WYSIWIS and Shared Windows

Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu

COUPLING

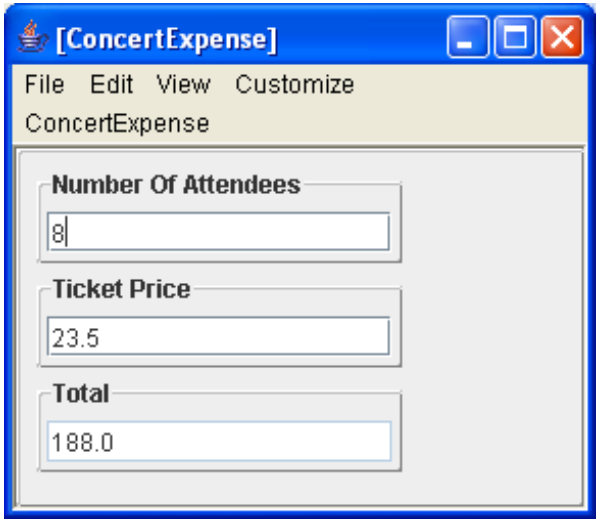
Issue	Description
Session Management	How do distributed users create, destroy, join, and leave collaborative sessions?
Single-user Interface	What are the application semantics if there is a single user in the session?
Coupling	What is the remote feedback of a user command and when is it given?
Access Control	How do we ensure that users do not execute unauthorized commands?
Concurrency Control	How do we ensure that concurrent users do not enter inconsistent commands?

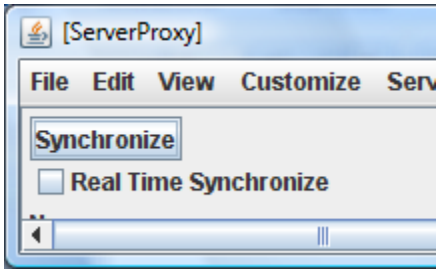
Application-Specific Coupling

```
Please enter an input line or quit or history
The woods are lovely dark and deep
[Alice]The woods are lovely dark and deep
Please enter an input line or quit or history
[Bob]But I have promises to keep
[Cathy]And miles to go before I sleep
history
[Alice]The woods are lovely dark and deep, [Bob]But I have promises to keep, [Cathy]And miles to go before I sleep

Please enter an input line or quit or history
```

Coupling depends on model-interactor division, when changes are announced, when they are sent, and when they are applied



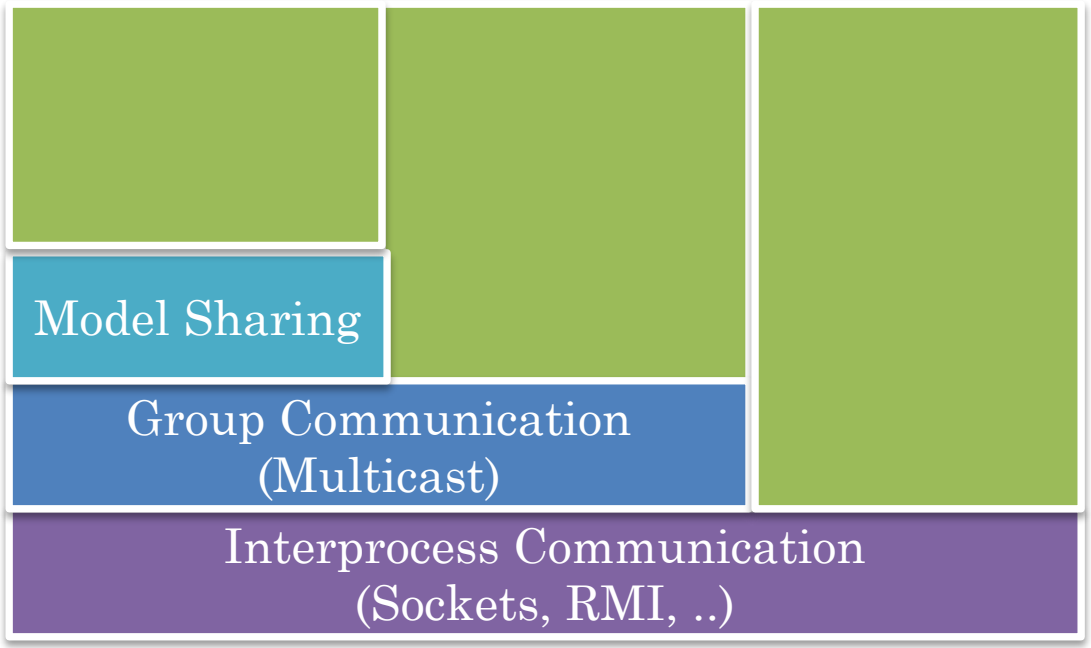


Application-Independent Coupling?

Why?

What?

Programmer Effort: Automation



Model is a blackbox and we had to make assumptions about it to automate sharing

Not all programmers are aware of or care about model-interactor division

End-User

- Easier to understand for the user
  - Is synchronization real-time or not real time?
  - What changes are sent?
  - …..

Challenges

- How to define it in application independent fashion.
  - Mapping between input and local feedback is application dependent.
  - Implies mapping between input and remote feedback is also app-dependent.
  - Relationship between local and remote feedback may be app-independent.

WYSIWIS: What You See Is What I See

Remote Feedback
=
Local Feedback

Application

When?

If Interaction Stopped, Remote User Will Eventually Get Feedback

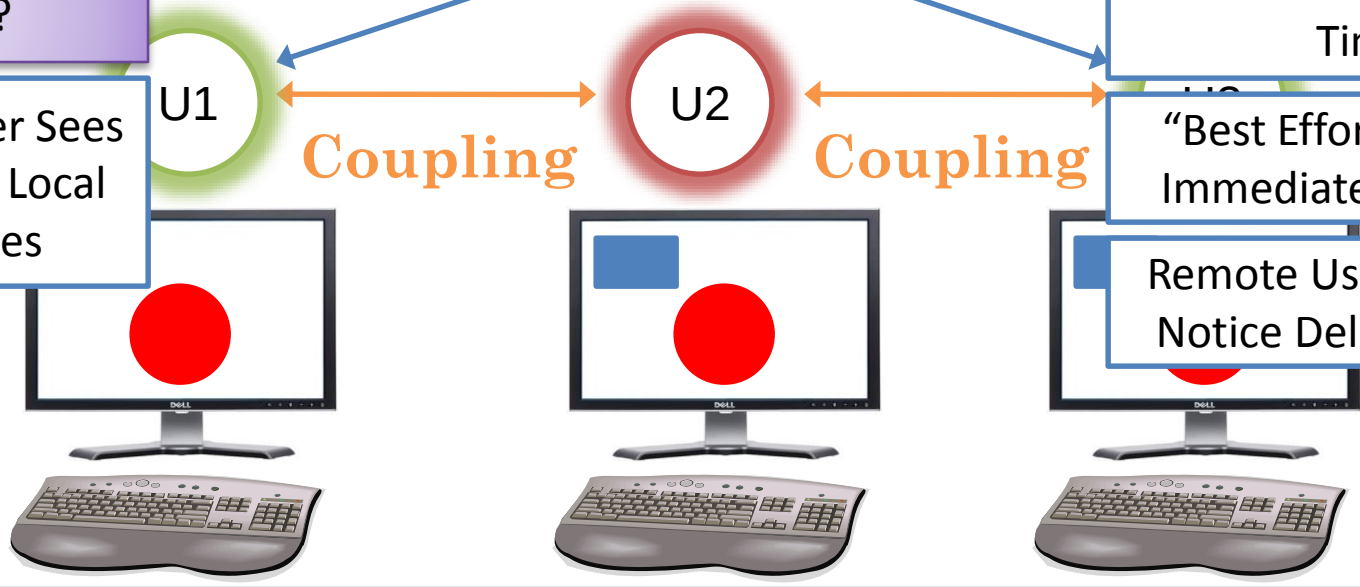
Remote User Feels Collaboration is Real Time

"Best Effort" to Give Immediate Feedback

Remote User does not Notice Delay (<50ms)

What?

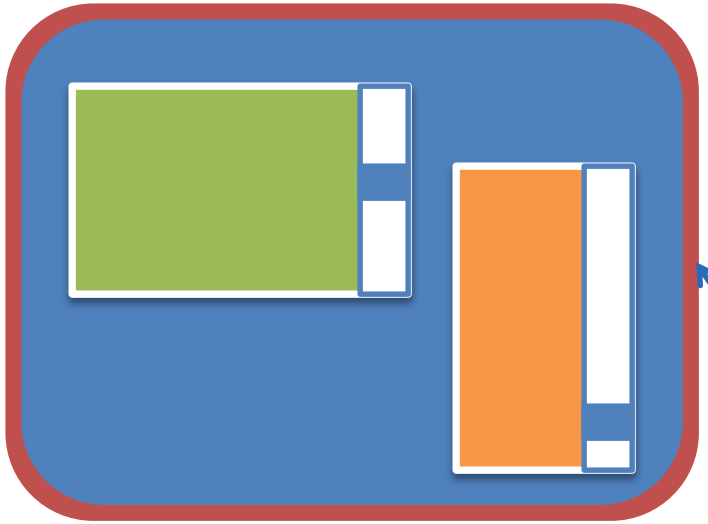
Remote User Sees Everything Local User Sees



Stefik, M., G. Foster, D.G. Bobrow, K. Kahn, S. Lanning, and L. Suchman (January 1987), *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings.* CACM, January 1987. **30**(1): p. 32-47.

STRICT WYSIWIS COUPLING

User 1

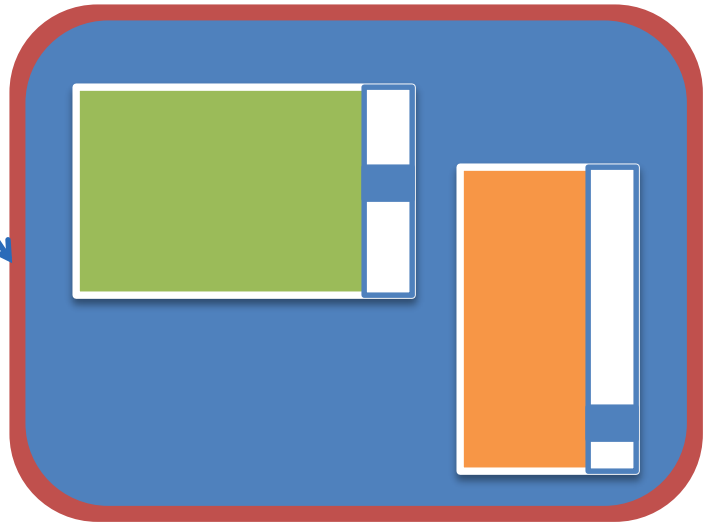


If user 1 moves green window

Then green window moves on user 2's screen

If user 2 scrolls up in orange window

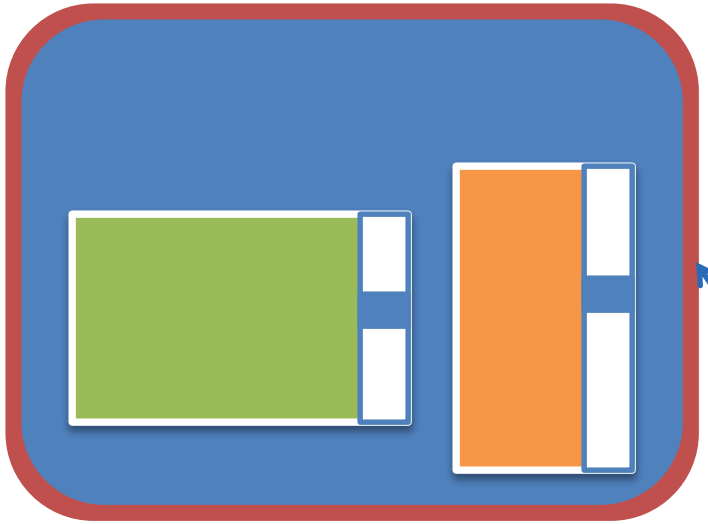
Then orange window scrolls up on user 2's screen



User 2

Pros and Cons

User 1



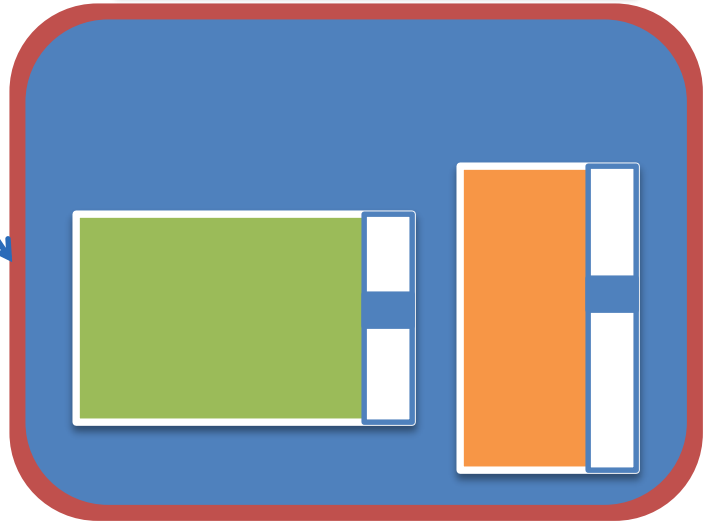
Pros

- Easy to understand
- Application-independent
- Automatable

Are some wars worse than others?

Cons

- Window and scroll wars
- Size and view wars
- Communication overhead



User 2

Near/Relaxed-WYSIWIS

- Window Positions
- Window Size
- Screen Pointer
- Mouse Move
- Window Border
- Scroll Positions
- Key Click
- Mouse Click
- Mouse Drag

Different window sizes create ambiguities (cropping, scaling) and scroll synchronization creates referential transparency

Screen pointer has not much meaning if some windows are shared or if windows are moved independently

How to point?

Telepointer: A shared shape (possibly per user) that can be dragged over any part of a top-level window

May have private windows which are obscured

How to automate WYSIWIS/near WYSIWIS?

Stefik, M., D.G. Bobrow, G. Foster, S. Lanning, and D. Tatar (April 1987), *WYSIWIS Revised: Early Experiences with Multiuser Interfaces.* ACM Transactions on Office Information Systems, April 1987. **5**(2): p. 147-167.

Automation & Pre-Requisites

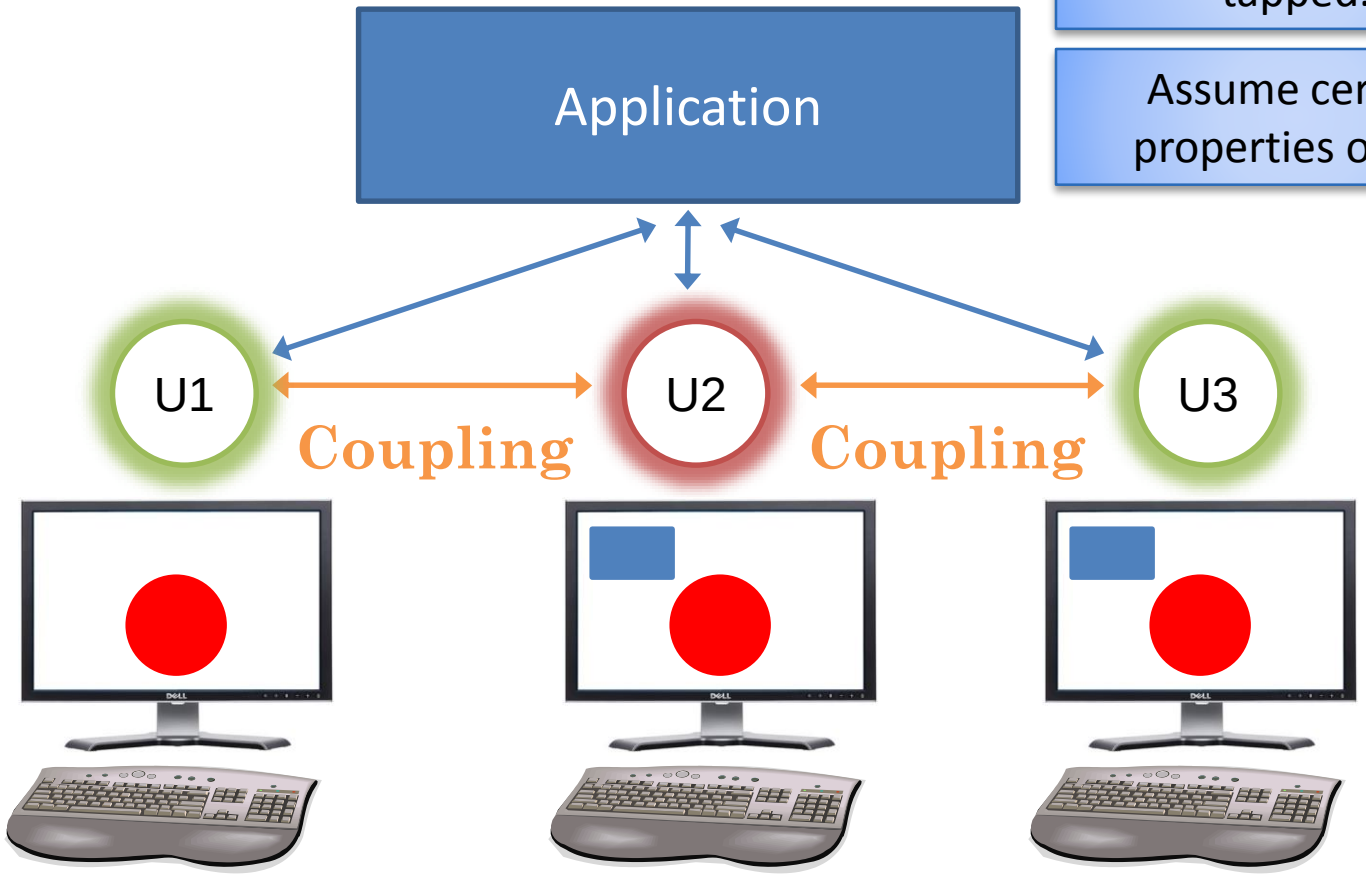
- Automation: Some application-independent infrastructure provides some functionality.
  - UI toolkit automates widgets.
  - Sync automates model sharing
- Use assumptions to provide the automation.
  - Swing and AWT assume applications do not want round widgets.
  - Sync assumes users do not want interactor sharing
- Need to make similar assumptions.
  - As not model sharing, assume I/O in interactor instead of write methods are tapped

Automating Constraints

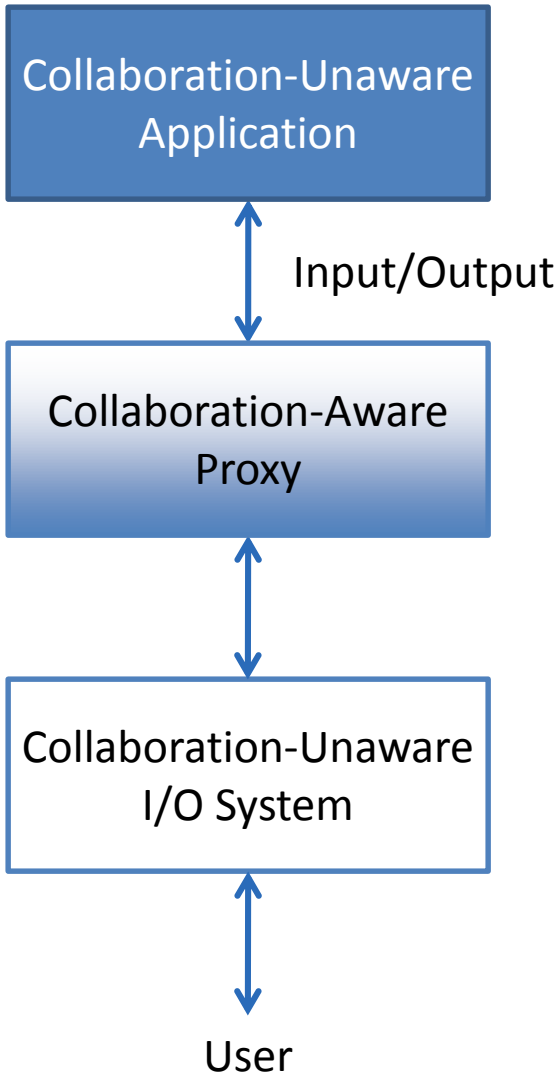
Need access to input and/or output

Assume I/O can be tapped.

