A Tour of SIENA, an Interoperability Infrastructure for Internet-Scale Distributed Architectures

David S. Rosenblum
Information & Computer Science
UC Irvine
http://www.ics.uci.edu/~dsr/
Collaborators

♦ Alexander L. Wolf
   (University of Colorado at Boulder)
♦ Antonio Carzaniga
   (University of Colorado at Boulder)

♦ Recent Additional Collaborations
  ▪ Dick Kemmerer and Giovanni Vigna
    (UC Santa Barbara)
  ▪ DARPA’s DASADA Program

http://www.cs.colorado.edu/serl/siena/
Air Travel Reservations Scenario

- low air fares to Europe
- special offers by United
- fly July DEN-MXP
- subscribe
- publish
- notify
- United offers DEN-MXP Sep
Possible Killer Apps

♦ Traffic reports, travel updates
♦ Product updates, recalls
  ▪ software
  ▪ cars
  ▪ food (smart refrigerator)
♦ Internet gaming
♦ Weather reports
♦ Medical notification
♦ Just-in-time everything
SIENA Event Notification Service
Notifications, Filters and Patterns
Interface of SIENA

♦ SIENA:

- publish(notification n)
- subscribe(string identity, pattern p)
- advertise(string identity, filter p)
- unsubscribe(string identity, pattern p)
- unadvertise(string identity, filter p)

♦ Subscriber:

- notify(notification n)
A notification is a list of attributes

attribute = (type, name, value)

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>class</td>
<td>travel/airlines/offers</td>
</tr>
<tr>
<td>date</td>
<td>starts</td>
<td>Jun 99</td>
</tr>
<tr>
<td>date</td>
<td>expires</td>
<td>Aug 99</td>
</tr>
<tr>
<td>string</td>
<td>origin</td>
<td>DEN</td>
</tr>
<tr>
<td>string</td>
<td>destination</td>
<td>MXP</td>
</tr>
<tr>
<td>string</td>
<td>carrier</td>
<td>BA</td>
</tr>
<tr>
<td>float</td>
<td>price</td>
<td>850</td>
</tr>
<tr>
<td>string</td>
<td>currency</td>
<td>USD</td>
</tr>
</tbody>
</table>
Filters

- A subscription is expressed via filters
- A filter is a list of attribute constraints

\[
\text{attribute constraint} = (\text{type, name, operator, value})
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>class</td>
<td>&gt;*</td>
<td>travel/airlines/</td>
</tr>
<tr>
<td>date</td>
<td>starts</td>
<td>&lt;</td>
<td>Jul 99</td>
</tr>
<tr>
<td>date</td>
<td>expires</td>
<td>&gt;</td>
<td>Jul 99</td>
</tr>
<tr>
<td>string</td>
<td>origin</td>
<td>=</td>
<td>DEN</td>
</tr>
<tr>
<td>string</td>
<td>destination</td>
<td>=</td>
<td>MXP</td>
</tr>
</tbody>
</table>
A pattern is an algebraic expression whose basic elements are filters

<table>
<thead>
<tr>
<th>string</th>
<th>class</th>
<th>varchar</th>
<th>travel/airlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>airline</td>
<td>varchar</td>
<td>UA</td>
</tr>
<tr>
<td>string</td>
<td>dest</td>
<td>varchar</td>
<td>MXP</td>
</tr>
<tr>
<td>int</td>
<td>price</td>
<td>int</td>
<td>&lt; 500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th>class</th>
<th>varchar</th>
<th>travel/airlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>airline</td>
<td>varchar</td>
<td>AA</td>
</tr>
<tr>
<td>string</td>
<td>dest</td>
<td>varchar</td>
<td>MXP</td>
</tr>
<tr>
<td>int</td>
<td>price</td>
<td>int</td>
<td>&lt; 400</td>
</tr>
</tbody>
</table>
Distributed Architecture of Siena
Content-Based Routing
A Network of SIENA Routers

- New York
- Boulder
- Milano
- Roma
- Irvine

SIENA routers
Principles

- **Downstream replication**
  - notifications propagated in single copy as far as possible (replicated only close to destinations)

- **Upstream filtering and pattern assembly**
  - notifications are filtered close to their sources

- **Common principles**
  - loop-free paths, efficient forwarding, reduced state, etc.

- **Implemented by two classes of algorithms**
  - subscription forwarding
  - advertisement forwarding
Subscription Forwarding
Subscription Forwarding

- Subscriptions are propagated to every server, leaving state along the path.

- Matching notifications follow (backwards) the paths set by subscriptions.

