trigger: \{[\text{class,}=,\text{trigger}], [\text{workflow,}=,\text{payroll}], [\text{instanceID,}=,10001]\}
Workflow Execution (Success)
Workflow Execution (Failure)
Summary

• Publish/Subscribe is a widely applicable paradigm
• Content-based pub/sub is efficiently possible for a large variety of languages and data models
• Pub/Sub is not like data base querying
• PADRES is a distributed, content-based publish/subscribe system
• PADRES targets decentralized workflow execution
• All PADRES interaction patterns are entirely publish and subscribe
  – Deployment
  – Execution
  – Failure detection
  – Monitoring
  – ...

Encrypted Content-based Routing in PADRES

Security for Publish/Subscribe Networks
Background: Content-Based Routing

Advertisement: [class, sensor_reading], [temp, >, 20], [light_level, <=, 9]

Subscription: [class, sensor_reading], [temp, >, 35], [light_level, <, 6]

Publication: [class, sensor_reading], [temp, 40], [light_level, 3]
PADRES: Message Encryption

• Observe that plain-text string representation of messages is effectively a *Unique ID*

• Encrypted representation of matched message is tunnelled in regular predicate as *Routing Token*

• Encryption/Decryption occurs only at locally *Trusted Brokers*
  – End-to-End security association
  – Transparent to clients
Encrypted Content-Based Routing

Advertisement: \([\text{class, sensor\_reading}], [\text{temp, >}, 20], [\text{light\_level, <=}, 9] \rightarrow E(\text{adv})\)

Subscription: \([\text{token}, E(\text{adv})], [\text{data}, E([\text{class, sensor\_reading}], [\text{temp, >}, 35], [\text{light\_level, <}, 8])]\)

Publication: \([\text{token}, E(\text{sub})], [\text{data}, E([\text{temp, 40}], [\text{light\_level, 3}])]\)
-- The End, Thank you --
ToPSS - Toronto Publish/Subscribe System

Information consumers subscribe to information of interest. Information producers publish information. ToPSS-broker(s) match and route relevant information to interested subscribers.

2001 - present

- G-/S-ToPSS (semantic)
- X-ToPSS (semi-structured data: XML)
- A-ToPSS (approximate)
- M-ToPSS (mobile)
- L-ToPSS (location-based & correlation)
- p2p-ToPSS (peer-to-peer)
- Rb-ToPSS (rule-based)
- persistent-ToPSS (Subject Spaces)
- Federated-ToPSS (federation of ToPSS brokers)

VLDB’02, ICDE’04, ICDE’01/Tutorial
VLDB’04
VLDB’03, VLDB/SEM’03
WWW’05
VLDB/TES’03, VLDB’04, MDM’05
VLDB/DBISP2P’03
MobiQuitous’05
MobiCom’05
IEEE MDM’04, DEBS’05, MobiCom’05
DEBS’02
VLDB’02, ICDE’04
ICDE’01/Tutorial
M.A.Sc. Thesis 3/04
M.A.Sc. Thesis 1/04
CASCON’02’03
ICFI’05
2001 - present

METIS Security Seminar Series, March, 2006
Broker Protocol Stack

- Business Process
- Representation
- Publish/Subscribe
- Overlay
- Transport

- BPEL
- XML
- PRT & SRT
- ORT
- Java RMI
Rule-based Matching

- Based on JESS (Java Expert System Shell)
- Enables a powerful subscription language
  - Variables, join conditions, composite subscriptions

\[ S = [\text{class}, \text{eq}, \text{trigger}], [\text{appl}, \text{eq}, \text{payroll}], [\text{gid}, \text{eq}, \text{g001}] \]

\{
  \text{Rule} \ S \\
  \text{(trigger (appl } ?x:(\text{eq } ?x \text{ “payroll”}) (gid } ?y:(\text{eq } ?y \text{ “g001”})) \\
  \Rightarrow (\text{send to forwarding targets}) \\
\}

- Publications become facts
Rule-based Matching Performance

![Graph showing matching time vs number of subscriptions for different algorithms: Naive Matching Algorithm, Predicate Counting, JESS.](image)
Routing Tables

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Handled By</th>
<th>Modifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertisement</td>
<td>ORT</td>
<td>SRT</td>
</tr>
<tr>
<td>Subscription</td>
<td>SRT</td>
<td>PRT</td>
</tr>
<tr>
<td>Publication</td>
<td>PRT</td>
<td>None</td>
</tr>
</tbody>
</table>

- The SRT and PRT are represented by a Rete, respectively.
- The Rete is formed by rules with subscriptions on the LHS and forwarding targets on the RHS.
- The actions are a list of next-hop addresses to send a matching message to.
Matching Algorithms

- **Counting algorithm**
  - Counts satisfied predicates per subscription
  - Subscription is matched if all its predicates are matched

- **Rete-algorithm (Forgy, 1979)**
  - Compiled network of nodes representing production rule left hand sides

- **Gryphon algorithm (Arguriella et al., 1999)**
  - Subscriptions are represented as a tree

- **Predicate Clustering (Pereira, Jacobsen et al., 2001)**
  - Search space is pruned through access predicates and subscription clusters
  - A cluster is disregarded, if its access predicate(s) are false
Historic Query Examples
Historic Data Access

- Publications are stored in databases distributed throughout the broker network.
- Historic subscriptions allow the clients to retrieve past publications in conjunction with future publications.
- Publications can be temporally joined using composite subscriptions.
Historic Data Access

- **Simple queries:**
  - \([\text{class, eq, trigger]} [\text{appl, eq, payroll}]
    \[\text{gid, =, } \$x\]
    \[\text{time, <, now+1hr}] [\text{time, >, now-1hr}]\)

- **Complex Queries:**
  - \([\text{class, eq, job\_status}] [\text{appl, eq, } \$y] [\text{gid, =, } \$x]\)
    AND
    \[[\text{class, eq, trigger}] [\text{appl, eq, } \$y] [\text{gid, =, } \$x]\]
    \[\text{time, >, 0000}] [\text{time, <, 0800}]\)
Workflow Evaluation
Workflow Deployment Traffic

- No Composite Subscription
- Decomposition at First Broker
- Distributed Decomposition

Network Traffic (KB)

Workflow A

Workflow B
Evaluations

- **Environment**
  - Intel Xeon 3GHz, 1GB RAM
  - JDK 1.4.2

- **Workload description**
  - Predicates based on 20 attributes with random operators
  - Values are uniformly distributed in the value range
  - Two workflows with different number of jobs

- **Metrics**
  - Matching time
  - Routing delay per composite subscription
  - Network traffic
Publication Matching Time

Routing Time (ms) vs Number of Subscriptions

- Naive Matching Algorithm
- Predicate Counting
- PADRES Broker

METIS Security Seminar Series, March, 2006
Composite Event Detection

Composite Event Detection Time (ms)

Number of Atomic Subscriptions per Composite Subscription

- Blue line: 100 publications
- Red line: 5000 publications
- Brown line: 10000 publications

METIS Security Seminar Series, March, 2006
Routing Delay

Number of Atomic Subscriptions per Composite Subscription

Routing Delay (ms)
Number of Notifications

- Without Composite Subscription
- Composite Subscription

Number of Publications

METIS Security Seminar Series, March, 2006
Workflow Deployment

Network Traffic (KB)

- Without Composite Subscription
- Composite Subscription

Workflow A
Workflow B

METIS Security Seminar Series, March, 2006
Workflow Execution

Network Traffic (KB)

- Without Composite Subscription
- Composite Subscription

Workflow A
Workflow B

METIS Security Seminar Series, March, 2006