Assume certain properties of I/O

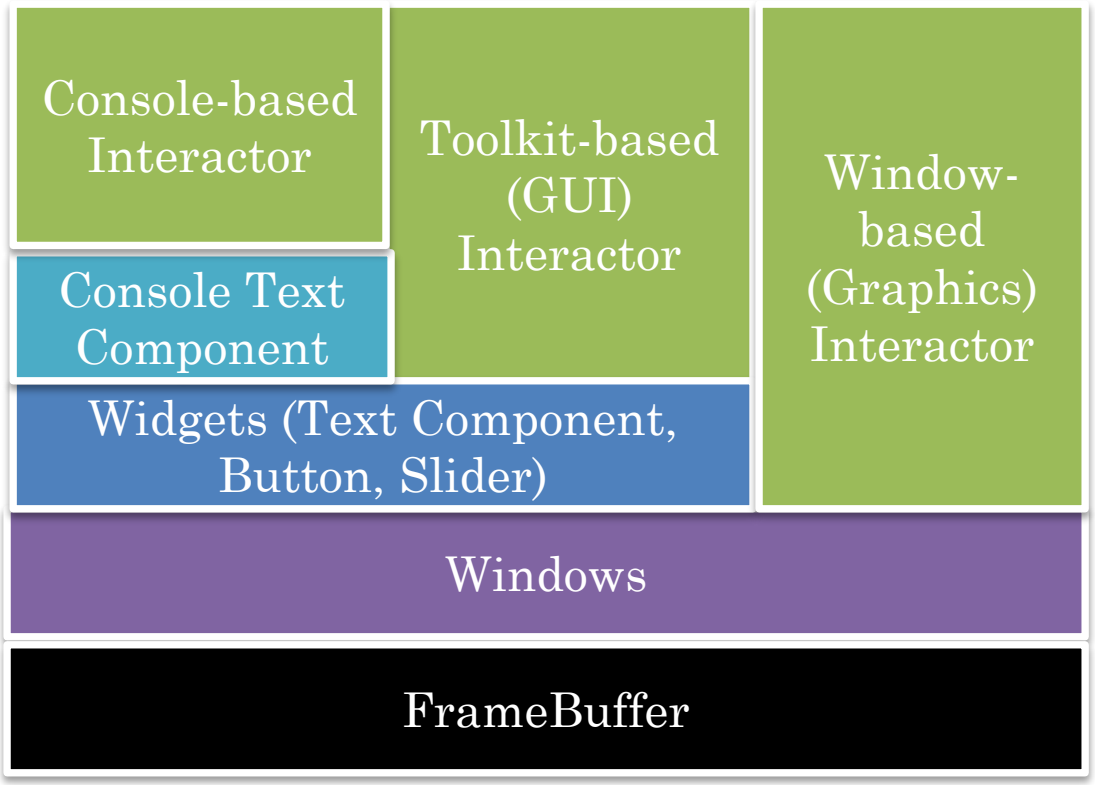


Shared I/O Systems



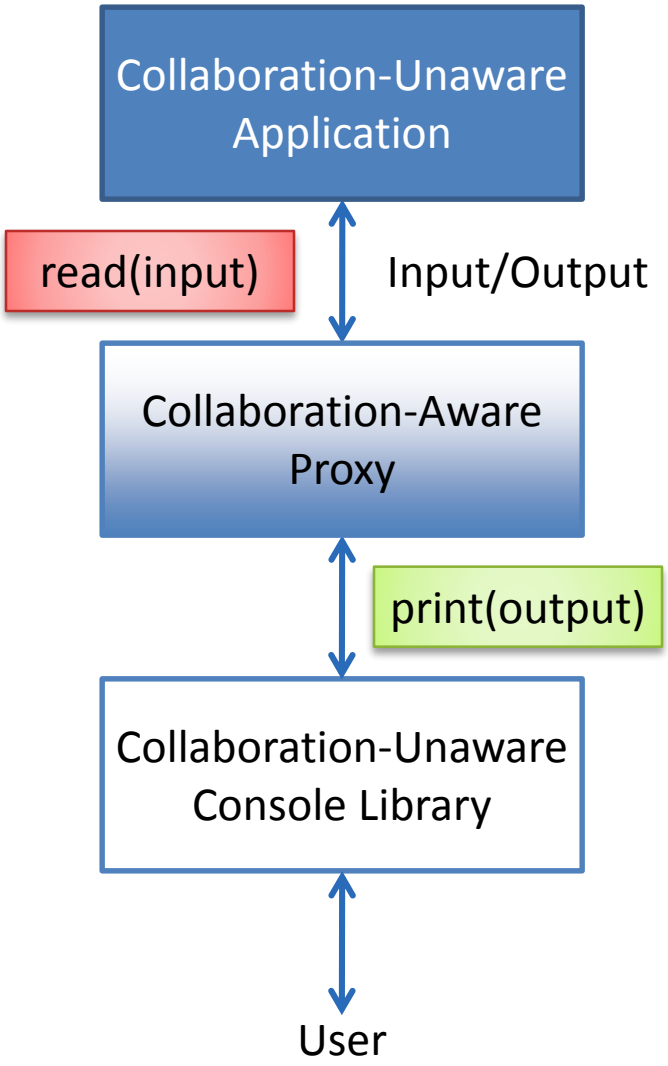
- Assume application uses some I/O system
- Will put a proxy module in between application and I/O system, if possible
- To the application it behaves like the I/O system
- To the I/O system it behaves like the application
- Application and I/O system are collaboration-unaware
- Proxy (infrastructure) is collaboration-aware
- Proxy will distribute I/O
- Much line proxies in shared model systems

I/O Abstractions



Flexibility vs. Automation Tradeoff in Abstraction Design

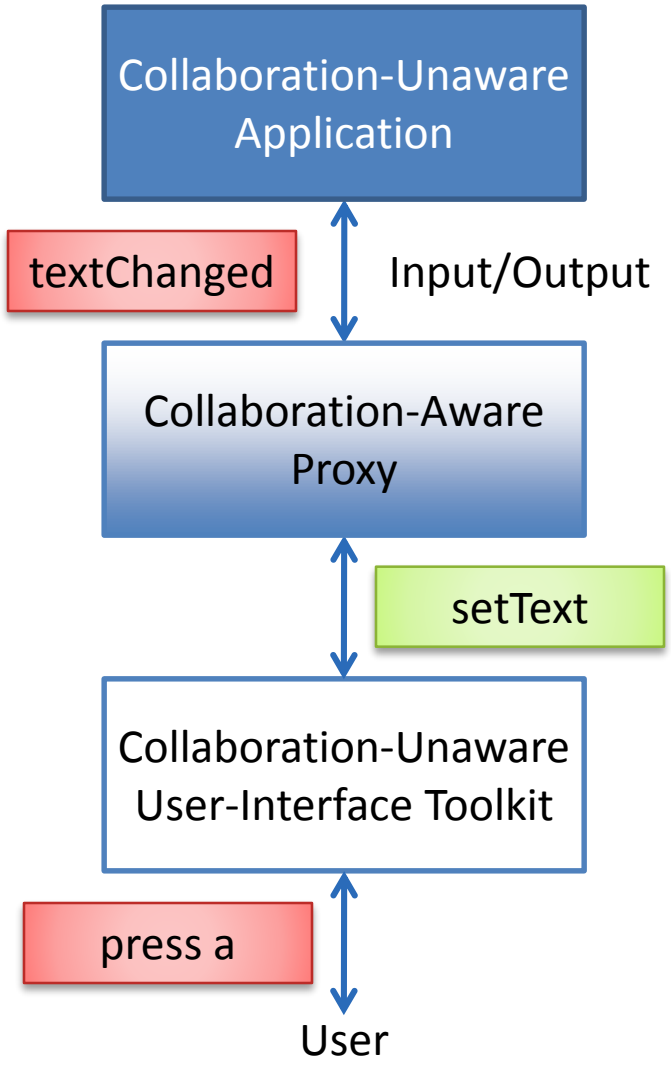
Console I/O



Programming languages and OS's provide teletype (console library)

Cannot handle GUIs

Toolkit Library I/O



Toolkit provides widgets such as text box, slider, and buttons.

Input is notifications about state changes

"Output" sets and gets widget state

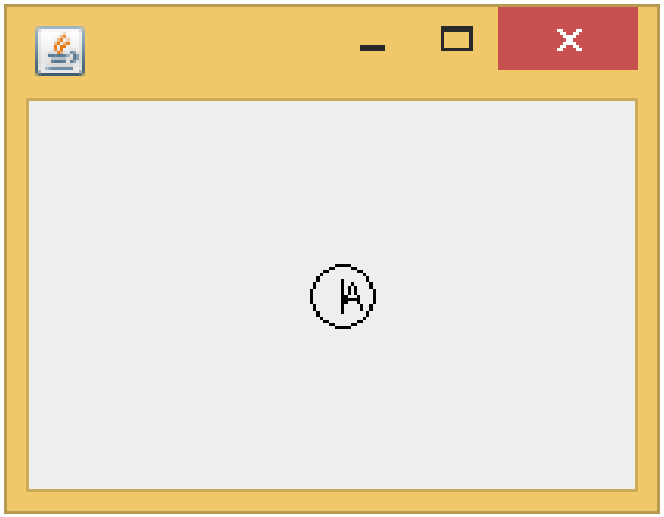
Widget automation bound to specific toolkit (Swing, AWT)

Toolkits are built on top of window system

Cannot be used to share window state (e.g. window size)

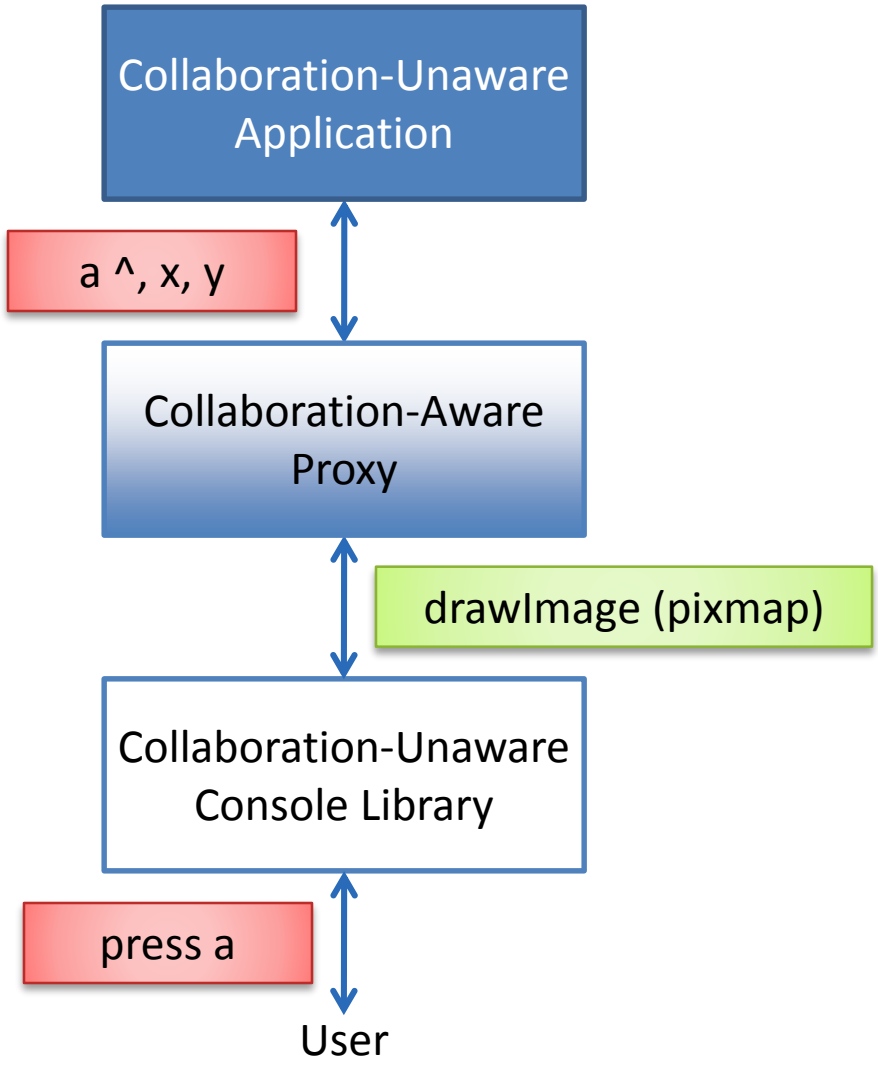
Cannot be used to share collaboration-unaware window apps

Custom Widget



Neither console nor toolkit sharing can share this user-interface

FrameBuffer I/O



Input: Mouse and character events with screen coordinates

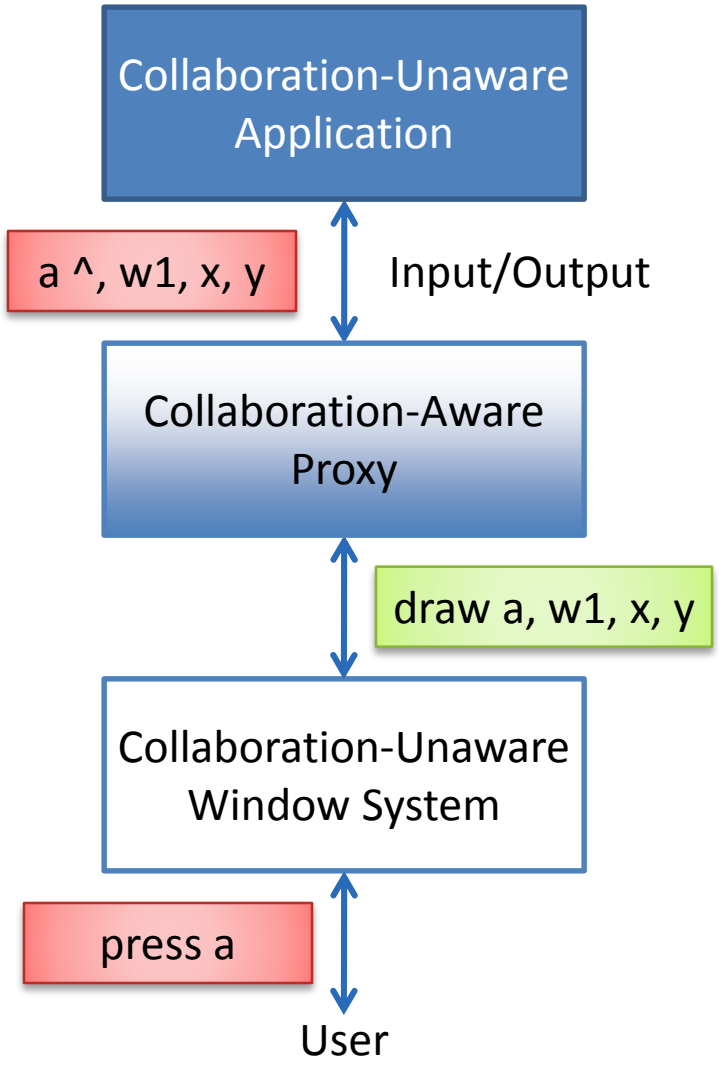
Output: Draw pixel images

Forced into WYSWIS

No private windows in pure FrameBuffer

Commercial systems combine shared window and framebuffer (later)

Window System (Simple Model)



Windows are untyped rectangular screen areas in which point is a pixel

Input indicates keyboard and mouse operations with window relative coordinates

Output draws text, shapes

Application = programmer-defined code + toolkit

Window-level automation accommodates all toolkits and allows private and public windows

Architecture with N users?

Window System I/O?

- Window Positions
- Window Size
- Screen Pointer
- Mouse Move
- Window Border
- Scroll Positions
- Key Click
- Mouse Click
- Mouse Drag

Window Manager

Window Manager

Input indicates keyboard and mouse operations with window relative coordinates

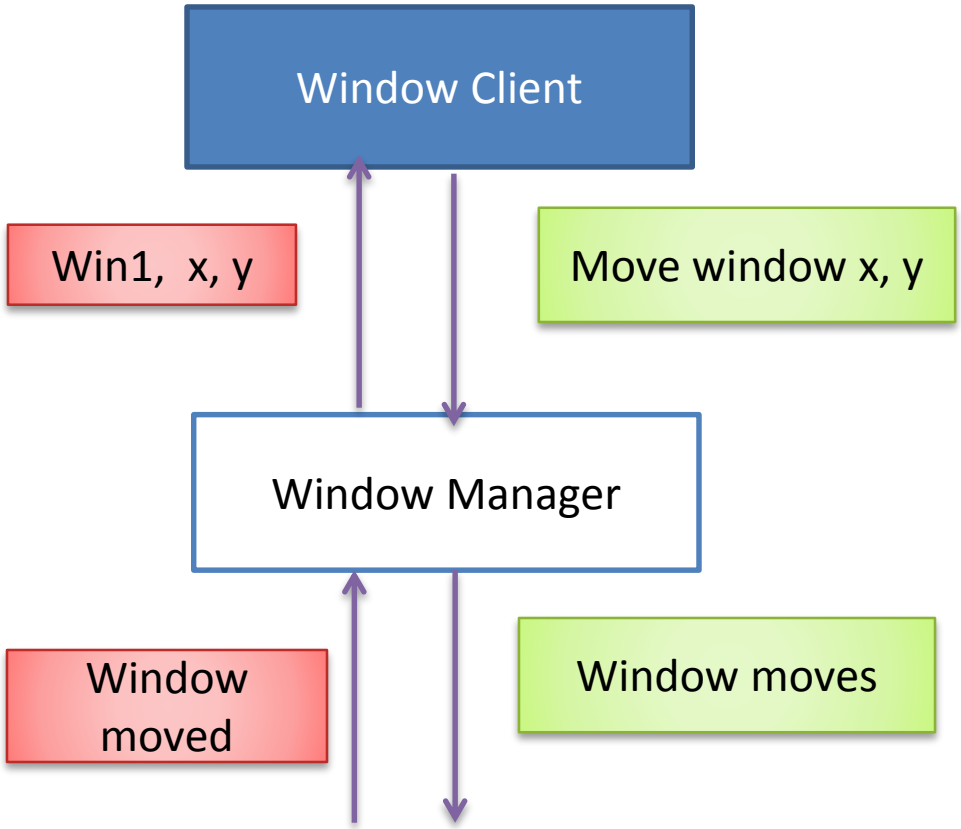
Window Manager

Toolkit

(Top-Level) window moves, resizes and mouse clicks processed by window manager

Scrolling is just mouse clicks processed by a scroll widget

Window Manager



"Input": Window move, resize user operations

"Output": Window move, resize

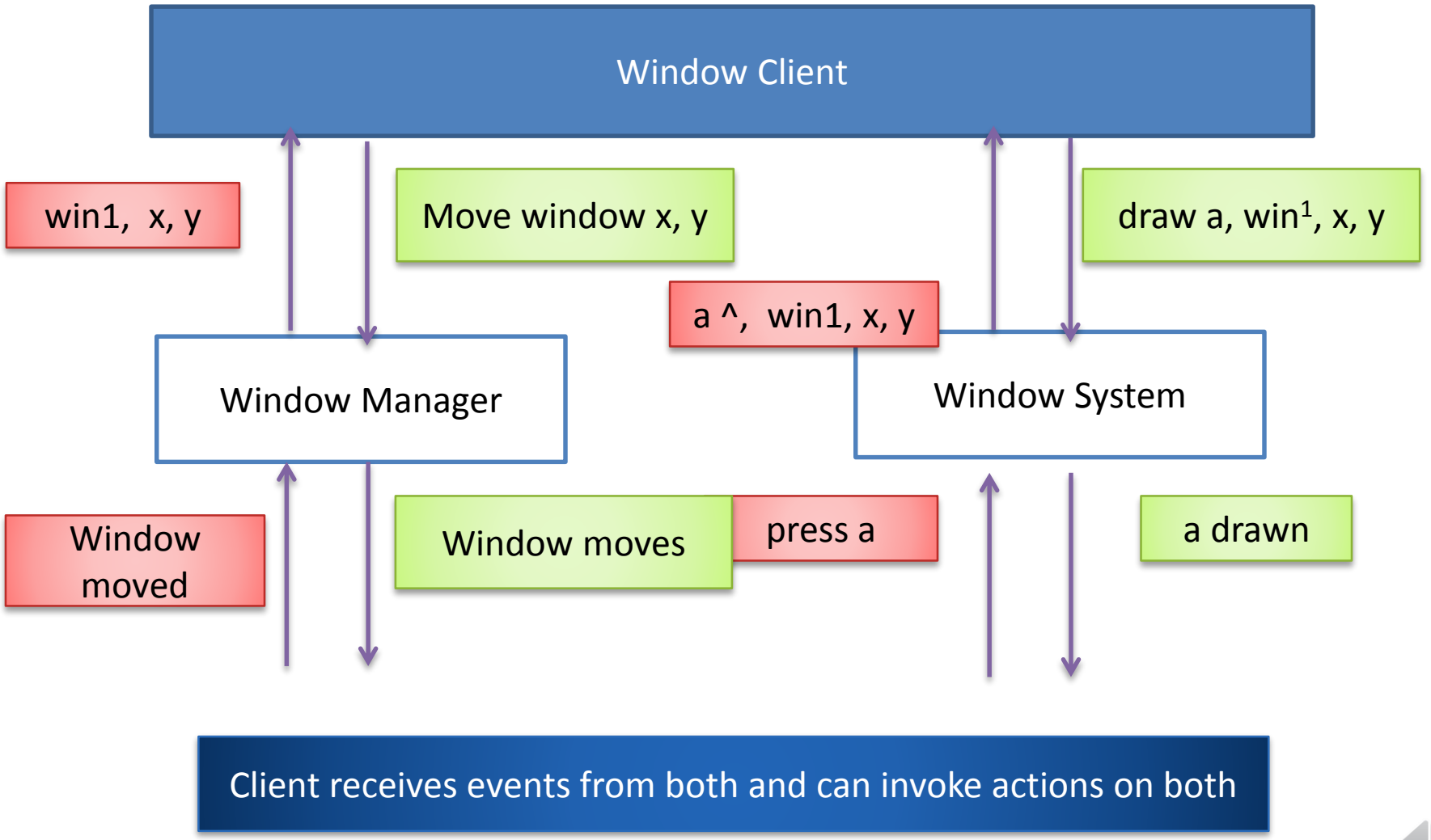
Window manager separate from window system