- Unsubscriptions are also propagated to undo the effect of subscriptions.
Reducing Filter Propagation

♦ Idea:
  - We can avoid propagating a filter where a more generic filter has already been forwarded
Reducing Filter Propagation

♦ Idea:
- We can avoid propagating a filter where a more generic filter has already been forwarded
Summary of SIENA
Advantages and Challenges
Advantages and Challenges

♦ Architectural Advantages
  - Strong decoupling of components
  - Network and address transparency
  - Lightweight messaging
  - Felicitous expressiveness/scalability tradeoff

♦ Architectural Challenges
  - “The ontology problem”: bootstrapping, naming, name registration, name advertisement, etc.
  - Performance
  - Security
  - Wireless
Challenge
Security of Publish/Subscribe Networks
Traditional Security Properties

♦ Authentication
  ▪ Bob wants to make sure Alice sent the message

♦ Confidentiality
  ▪ Alice wants to ensure only Bob can read the message

♦ Integrity
  ▪ Alice wants to ensure Bob receives the message, and exactly the message she sent

♦ Accountability
  ▪ Bob wants to prove only Alice could have sent the message

♦ These are all expressed in terms of explicit addresses (Bob and Alice)!
Implicit addressing requires reformulation of security properties

- Example: Authentication
  - Bob ultimately wants to authenticate content, not sender

- Example: Confidentiality
  - Bob doesn’t want anyone to see his subscriptions, but doesn’t care who sees his notifications

Pub/sub admits new kinds of vulnerabilities

- Example: Denial of Service
  - Highly generic filter ("Price > 0") causes flood of notifications to subscriber
Conclusions

**SIENA**

- Wide-area event notification (pub/sub) service
- Distributed architectures
  - different architectures not shown in this talk
- Content-based routing
  - Algorithms for wide-area event notification
- Expressive data model and selection mechanism
  - Approximates SQL
- Scalability confirmed by simulation studies
  - results not shown in this talk
- C++ and Java prototypes available
Challenge
Content-Based Routing in the Network
Middleware or Network Service?

Today: SIENA as Middleware

Future: SIENA as Network Service
Why Do This?

- Eliminate protocol layers
  - Greater efficiency of routing and forwarding
- TCP/IP performs explicit-address routing
  - Content-based routing involves implicit routing of content based on filters, not addresses
  - SIENA may thus “conflict” with the work of TCP/IP
- Possible Risks
  - Content-based routing is computationally more expensive than explicit-address routing or subject-based routing
  - Current approaches try to make network elements dumber, not smarter
Challenge
Satellite-Based Wireless Event Notification
Why Satellite-Based Wireless?

♦ The routing problem goes away
  - Broadcast rather than point-to-point
  - Notify large numbers of interested parties in a single “hop”

♦ Possible risks
  - Wireless networking is more unreliable than wired networking
  - And wireless devices are resource-deficient
Wireless Event Notification
Wireless Event Notification
Wireless Event Notification
Expressiveness vs. Scalability

- **Expressiveness**
  - flexible notification data modeling
  - notification filtering
  - notification patterns

- **Scalability**
  - vast dimensions (objects, distance, time)
  - heterogeneity (hardware/software platforms)
  - decentralization (administrative domains)
  - resources (latency, bandwidth, reliability)

Expressiveness and scalability are in conflict!
Complexity of Covering Relations

- Notifications and filters are quite expressive
  - Filters offer a significant subset of SQL

- Efficient computation of covering relations
  - Subscription–notification: $O(s+n)$
  - Subscription–subscription: $O(s_1s_2)$

- Example: $[x > k_1]$ covers $[x > k_2] \iff k_2 \geq k_1$

- Limited complexity means better scalability

- A more expressive model (e.g., user-defined operators) may make filter–filter undecidable
## Related Technology

<table>
<thead>
<tr>
<th>Subscription language</th>
<th>architecture</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>centralized</td>
<td>hierarchical</td>
</tr>
<tr>
<td>channel</td>
<td>Field, CORBA ES, Java ES</td>
<td>CORBA ES (IONA)</td>
</tr>
<tr>
<td>subject</td>
<td>ToolTalk</td>
<td>NNTP, TIB-Rv, JEDI</td>
</tr>
<tr>
<td>content</td>
<td>CORBA NS, JMS, Elvin</td>
<td>Keryx, Yu et al.</td>
</tr>
<tr>
<td>content + patterns</td>
<td>Yeast, GEM, active dbases</td>
<td><strong>SIENA</strong></td>
</tr>
</tbody>
</table>
SIENA and IP Multicast

- **SIENA** is conceptually similar to IP multicast

<table>
<thead>
<tr>
<th>IP Multicast</th>
<th>SIENA</th>
</tr>
</thead>
<tbody>
<tr>
<td>joinGroup(host,group)</td>
<td>subscribe(ID,expr)</td>
</tr>
<tr>
<td>leaveGroup(host,group)</td>
<td>unsubscribe(ID,expr)</td>
</tr>
<tr>
<td>ip_send(datagram)</td>
<td>publish(notification)</td>
</tr>
</tbody>
</table>

host ↔ ID?
datagram ↔ notification?
group ↔ expr?

- But it’s not the same problem!
Expressiveness of IP Groups

Space of IP datagrams

- IP groups partition the dataspace!
  - A datagram belongs to at most one group
Subscriptions specify arbitrary subsets!
- A notification can match multiple subscriptions