Objective

- Multiple users wish to share a java applet
- One user drives the applet: the publisher
- Other users see what the first user does: the subscribers
Approaches

• Rewrite AWT to handle remote communication
• Capture system events and transfer them to subscribers (lots of info)
• Capture higher level events (ActionEvents, etc.)
• Simply send bitmaps
Existing and past work

• Very incomplete
• NAWT
• CollAWT
• RAWT (Remote AWT from IBM)
  – Client-server
• Various strategies involving swing
  – Platform independent (lightweight components)
• Publish/subscribe (current approach)
Imposed Restrictions

• Do not wish to rewrite AWT and Swing classes
• Users should not have to modify existing applets
• At worst, modifications to existing applets should be very minimal
• Changes to applets should be sufficiently robust to allow for automation
Event models

- Components
- Containers
- Panels, Frames
- Buttons, Textfields, Scrollbars, etc.
Event Models

- Top level button receives an event
- The event is not transmitted to panel below the button (Java 1.x, x > 0)
- Event: time, (x,y), action, command (text field), etc.
- Event: serializable
Event models in Java

• Events have a source
• Events have one or more destinations: listeners
• Listeners (event destinations) are set by the event sources
  – E.g.: a user-defined button states that a panel in another window is listening to the action of this button.
Event Hierarchy

• System level events (mouse clicks, mouse press, mouse drag, keyboard)
  – Very simple: x,y, button state, key combination
• Events occur over a widget
• Widget translates even into EventObject or one of its subclasses
  – ActionEvent, MouseEvent, ItemEvent, etc.
  – These events have increased information:
  – Time of occurrence, number of clicks, command associated with event (disable, press, highlight, etc.)
Approaches

• Use Robots
  – Allows application to post a system event
  – System event is identical to event generated by a mouse click or a key stroke
  – Java components respond accordingly

• Disadvantages of robots
  – Too many events generated
  – Cannot take advantage of event merging (managed by Java System Queue)
Approaches

• Better Approach
  – Send serialized EventObjects (root of event hierarchy) across the network
  – Recipient will transmit these events to their destinations
    • Post these events to the system event queue
  – Problem: event sources are not updated
    • E.g. text fields, selection items, etc.
  – Solution (not satisfactory):
    • Update sources based on event and event content
    • Takes valuable computer time
Other issues

• EventObject contains source of event
• Event source is transient: identical objects on different platforms cannot be checked for equality
• Solution:
  – Create hash tables: to each object associate an index
  – To each index, associate an object
  – Assumption: collaborating applets are identical
  – Send object ID along with event
Publish/Subscribe

• Use Narada: Developed by Shreedeep

• Narada implements the JMS (Java Messaging Service) interface

• One applet is the publisher

• Other applets are the subscribers

• All applets have identical hashtables (same number of objects, same ids, different object references)
Approach

• Publisher sends message to server
  – Serialized event + metadata (object id)

• Subscriber receives package from server
  – Unserializes the event, sets the event source (AWTEvent) based on the object reference obtained from the hash table (the id is known)
  – Post the message to the system queue
  – Listeners receive the message and act upon it (in exactly the same way as the publisher)
  – Low level routine (jms.processEvents) checks the source of the event and updates it accordingly (buttons, text fields, etc.)
Problems

• Swing components are not working under this approach: reason unknown
• AWT components work fine
• Multiple applets per page correspond to multiple instances of the applet class. These are all subscribers or all publishers.
  – Result: (molecule demo): one applet will drive all applets on the page (will create problems unless all applets are the same)
• How to select who is publisher and who is subscriber. Need mechanism to all publisher to change “on the fly”
Benchmarks

• Create synthetic events
  – Bypass Narada
  – Measure # mouse events received/sec
  – Turn Narada on but don’t send messages
  – Time with narada on same server as running applet or on different server
  – Time when two servers interacting (1 pub + 1 sub
  – Time with increasing number of subscribers (on different machines)
  – How many mouse clicks/sec before messages do not get through.
TODO

• How to reduce number of messages sent (mouse moves and drags)
• How to update source widgets in a more generic fashion
• How to deal with swing
• What should be strategy for topic naming
• Applet resource discovery mechanism
Flowchart

- Show how information flows between publisher and subscriber
- Show image of Narada in the network
- Show how people with PDAs, servers, PCs, remotely distributed can all interact
Examples

• Font tables with item menu and font selection list
• Graph Layout
• Molecule (problem of 4 applets on a page)
• Webmap (will probably not work)