Can change window manager for a window system

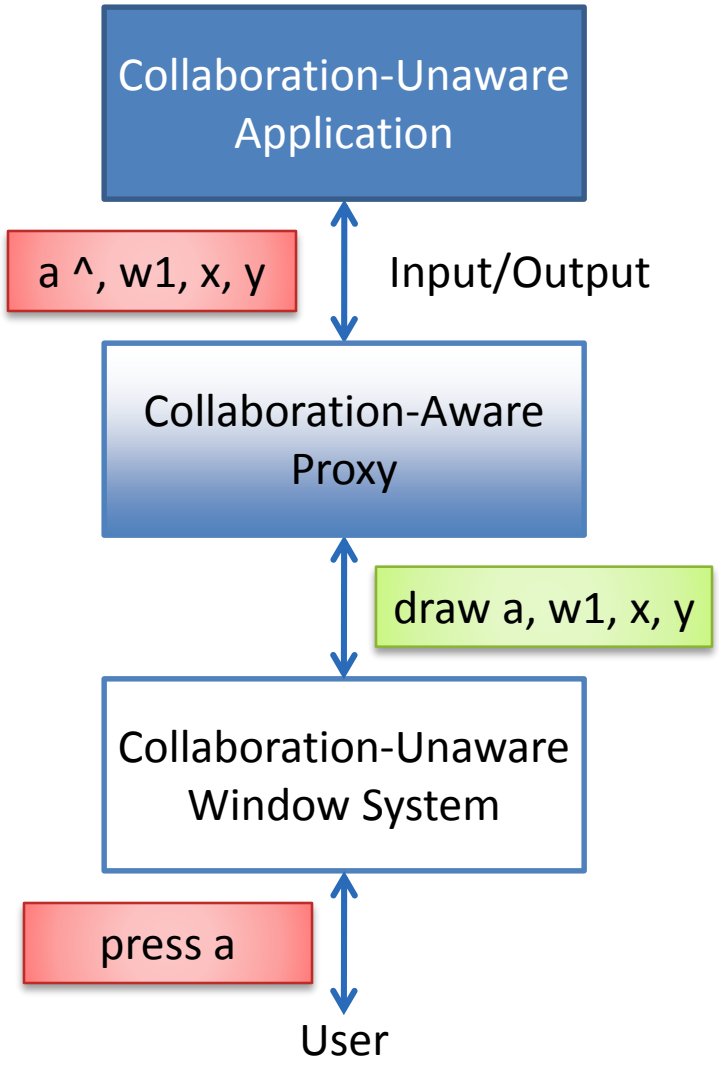
Window manager is a separate module/process

Puts decorations on window for moving resizing

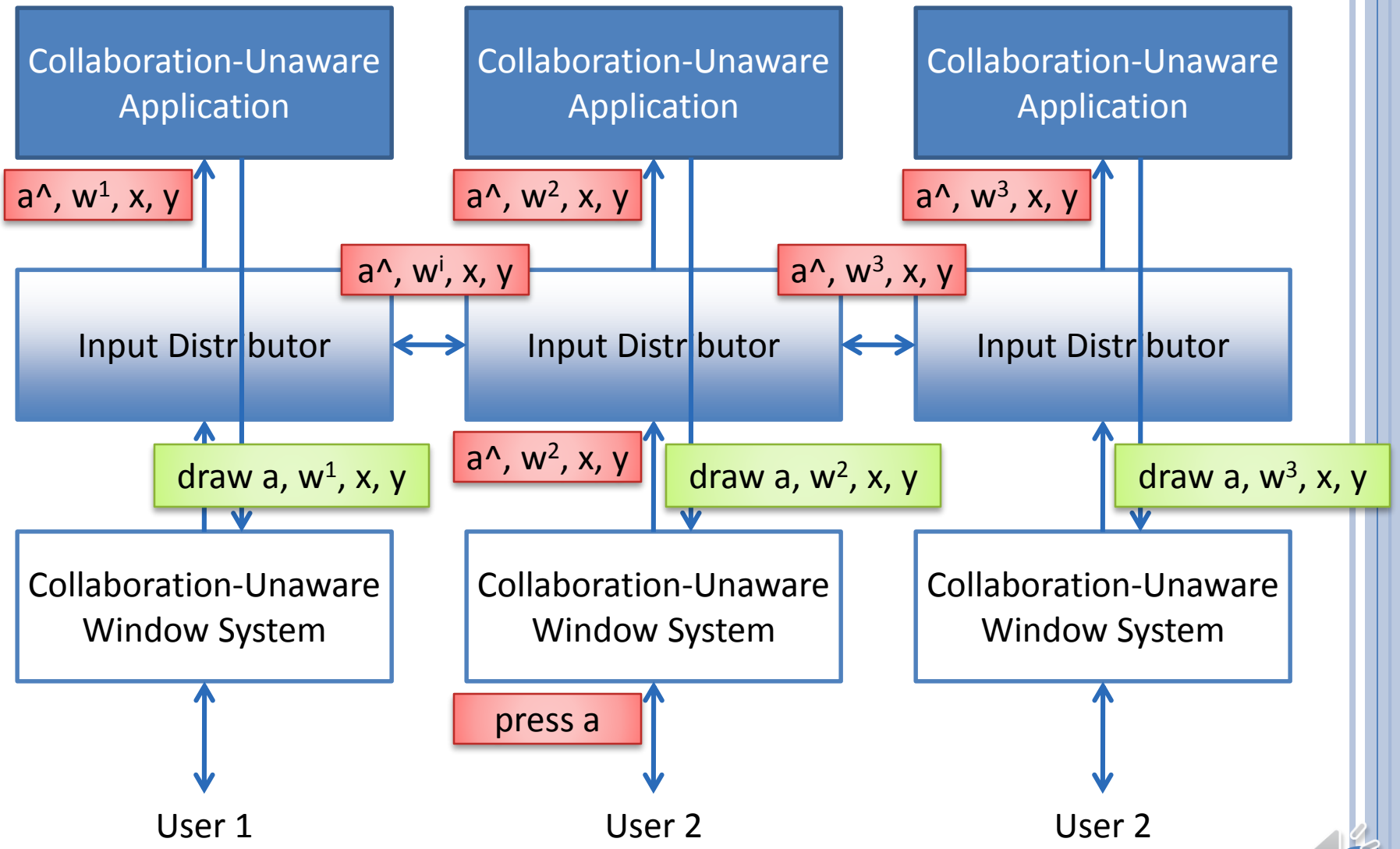
Window Manager vs. System



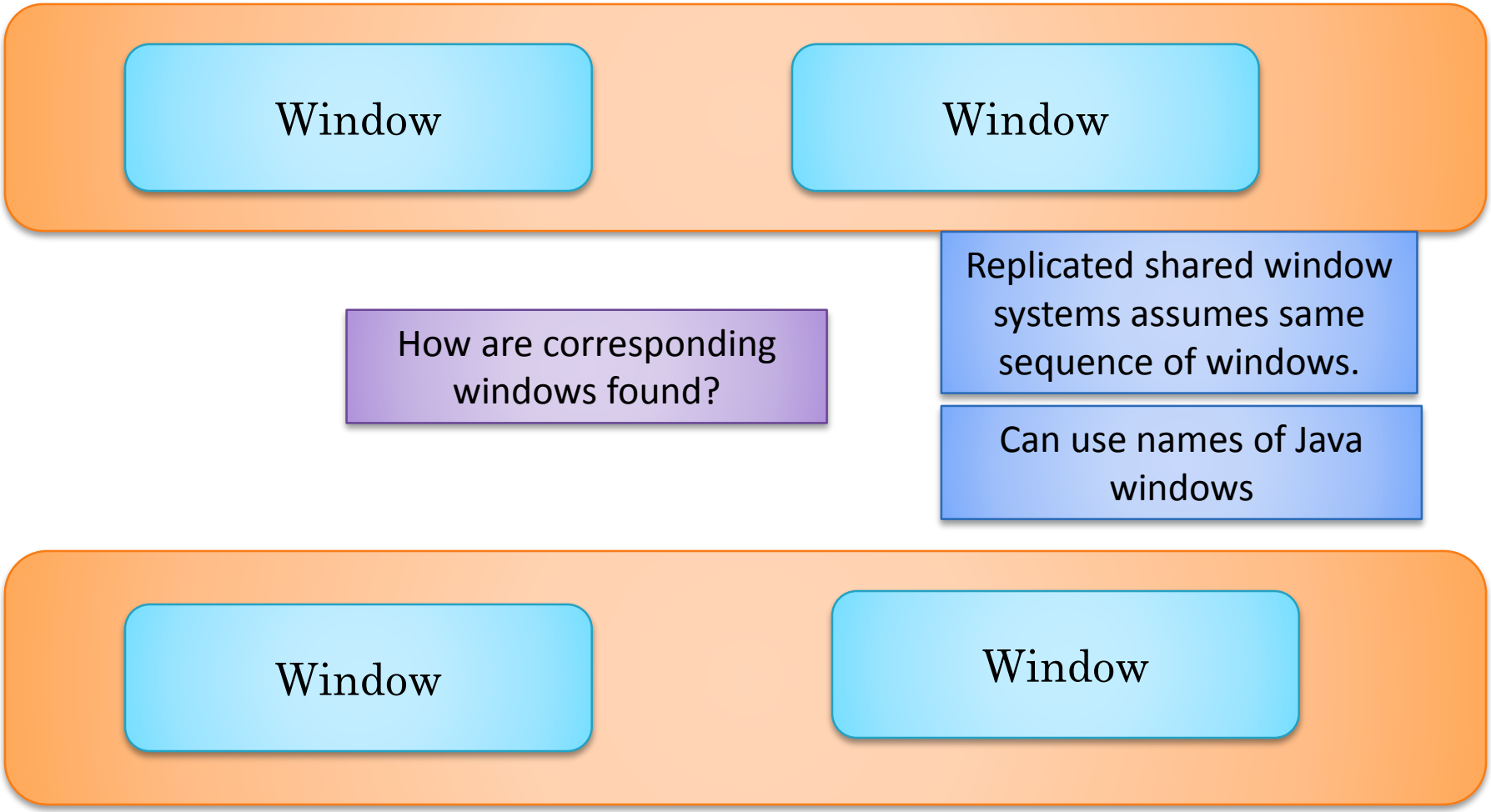
Proxy-based Injection and Interception



Replicated Window System



Connecting Replica Windows



Replicated Model Algorithm

For each input I

I should be followed by matching EditInput, EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, I should be followed by matching EditSent to Others

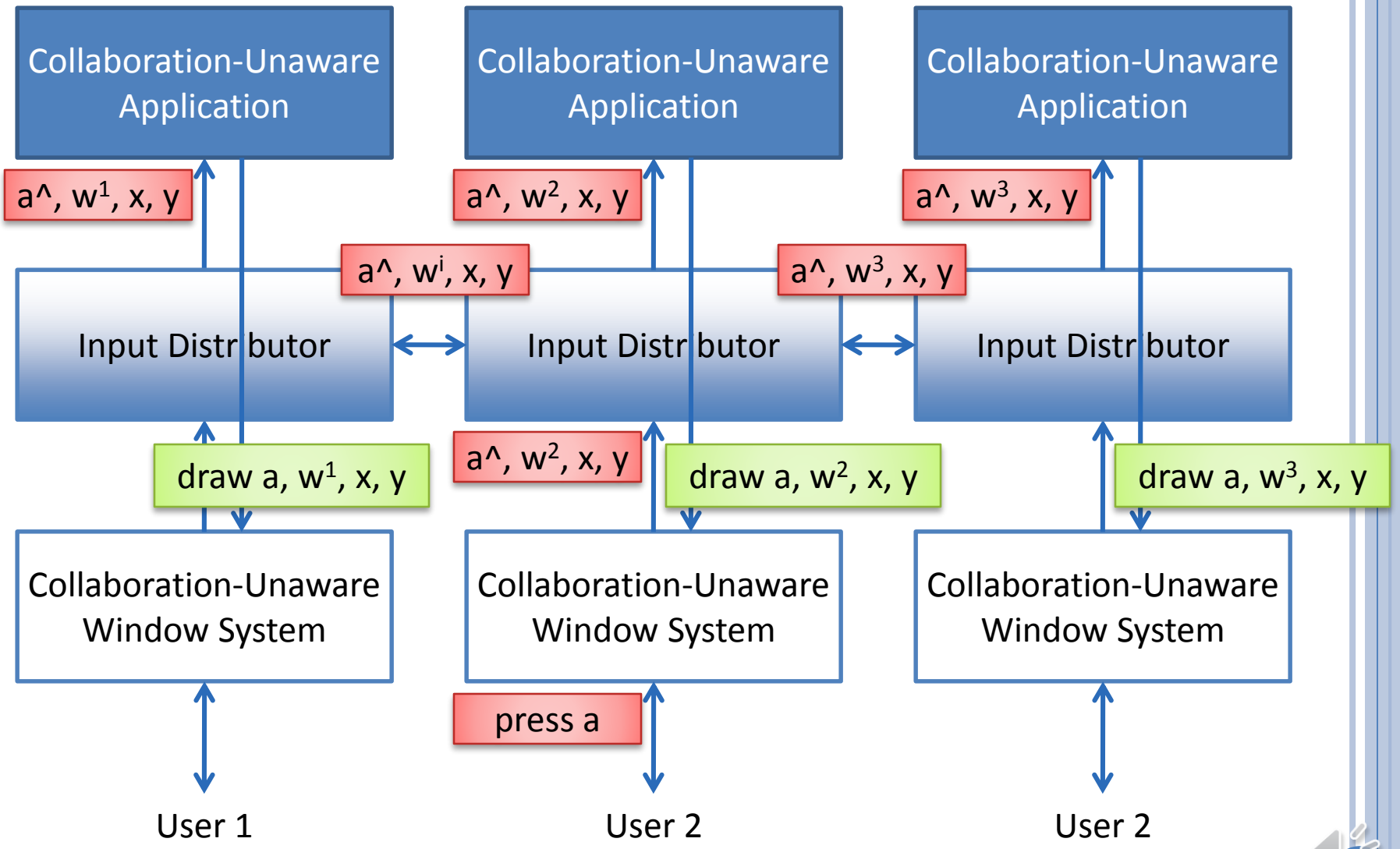
How to change to replicated window system

For each EditReceived R

R should be followed by matching EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, R should be [illegible]wed by matching EditSent

Replicated Window System (Review)



Replicated Model Algorithm

For each input I

I should be followed by matching EditInput, EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, I should be followed by matching EditSent to Others

How to change to replicated window

For each EditReceived R

R should be followed by matching EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, R should be followed by matching EditSent

Replicated Window Algorithm

For each Window w, create Telepointer w^t

Dispatching means giving it to appropriate (sub) window listeners to process

Window system can only guarantee delivery not processing

For each Window (including Telepointer) Input I

I should be followed by matching WindowEventDispatched

For each replica, if Transmit(I) then I should be followed by matching WindowEventSent

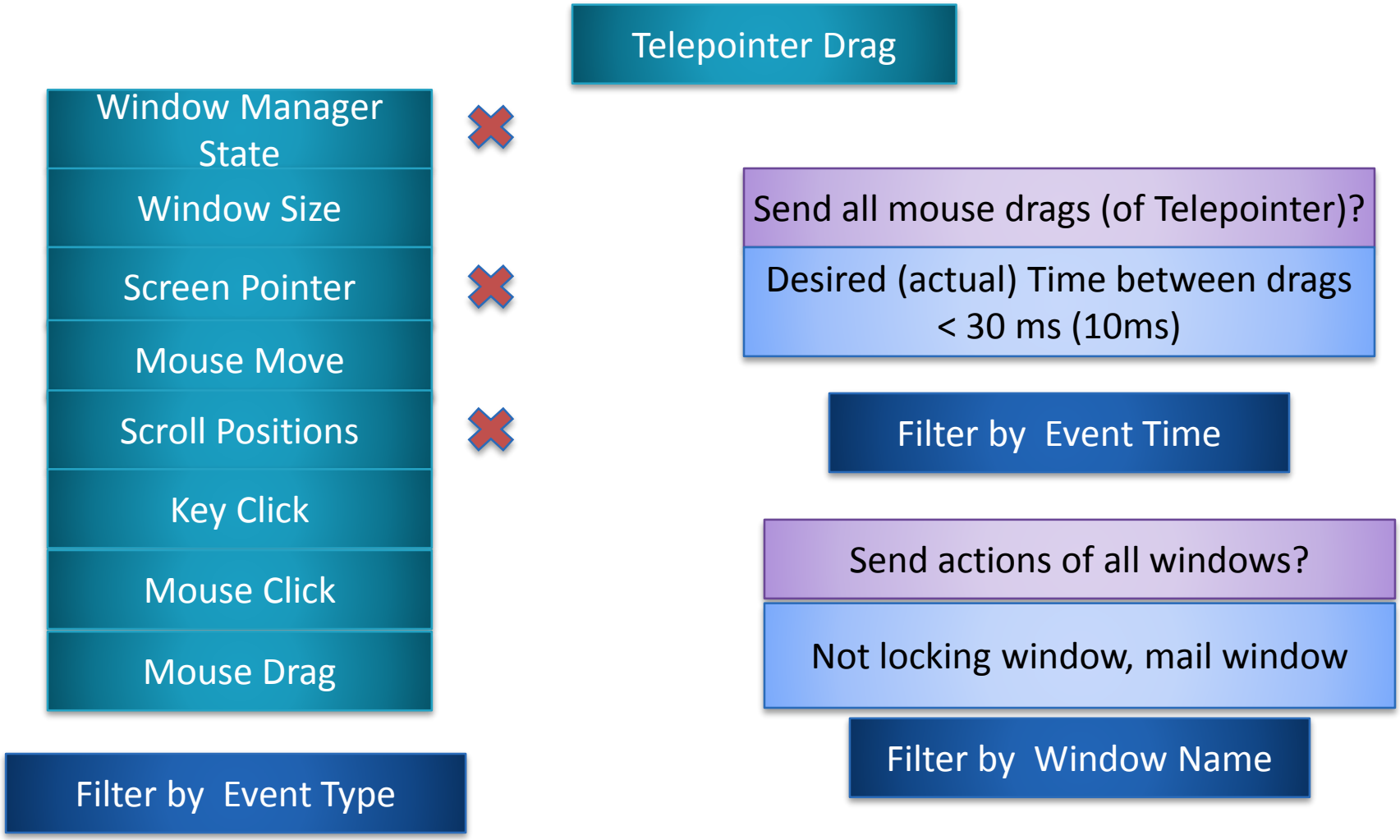
Not all events may be sent (relaxed WYSIWIS)

For each WindowEventReceived R

R should be followed by matching WIndowEventDispatched

For each replica, R should be followed by matching WindowEventSent

Transmit Function



System-Specific Issues

For each Window w, create Telepointer w^t

How to create telepointer?

For each Window (including Telepointer) Input I

I should be followed by matching WindowEventDispatched

For each replica, if Transmit(I) then I should be followed by matching WindowEventSent

How to intercept I?

For each WindowEventReceived R

How to inject I?

R should be followed by matching WIndowEventDispatched

How to filter events?

hould be followed by mat

How to translate window IDs?

Concrete Java-based Window System

Discussion so far fairly abstract

Need real window system to make it concrete

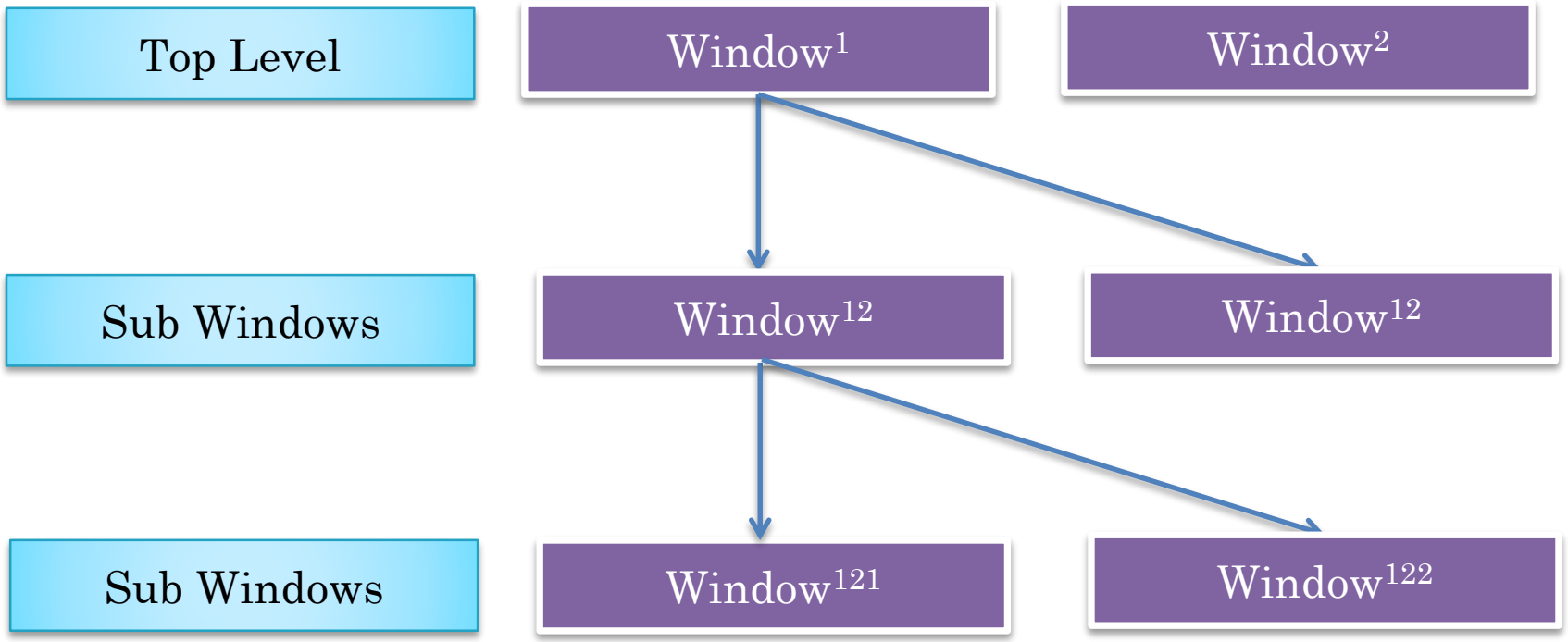
Will use Java as language for exercises and class examples

Case Study: Java AWT

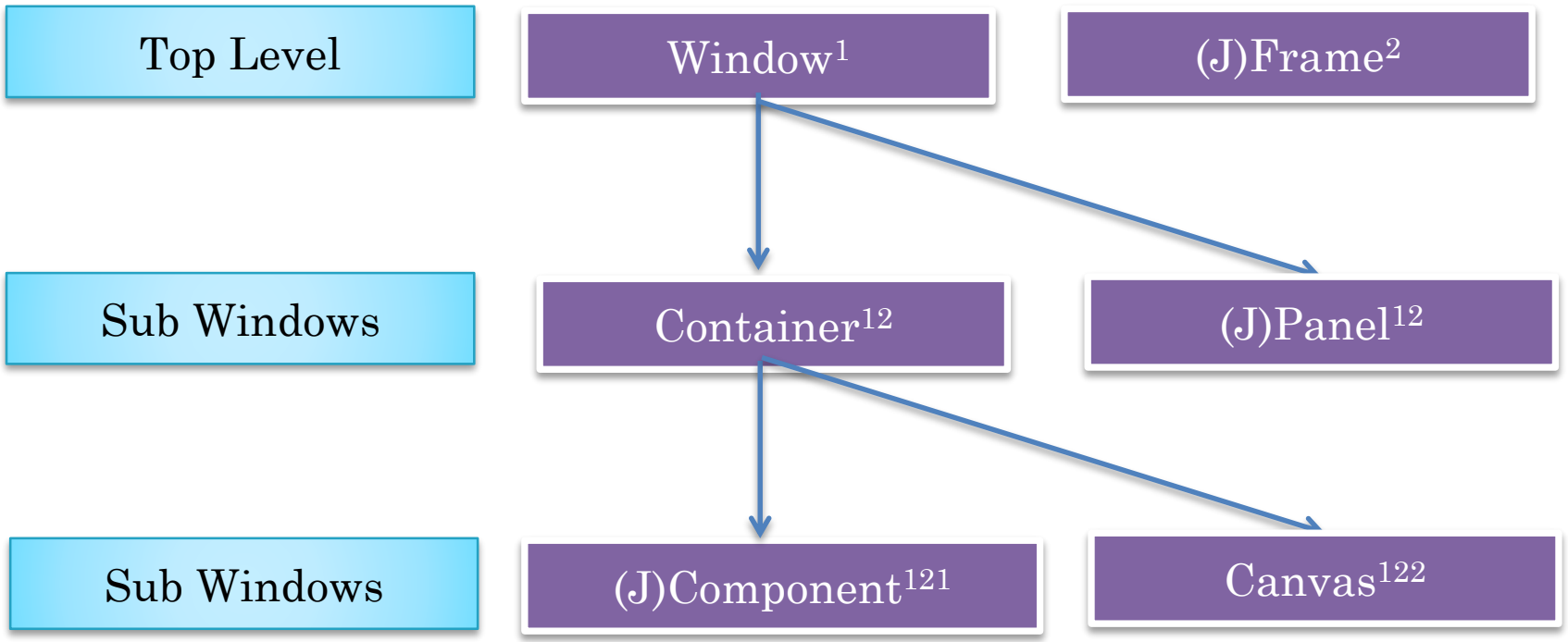
Hides the underlying window system from programmer

Portability

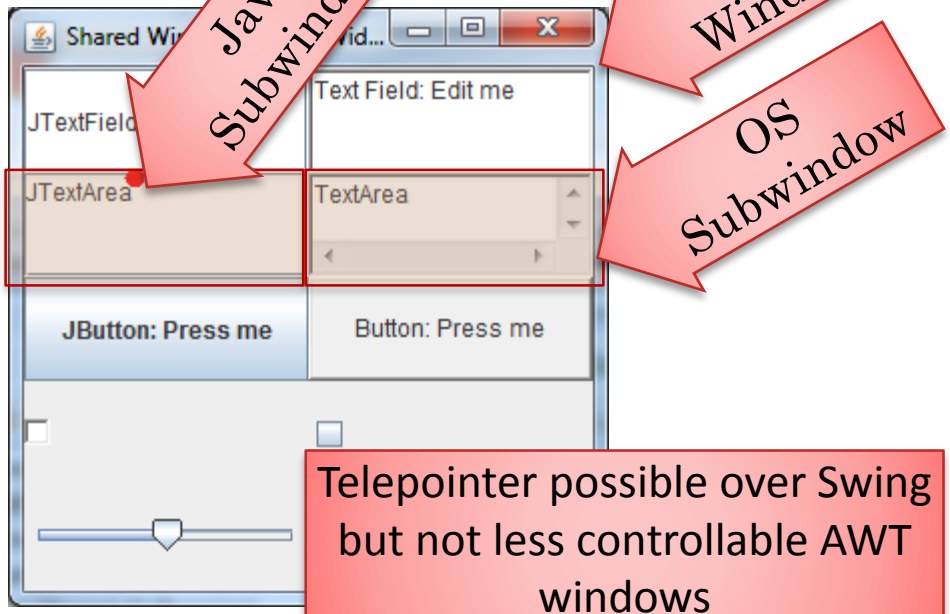
Window System Classes



AWT/Swing Window System Classes



AWT/Swing Layering



Telepointer possible over Swing but not less controllable AWT windows

Swing Windows

AWT Windows

All OS events forwarded to AWT can be intercepted and injected

All Swing events may not be interceptable/injectable

Could not intercept/inject caret position in text components

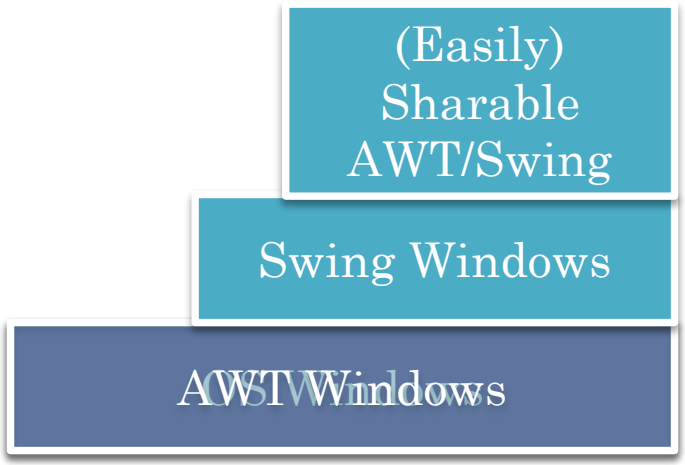
Some OS events may not be correctly interceptable/injectable

Seem to inject two events for AWT checkbox

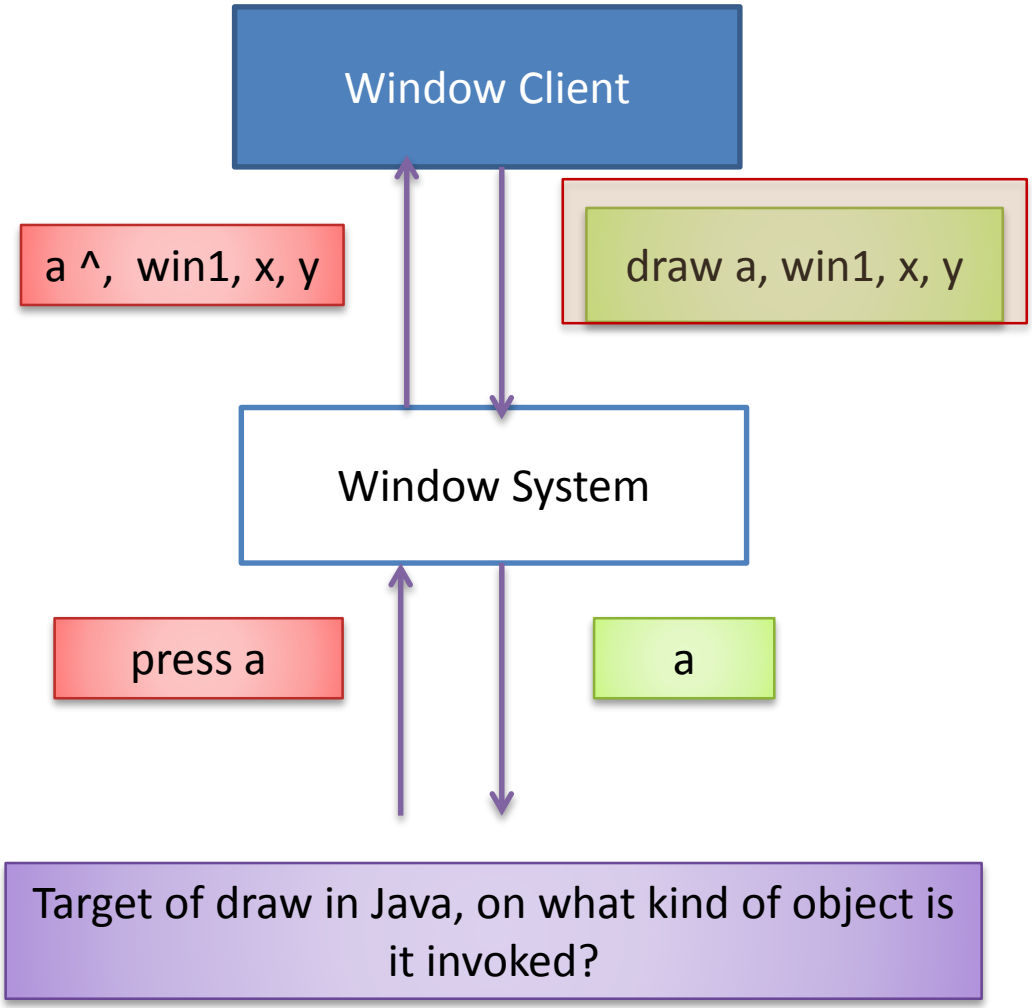
Which layer is more sharable: AWT/Swing?

Class AWT/Swing Library

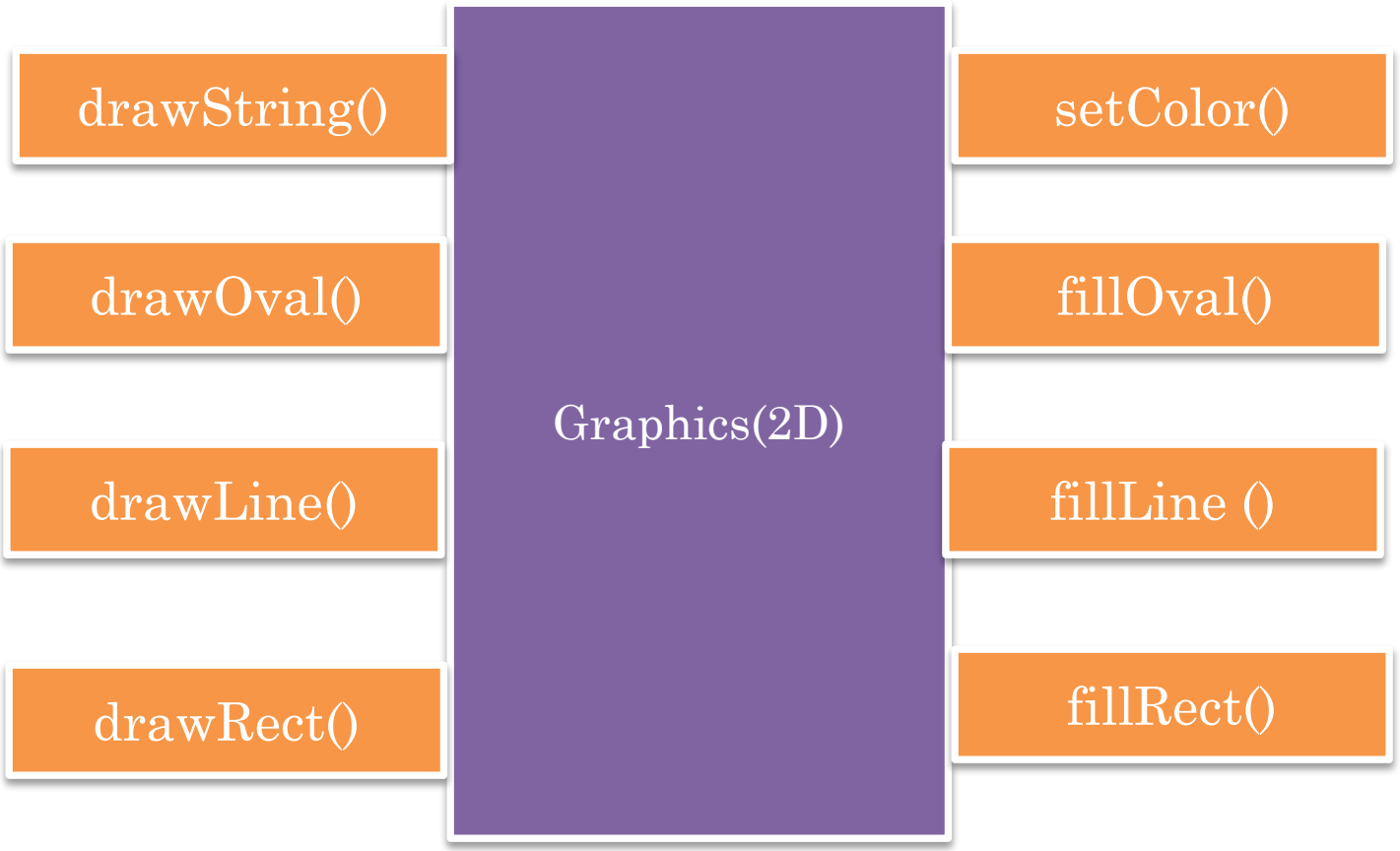
Library provided to make hide messy details of AWT/Swing



Window I/O: Output

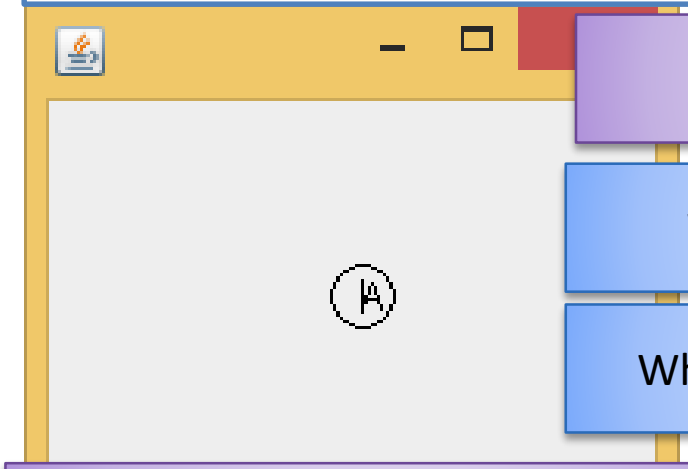


Java Helper Class: Graphics(2D) Context



Example Graphics Calls

```
 g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER,
DIAMETER);
 g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
 g.drawString("" + lastChar, charX, charY);
```



Called when?

When input is given

What if window is hidden?

When output is required

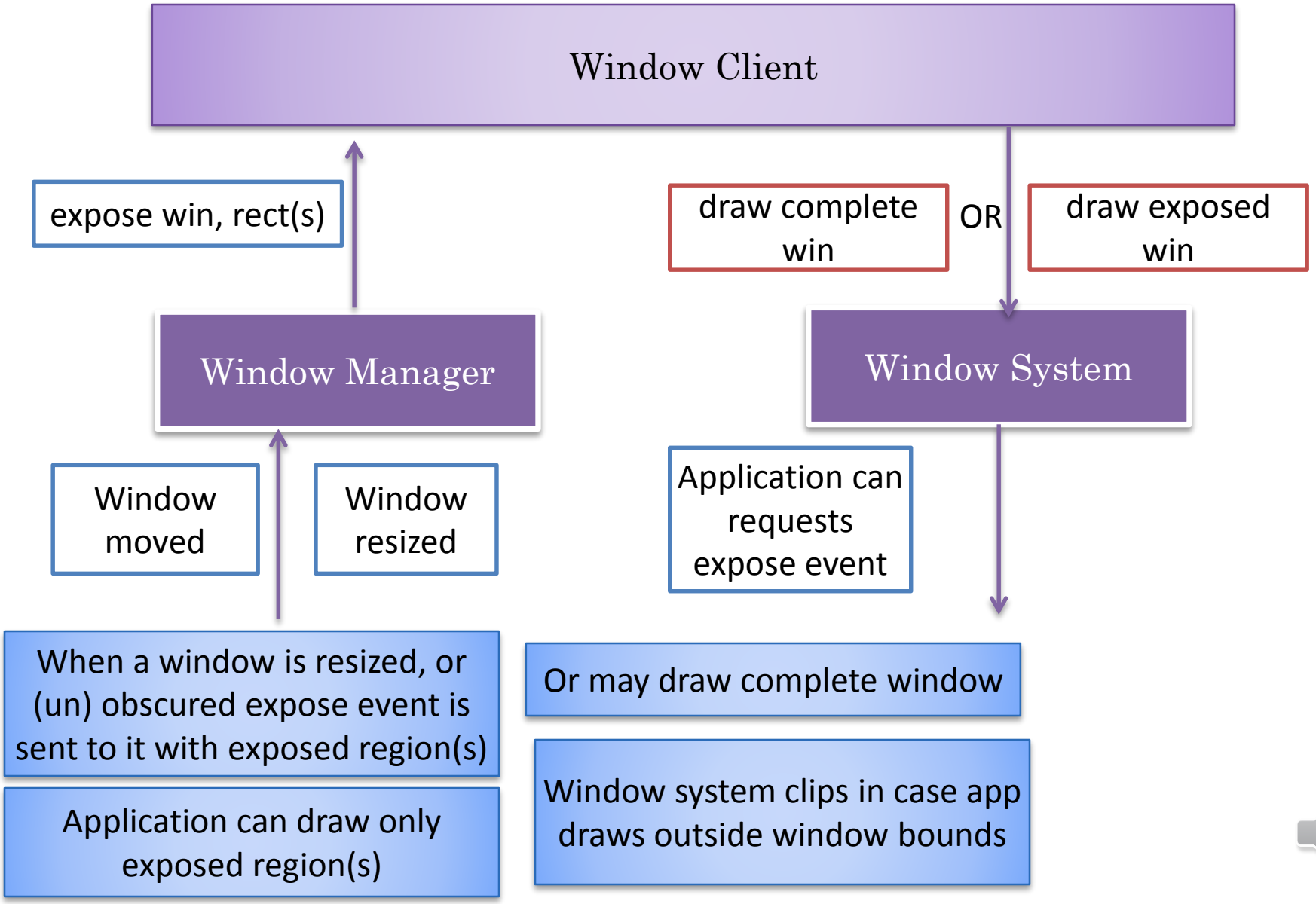
0 to N times for each input

Who repaints the window when it is exposed: window system or application

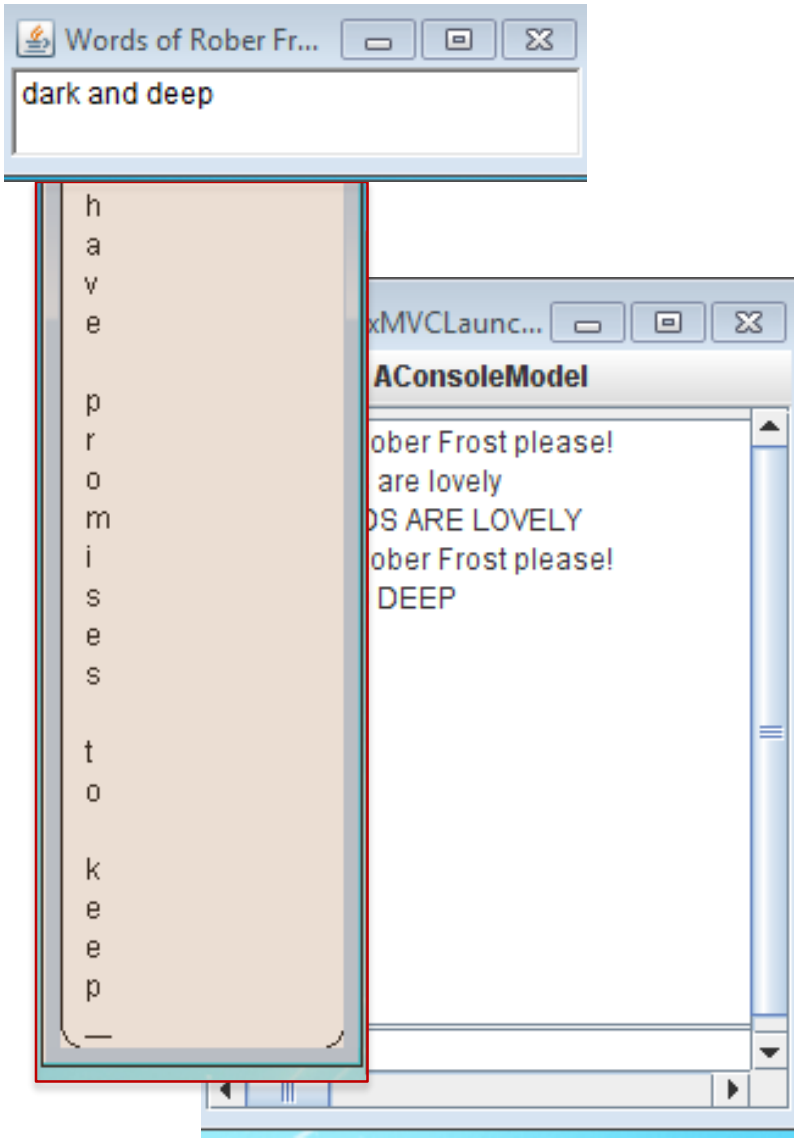
While the component is unexposed, the drawing might have changed, so system copy might not be current

So, in general the application must redraw it.

Expose Events

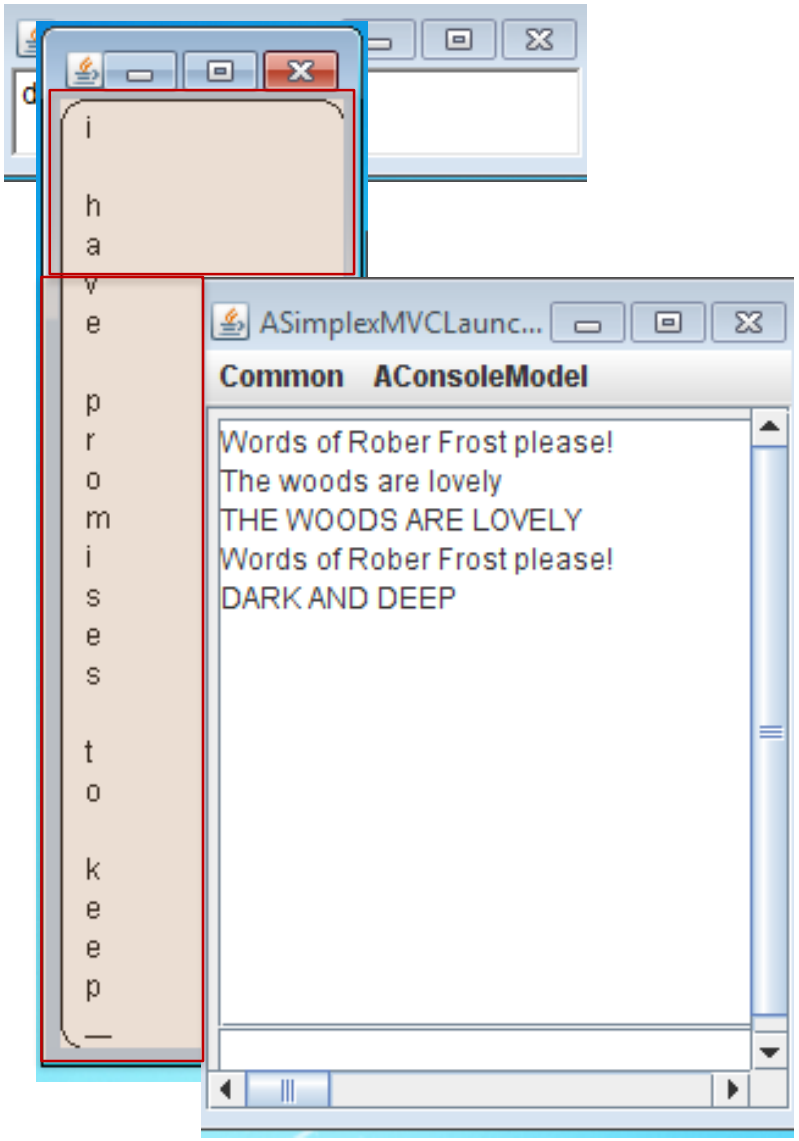


Overlapping Windows



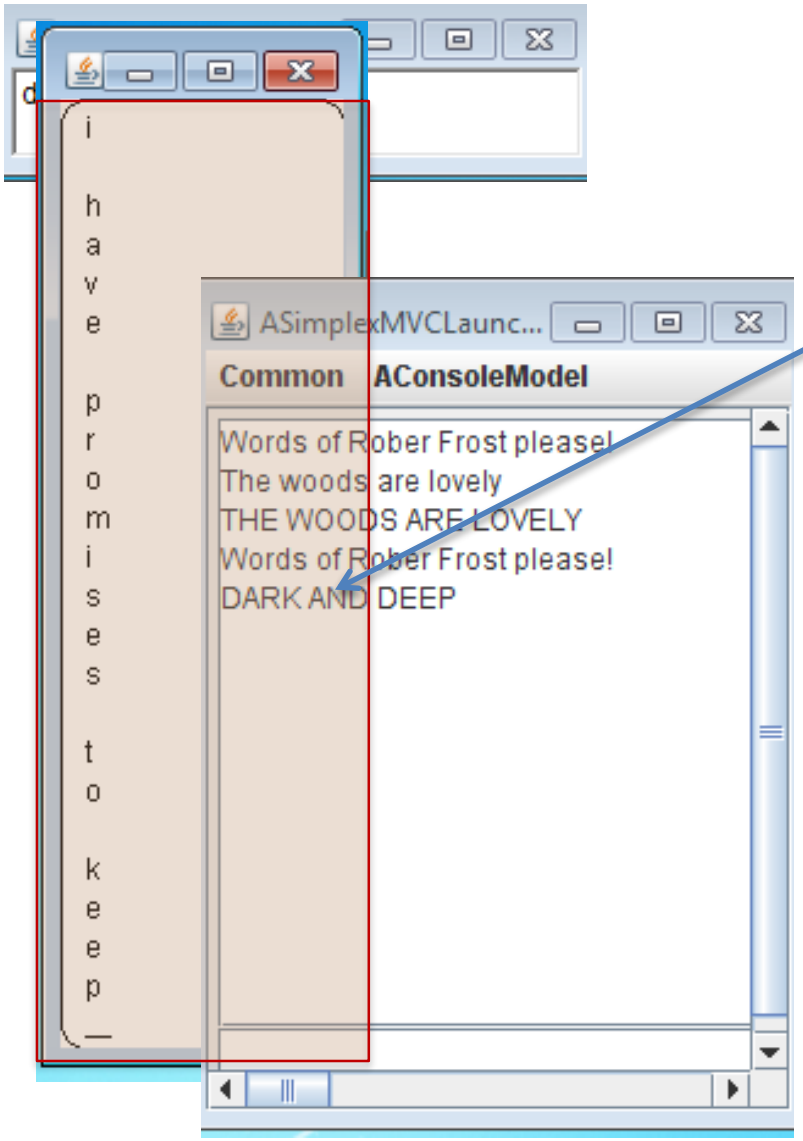
Exposed rectangle

Overlapping Windows (Vertical Window on Top)



Multiple exposed rectangles

Single Exposed



Single exposed rectangle

Drawing outside exposed region will be clipped

Window system could also keep last drawn pixels of each window as backing store

Repainting trades off time efficiency for space

Important if windows are lightweight and nested

Calling Painting Code

```
 g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER,
DIAMETER);
 g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
 g.drawString("" + lastChar, charX, charY);
```

Called when?

When what is to be drawn changes

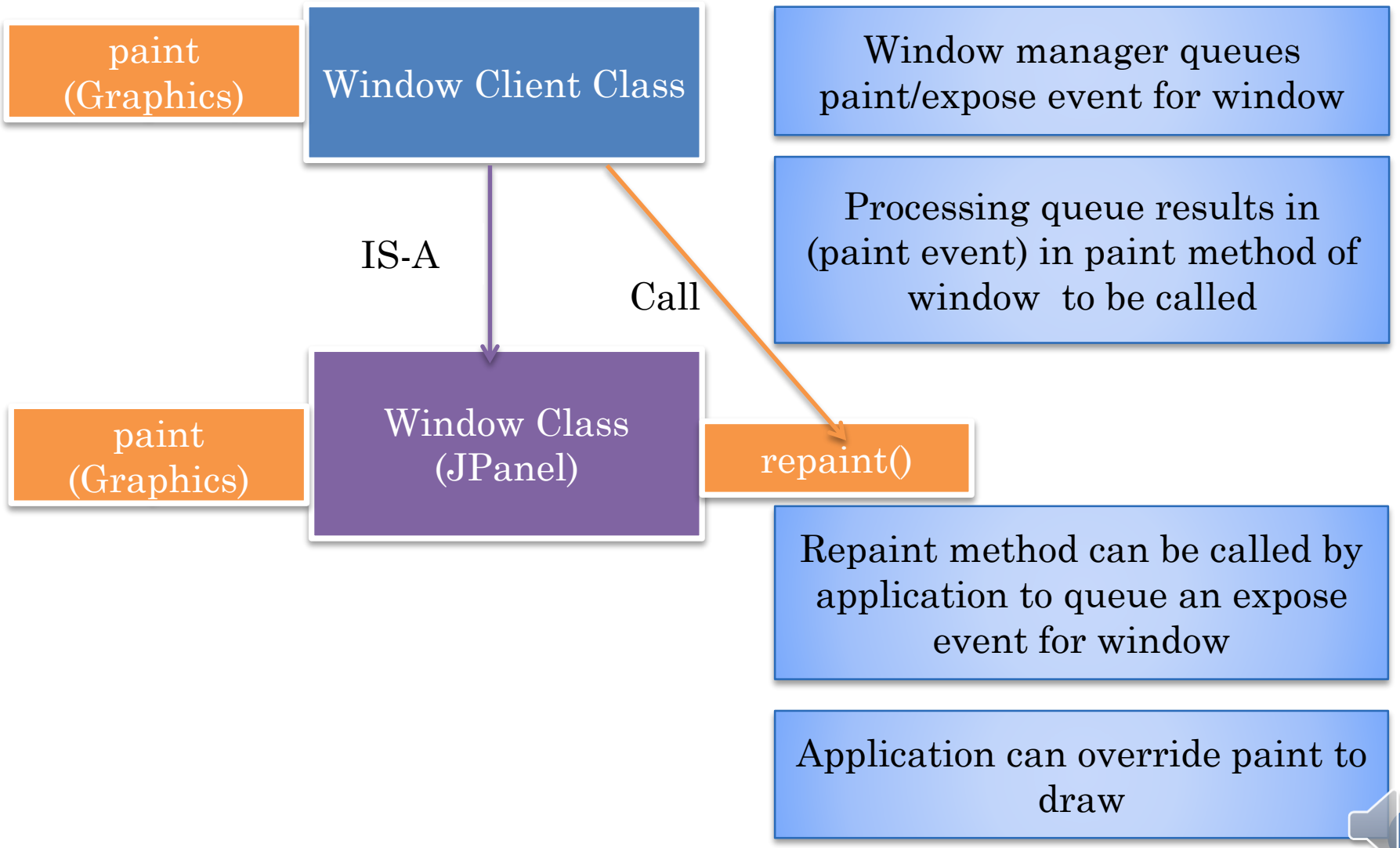
What is the Java API for receiving expose events?

lastChar, charX, charY

How to accommodate both?

When exposed area changes?

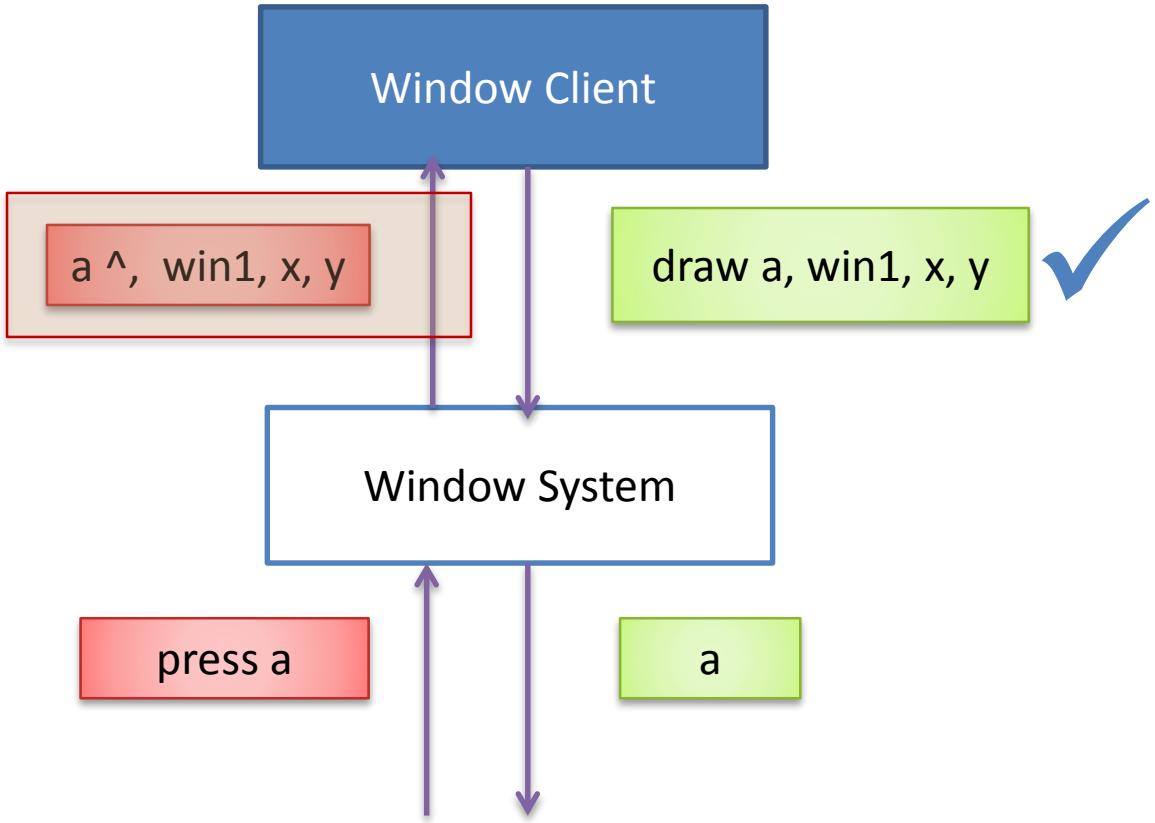
AWT based Expose Event Processing



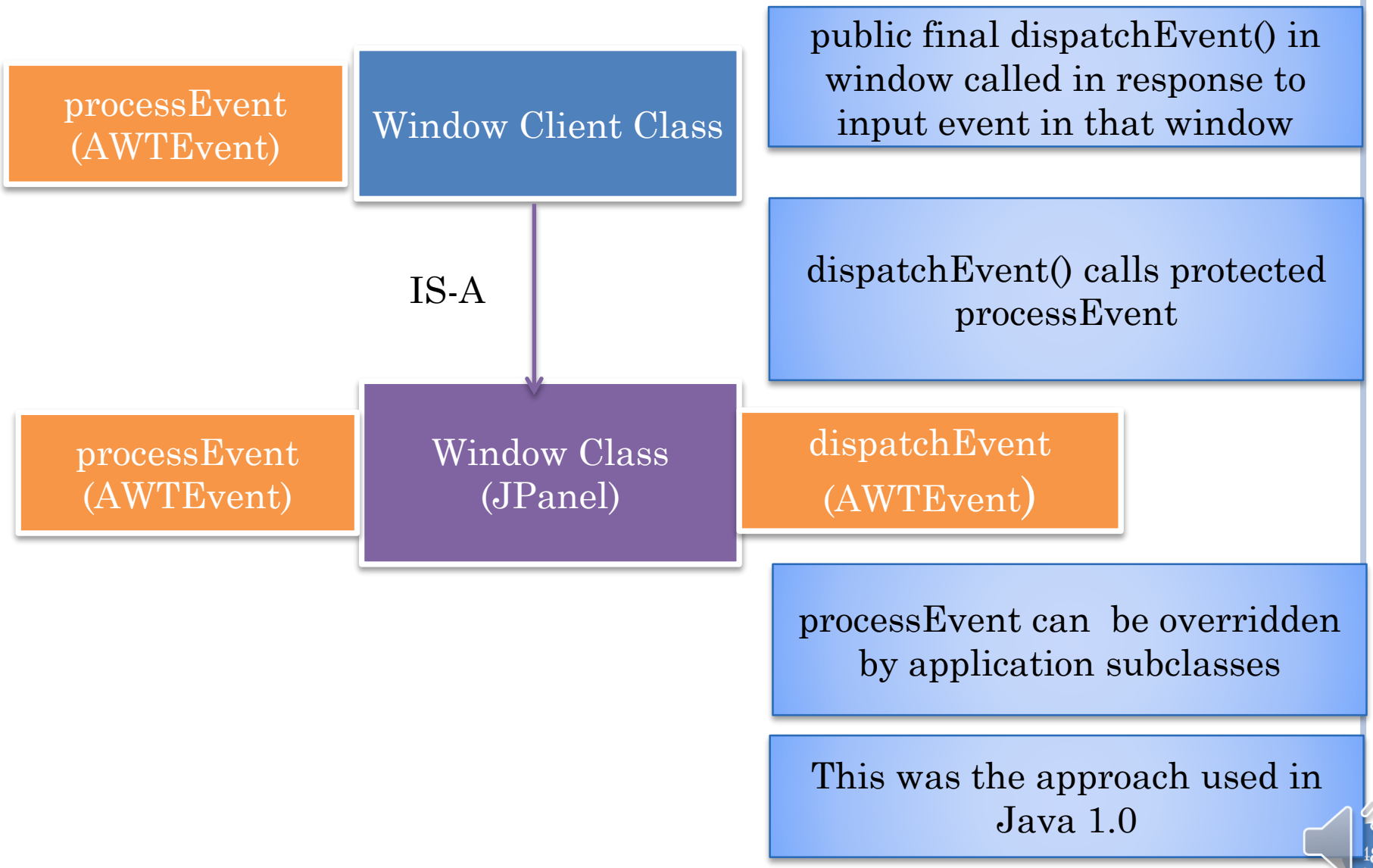
Overriding Paint Method

```
public class ACircledCharacterDrawer extends JFrame implements
MouseListener, KeyListener  {
…
// called when an enqueued paint event for this component is dequeued
public void paint (Graphics g) {
  super.paint(g); // clears the window
  // better to use FontMetrics to center circle
  g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER, DIAMETER);
  g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
  g.drawString("" + lastChar, charX, charY);
}
public void keyTyped(KeyEvent event) {
  setChar(event.getKeyChar());
}
public void setChar(char newValue) {
  lastChar = newValue;
  repaint();// enqueues a paint event
}
public void mousePressed(MouseEvent event) {
  charX = event.getX();
  charY = event.getY();
  repaint(); // enqueues a paint event
}
```

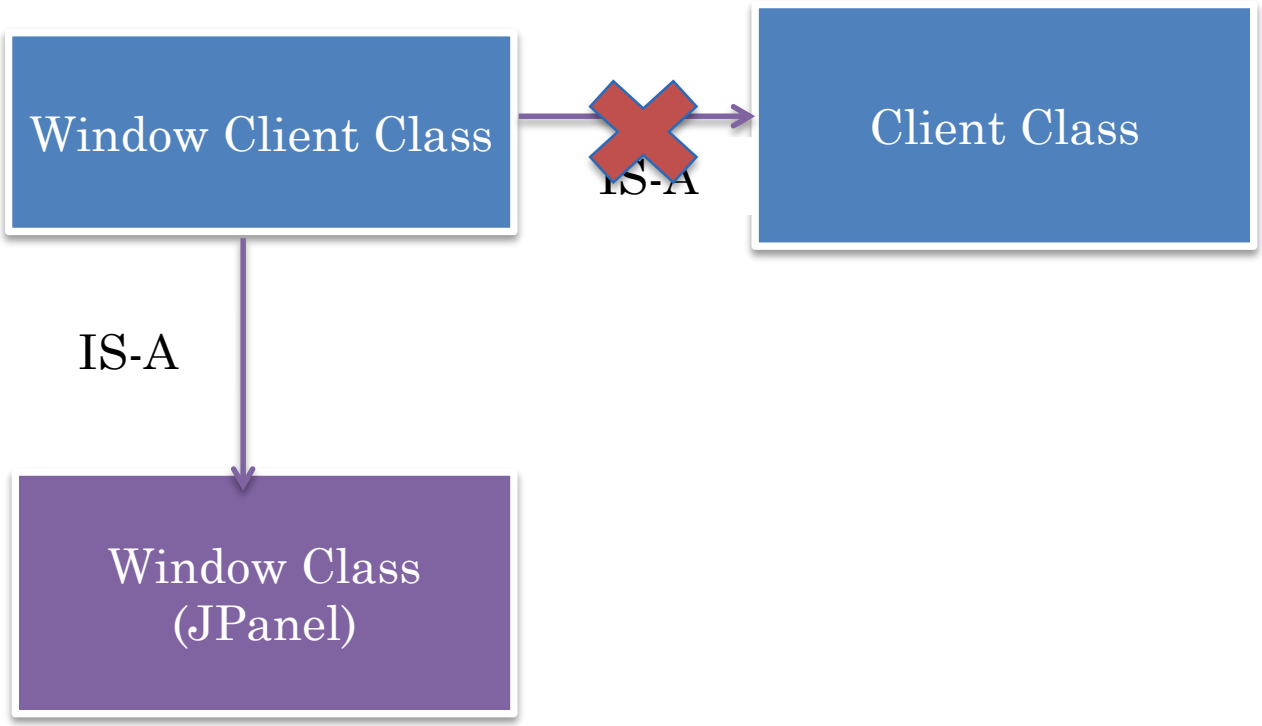
Window I/O: Input



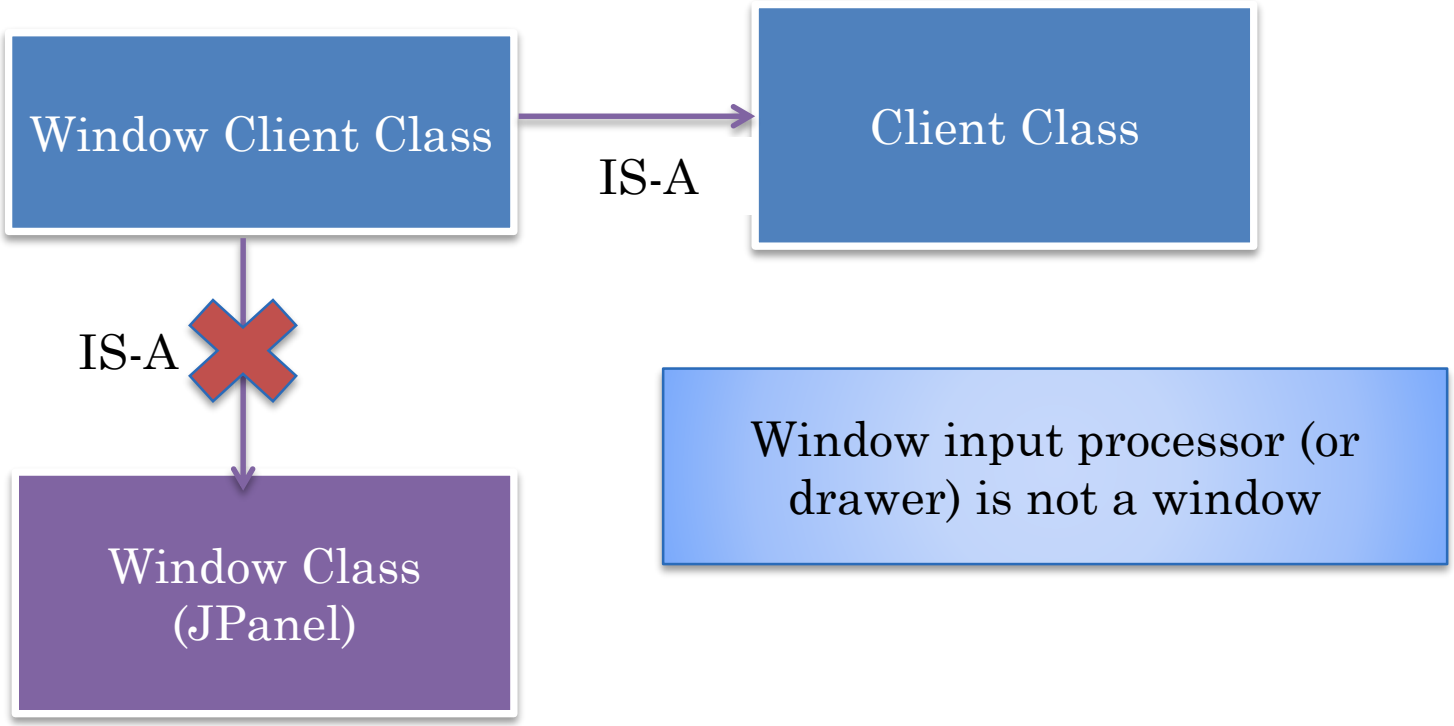
Java Inheritance based Input Event Processing



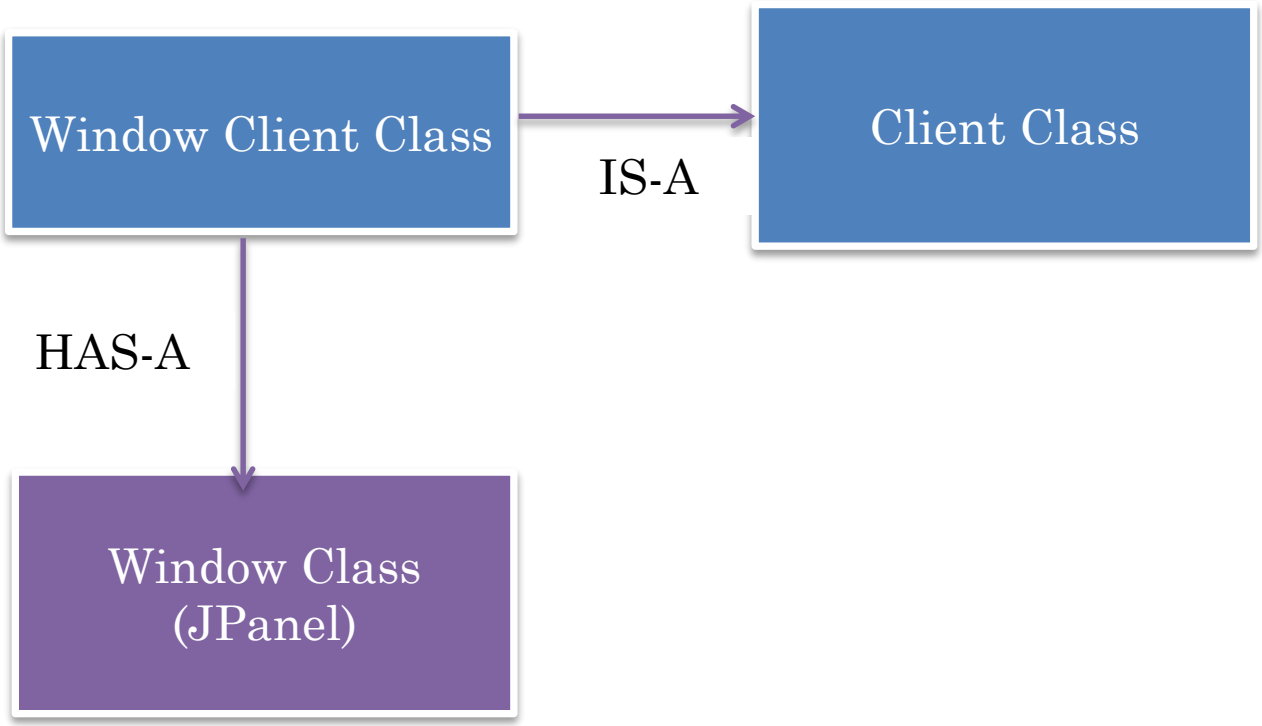
Single-Inheritance Problem



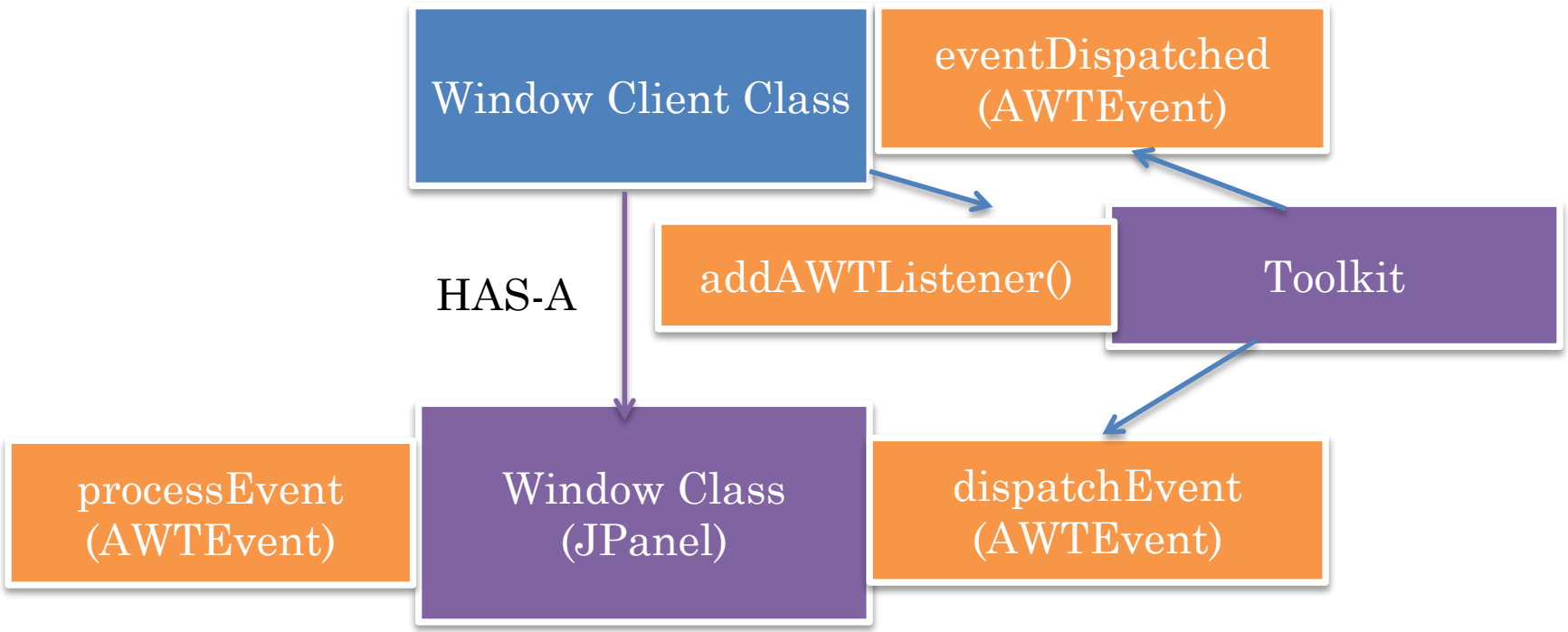
Conceptual Problem with Inheritance



Delegating to window system



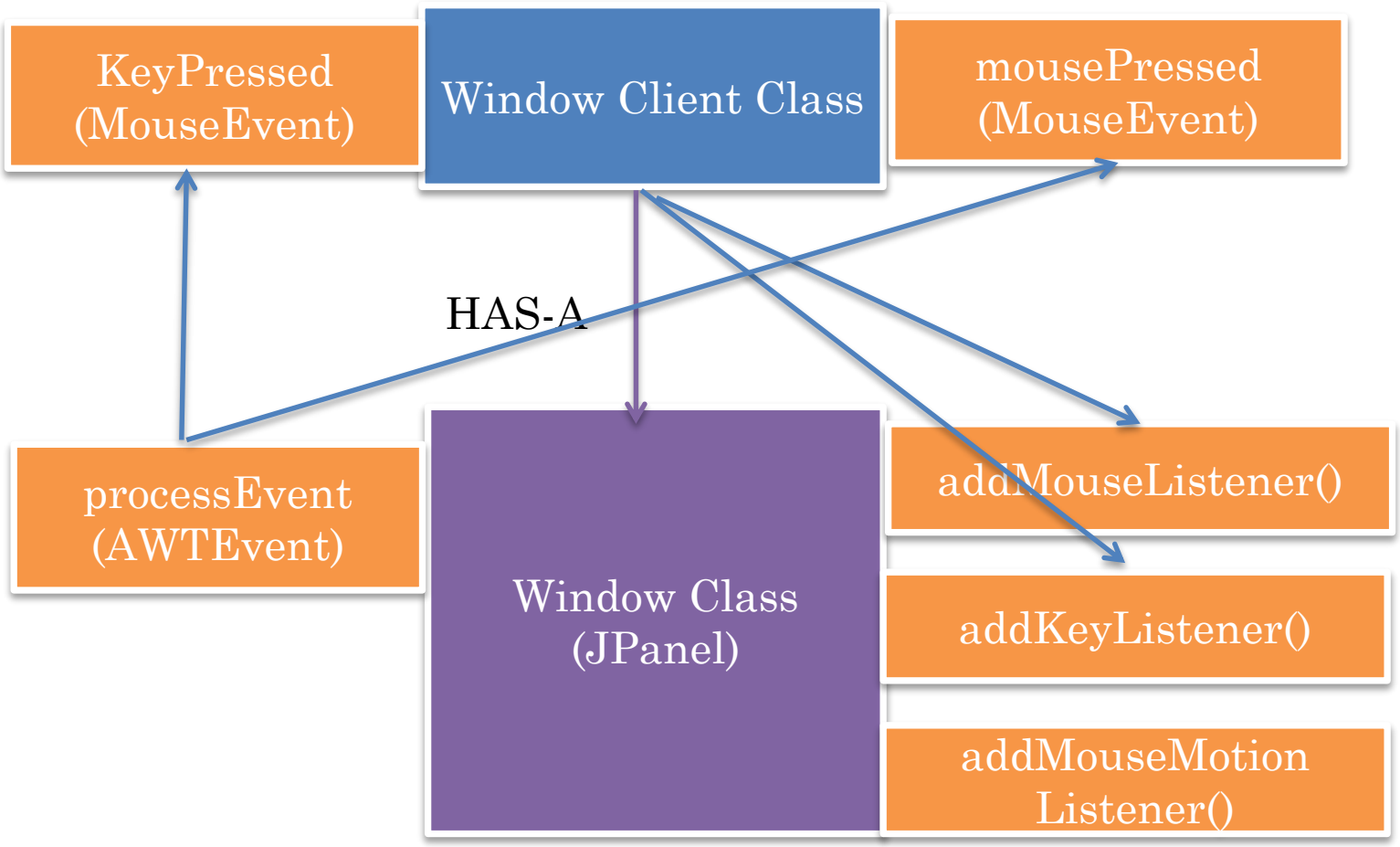
Java Coarse-Grained Delegation-based Input Event Processing



Must distinguish between mouse and event and key pressed, key typed, mousepressed, mousedragged, and other actions

Single way to get all events and then possibly dispatch them – useful for sharing events and telepointer

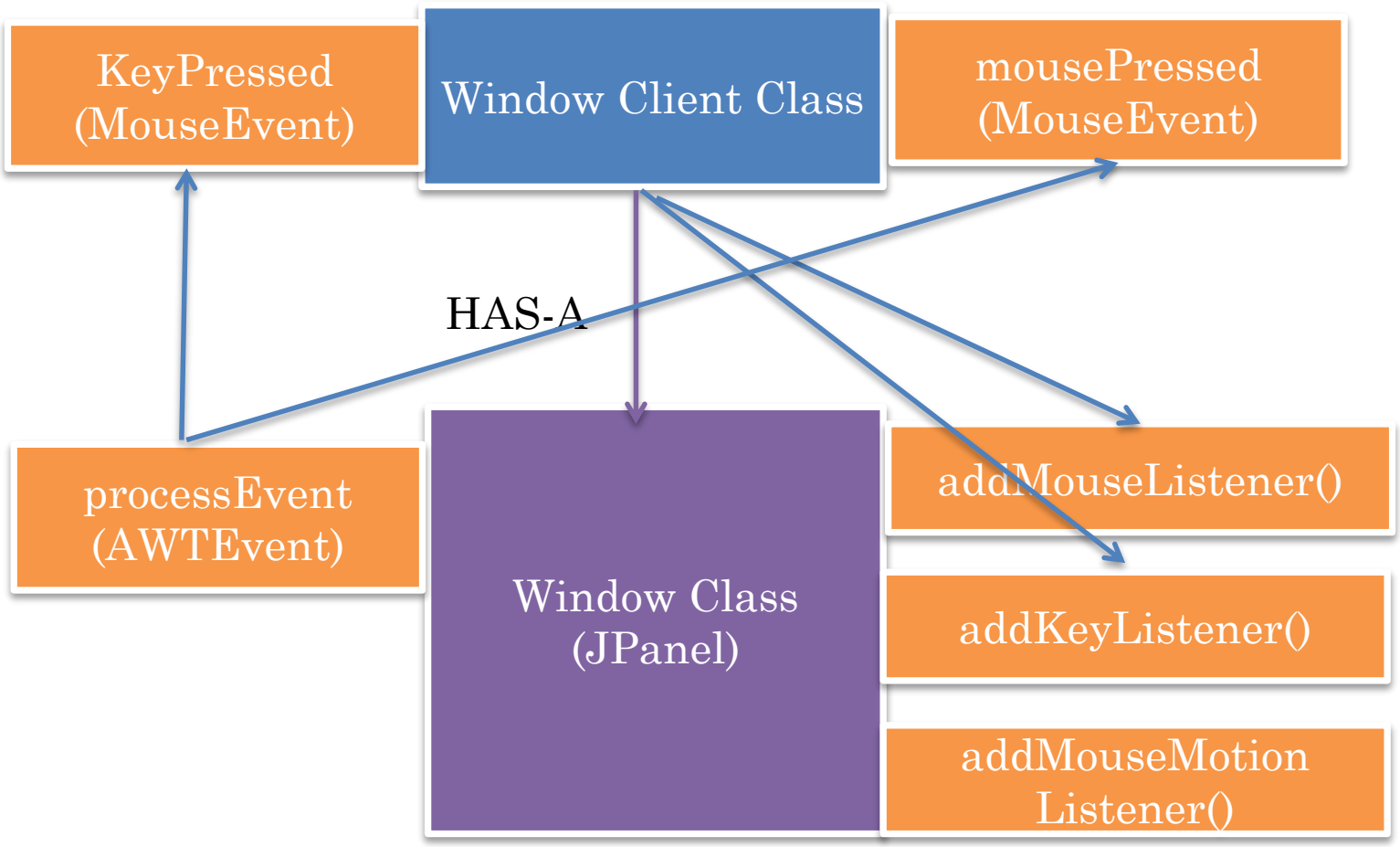
Higher-Level, Per-Window Listeners



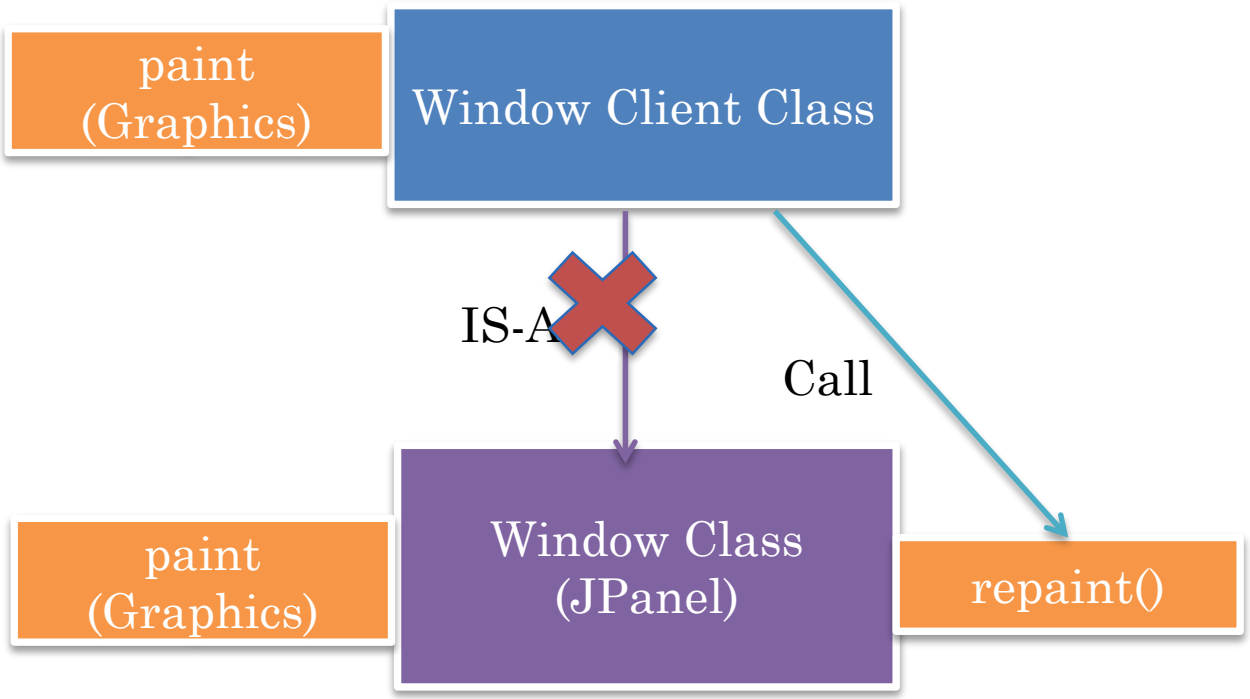
Fine-Grained Delegation Model

```
public class ACircledCharacterDrawer extends JFrame implements
MouseListener, KeyListener  {
public ACircledCharacterDrawer() {
  addMouseListener(this);
  addKeyListener(this);
}
public void keyTyped(KeyEvent event) {
  setChar(event.getKeyChar());
}
public void mousePressed(MouseEvent event) {
  charX = event.getX();
  charY = event.getY();
  repaint(); // enqueues a paint event
}
```

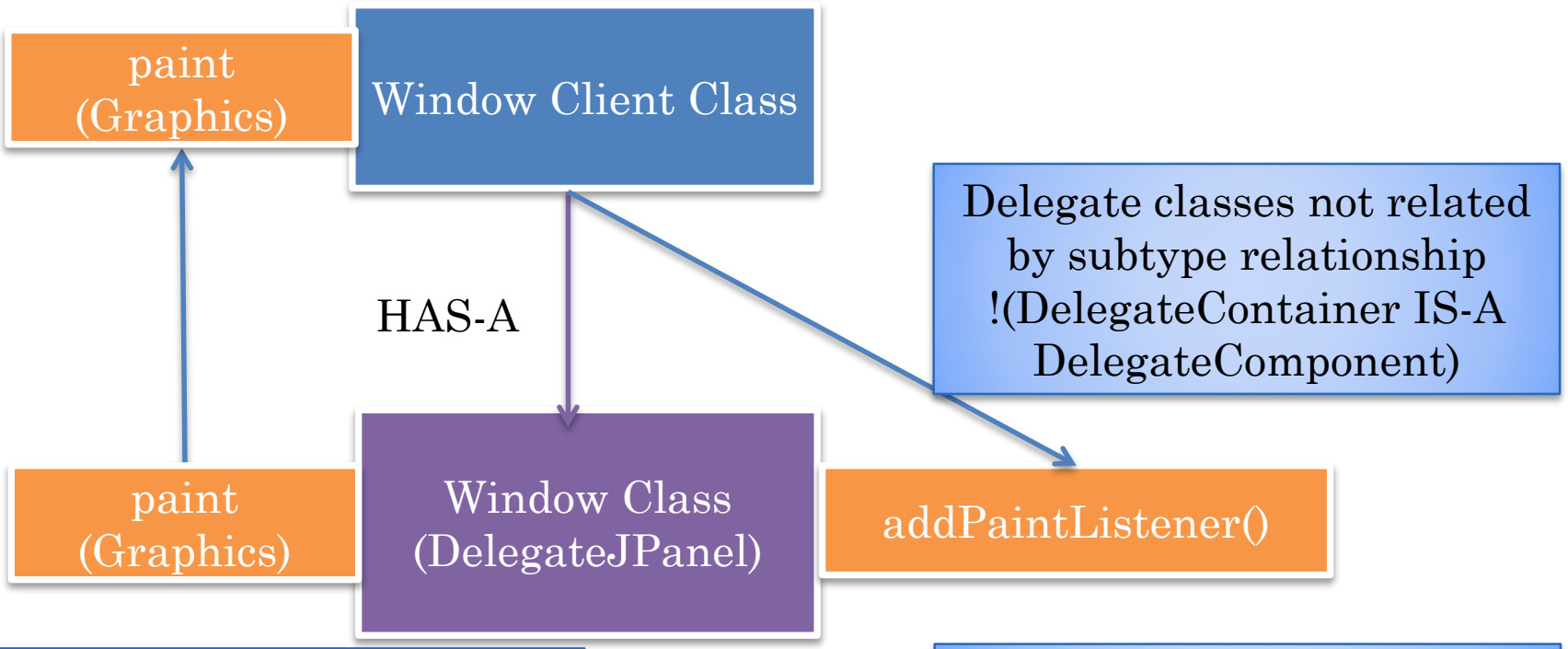
Fine-Grained Implementation



Output Processing: Inheritance



Output Processing: Delegation



The paint method of library classes will call paint methods in delegates

SWT supports delegation model in toolkit

As does Shareable AWT/Swing Library

Understanding Java Window System

```
public class ACircledCharacterDrawer extends JFrame implements
MouseListener, KeyListener  {
…
// called when an enqueued paint event for this component is dequeued
public void paint (Graphics g) {
  super.paint(g); // clears the window
  // better to use FontMetrics to center circle
  g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER, DIAMETER);
  g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
  g.drawString("" + lastChar, charX, charY);
}
public void keyTyped(KeyEvent event) {
  setChar(event.getKeyChar());
}
public void setChar(char newValue) {
  lastChar = newValue;
  repaint();// enqueues a paint event
}
public void mousePressed(MouseEvent event) {
  charX = event.getX();
  charY = event.getY();
  repaint(); // enqueues a paint event
}
```

System-Specific Issues

How to create a telepointer?

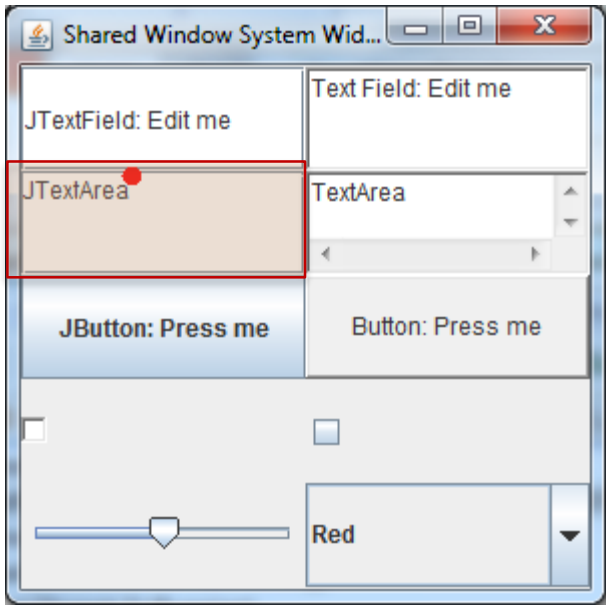
How to intercept input for broadcast?

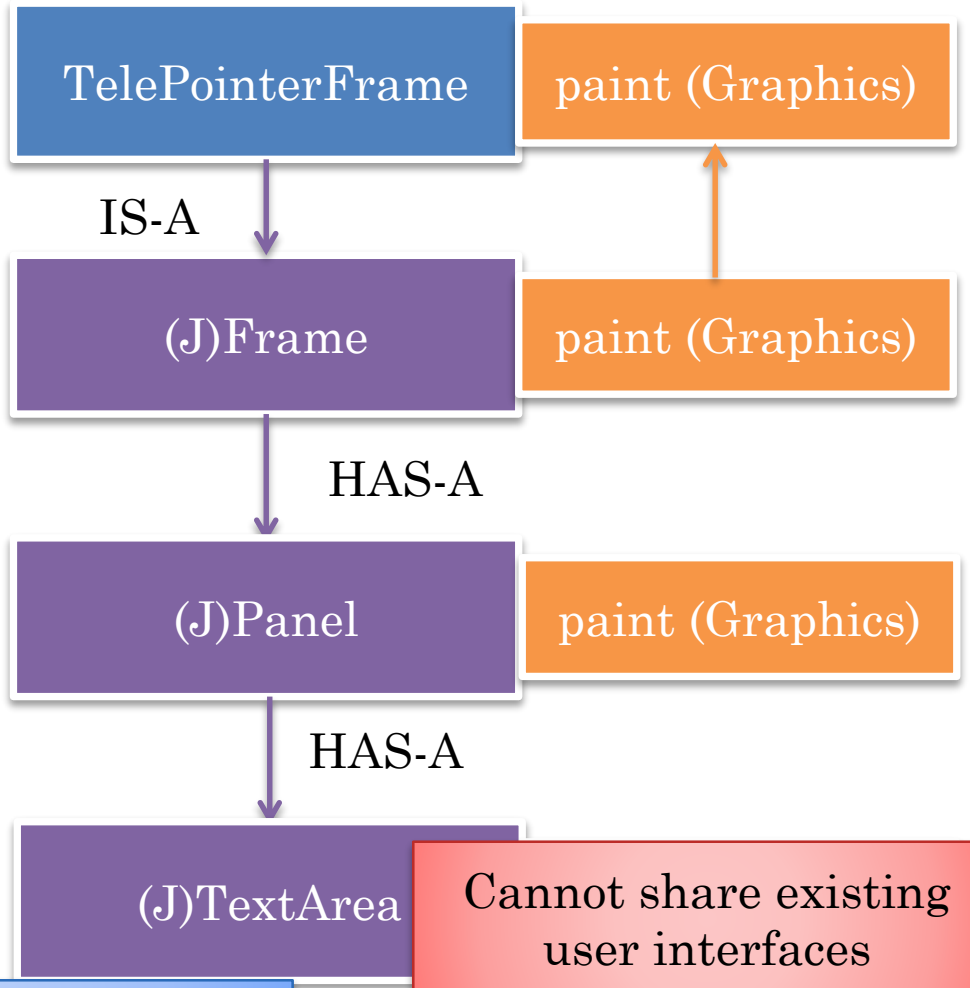
How to inject received input?

How to translate window IDs?

How to filter events?

How to Create A Window TelePointer





A component can be painted by it and all of its ancestors

A key or mouse event in a component is also an event in all of its ancestors

Cannot share existing user interfaces

Nesting: smaller component overrides drawing and input processing of enclosing static components

Cannot use nesting to draw telepointer

How to Create a Window Telepointer?

Replace the top-level frame's window with one that draws movable telepointer shape?

Cannot use nesting to draw telepointer

Even if we could, cannot share existing user interfaces

Layering vs. Nesting

Frame

setGlassPane()

Frame and components

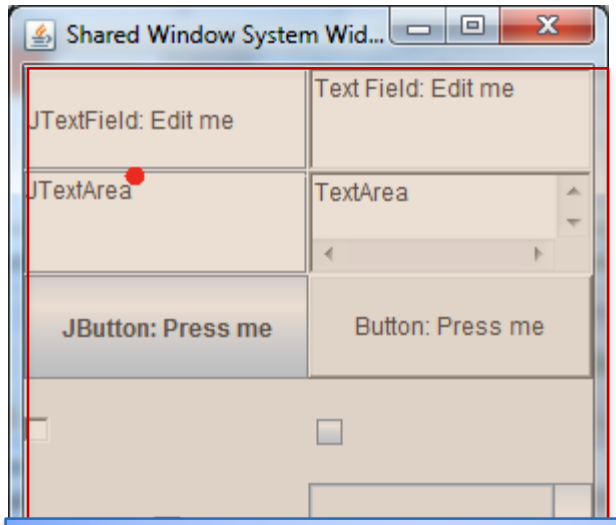
Glass Pane

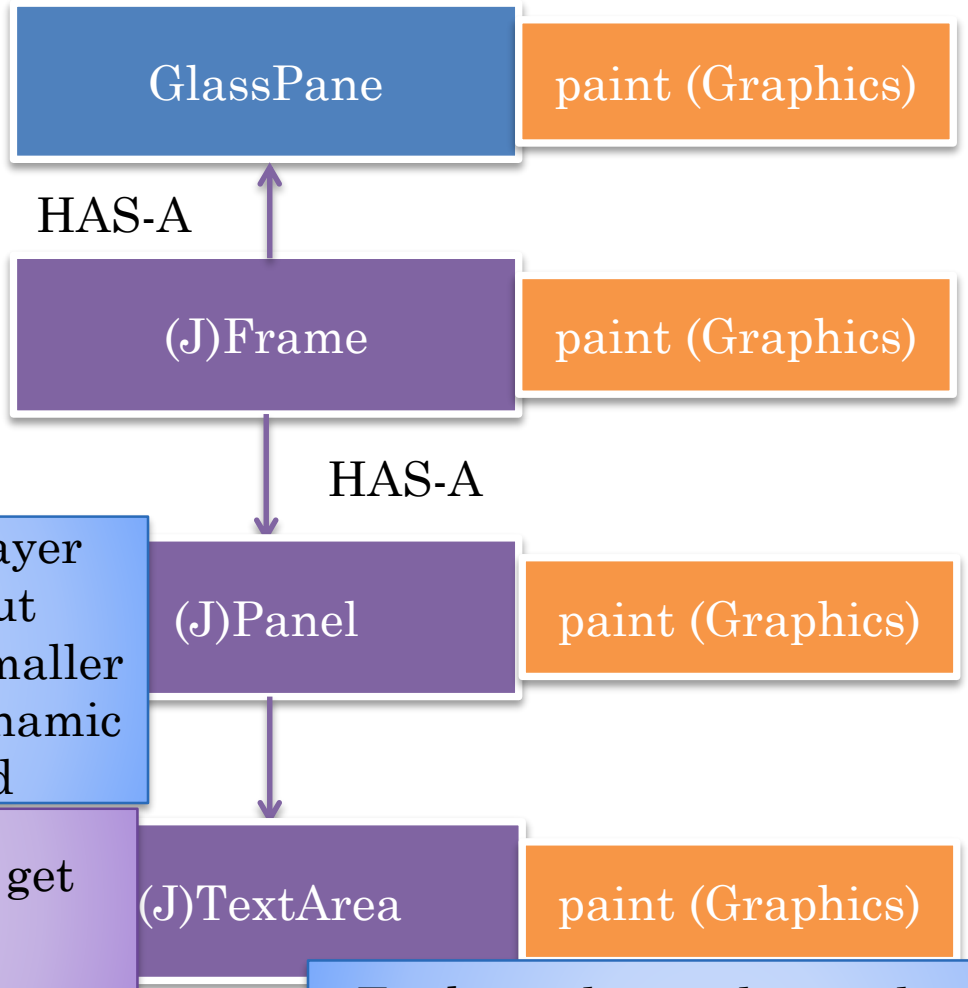
Nesting: smaller component overrides drawing and input processing of enclosing static components

Layering: Higher dynamic layer overrides drawing and input processing of lower, possibly smaller components. Can simulate dynamic multiple parents of a child

Type based vs. structure based overriding

How to Create A Window TelePointer





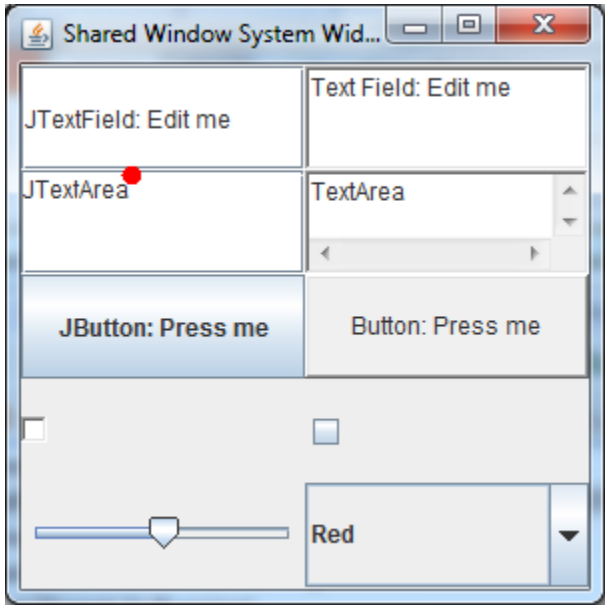
Layering: Higher dynamic layer overrides drawing and input processing of lower, possibly smaller components. Can simulate dynamic multiple parents of a child

How do children components get events?

Glass pane will consume its events (drag of telepointer) and re-dispatch others to deepest children

Tricky and my solution does not always work (menus) – based on code found on the web

How to Create A Window TelePointer Abstraction?



TelepointerUtility

attach(Frame)

Cannot draw our own shape

GraphicsPainter

paint (Graphics)

AnExtendibleTelePointerGlassPane (Frame)

AnExtendible TelePointer GlassPane

addPainter (GraphicsPainter)

getPointerX/Y()

GlassPaneController getGlassPaneController()

Attaching a Telepointer and Painter

```
public interface GraphicsPainter {
  void paint(Graphics g);
}
```

```
glassPane = new AnExtendibleTelePointerGlassPane(telePointedFrame);
glassPane.addPainter(createTelePointerPainter());
```

```
public interface GlassPaneController {
  int getPointerSize();
  void setPointerSize(int aSize);
  int getPointerWidth();
  void setPointerWidth(int aWidth);
  int getPointerHeight();
  void setPointerHeight(int aHeight);
  boolean isShowTelePointer();
  void setShowTelePointer(boolean showTelePointer);
}
```

Painter should use the dimensions in controller to draw shape

How to Create a Telepointer

- Instantiate a telepointer glasspane, passing it a JFrame
- Implement a telepointer painter
- Painter should reference the telepointer glass pane to get paint position
- Painter should reference the telepointer controller to get paint dimensions

System-Specific Issues

How to create a telepointer? ✓

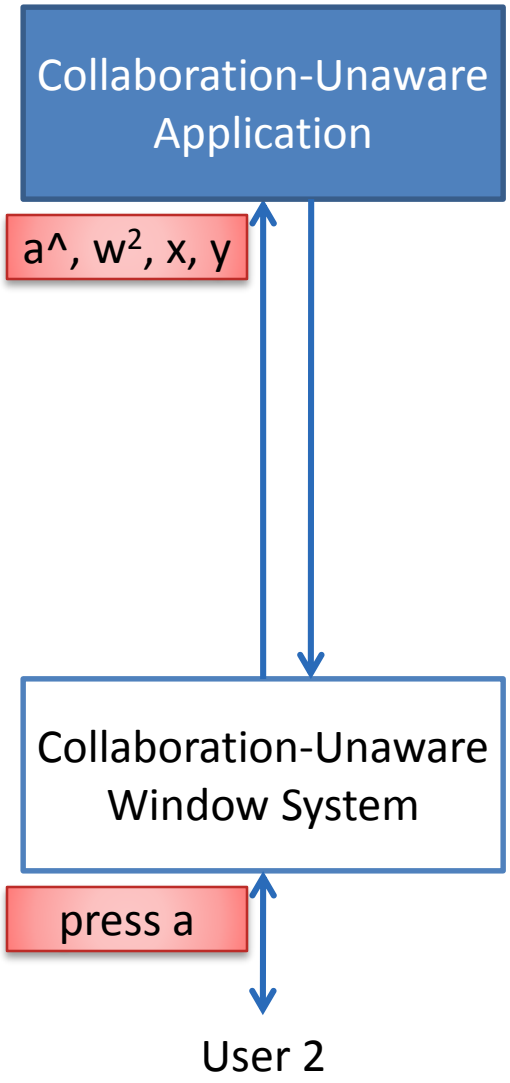
How to intercept input for broadcast?

How to inject received input?

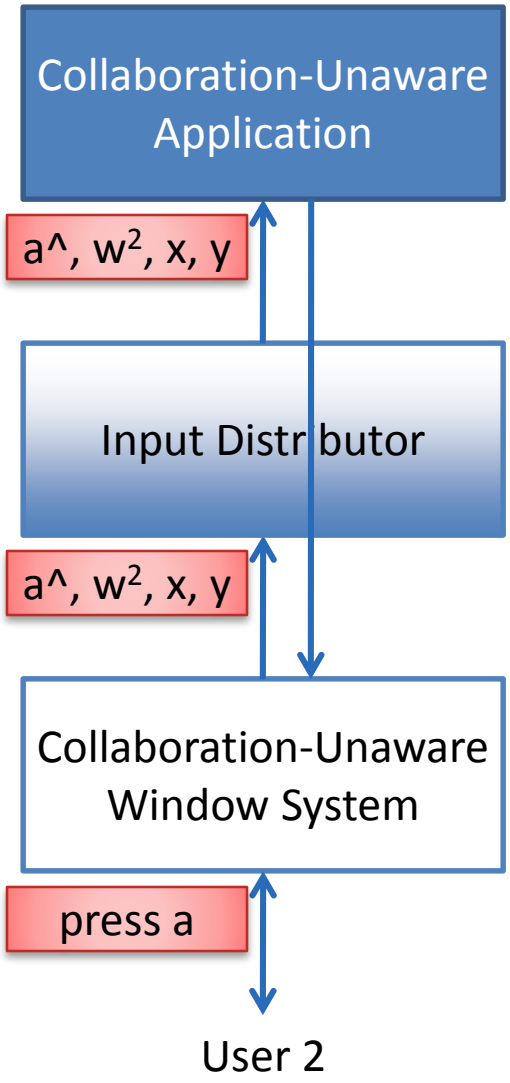
How to translate window IDs?

How to filter events?

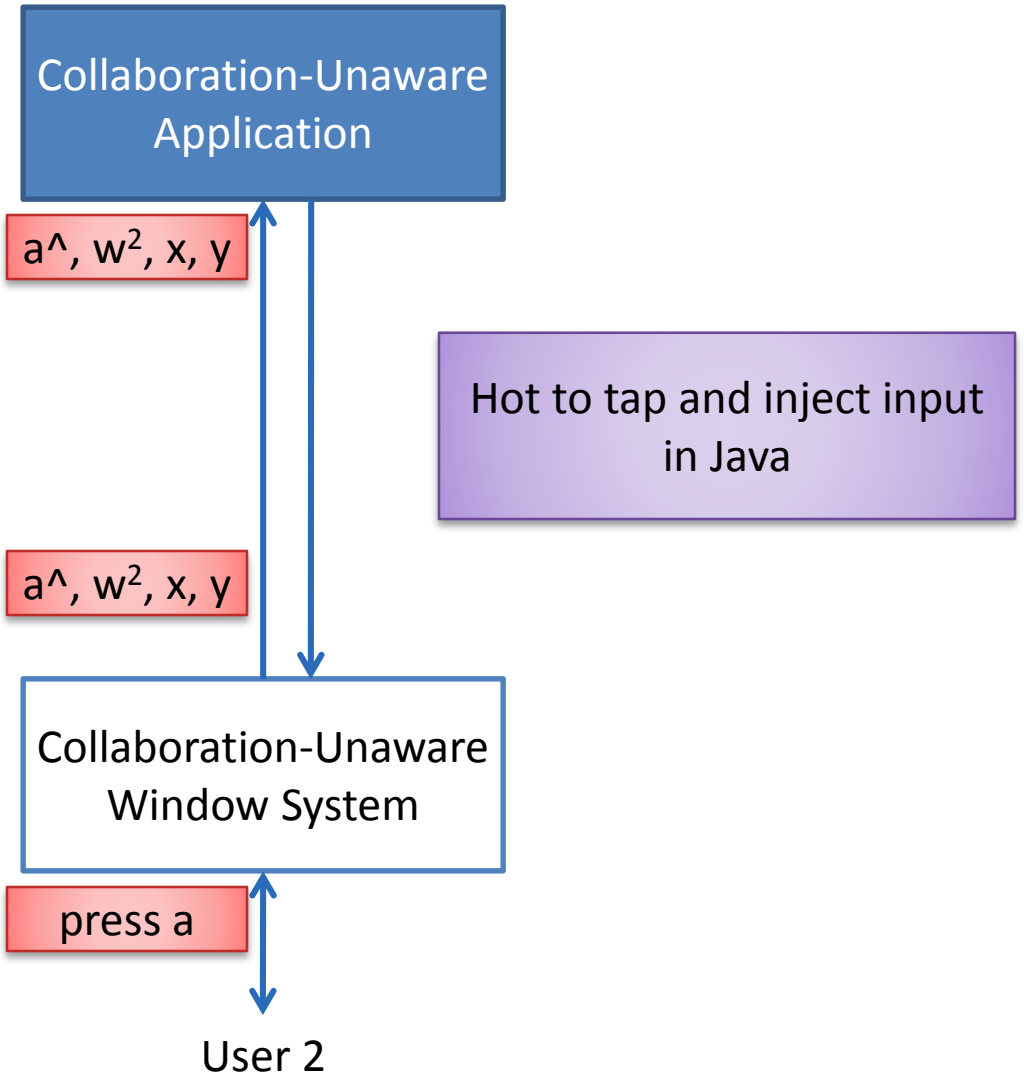
General Model of Single-User Window System



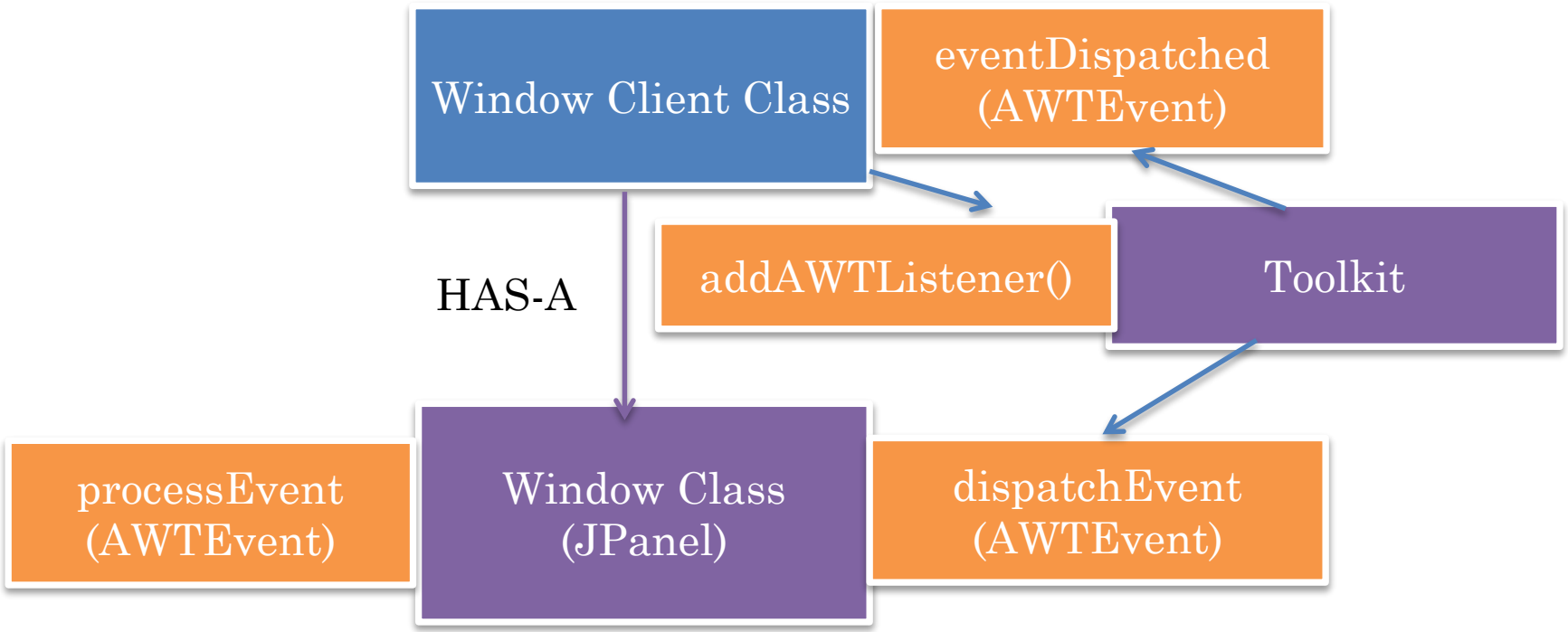
General Model of Replicated Window System



General Model of Single-User Window System Again



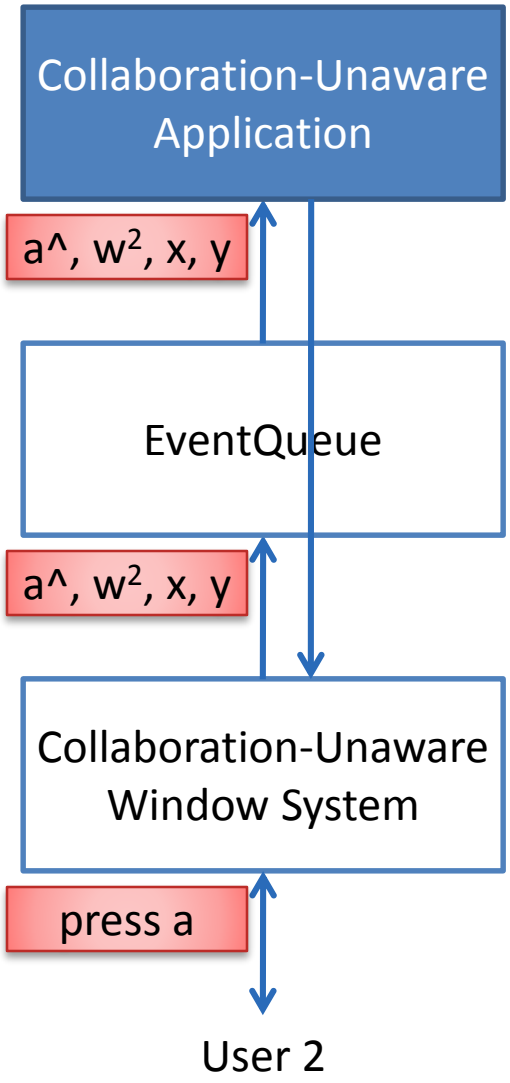
Java Coarse-Grained Delegation-based Input Event Processing



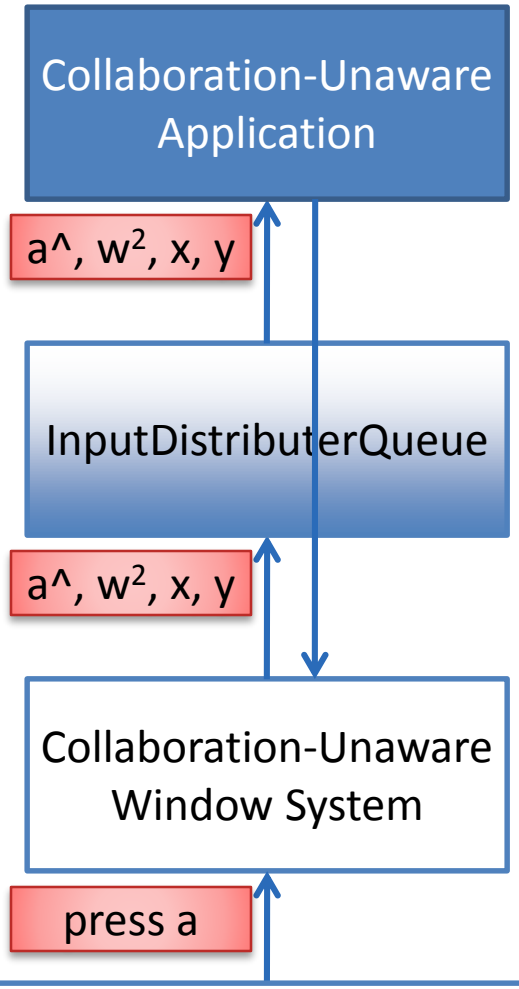
Can intercept events at the same time they are dispatched to local components

No direct way to inject or stop events – need to inspect event and send it to appropriate component

Java Input Queue



Replacing Queue



Can intercept events before they are dispatched to application

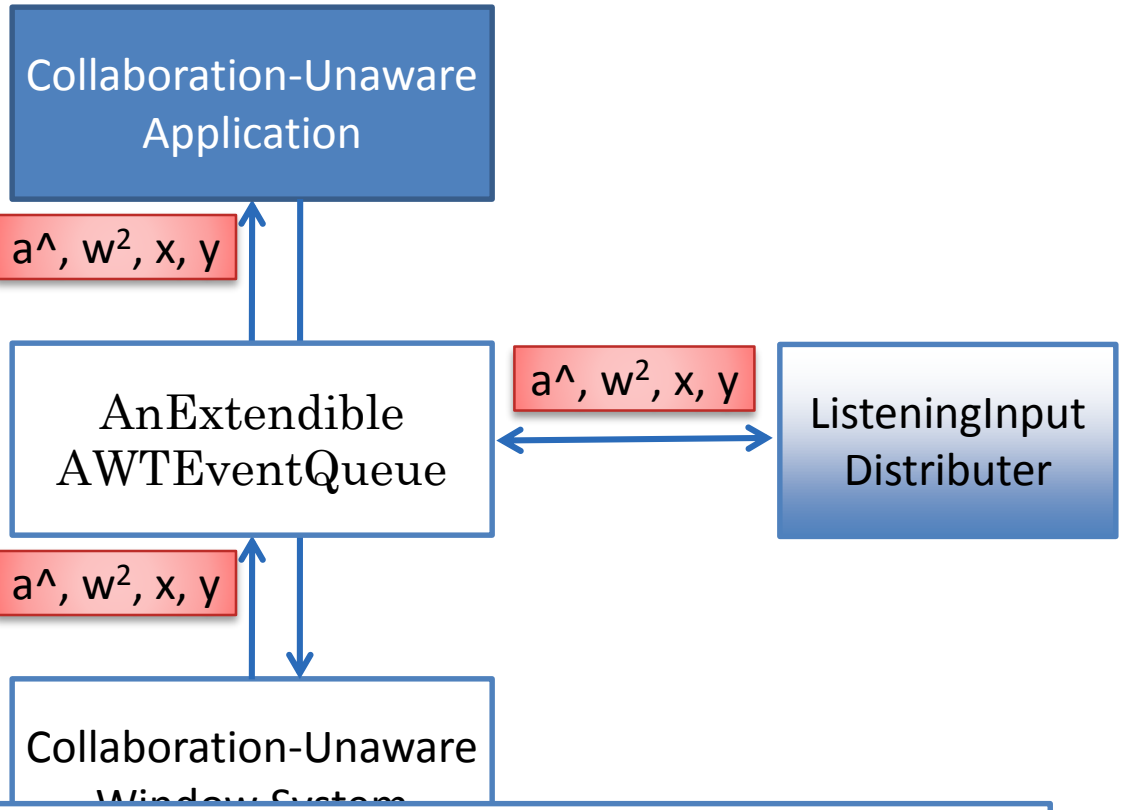
Must remember to forward to application

Must prevent cycles

Must filter uncoupled events

```
Toolkit.getDefaultToolkit().getSystemEventQueue().push(
                                    new InputDistributingQueue()) ;
```

Library Listenable Event Queue




```
Toolkit.getDefaultToolkit().getSystemEventQueue().push(
                                                this)
```

```
AnExtendibleAWTEventQueue.getEventQueue().
        addEventQueueHandler(new ListentingInputDistributer());
```

How to Intercept and Inject Window Events

- Window Manager State
- Window Size
- Screen Pointer
- Mouse Move
- Key Click
- Mouse Click
- Mouse Drag

Will define a listener to get filtered events

AWTEventQueue Handler

newEvent (AWTEvent)

Inject event?

Singleton class, invoke static method in it to get global queue

static getEventQueue()

AnExtendible AWTEventQueue

addEventQueueHandler (AWTEventQueueHandler)

getCommunication EventSupport()

dispatchReceivedEvent (AWTEvent)

Will convert serializable event to local event

Does not fire new event

Listenable, Injectable Event Queue

```
package util.awt;
public interface ExtendibleAWTEventQueue extends
PropertyVetoerRegistrar {
  public void addEventQueueHandler(AWTEventQueueHandler listener);
  public void removeEventQueueHandler(AWTEventQueueHandler listener);
  public void clearEventQueuehandlers();
  public void dispatchEvent(AWTEvent event);
  void dispatchReceivedEvent(AWTEvent anEvent);
}
```

dispatchEvent vs. dispatchReceivedEvent ~ replicatedAdd vs. observableAdd

System-Specific Issues

How to create a telepointer? ✓

How to intercept input for broadcast? ✓

How to inject received input? ✓

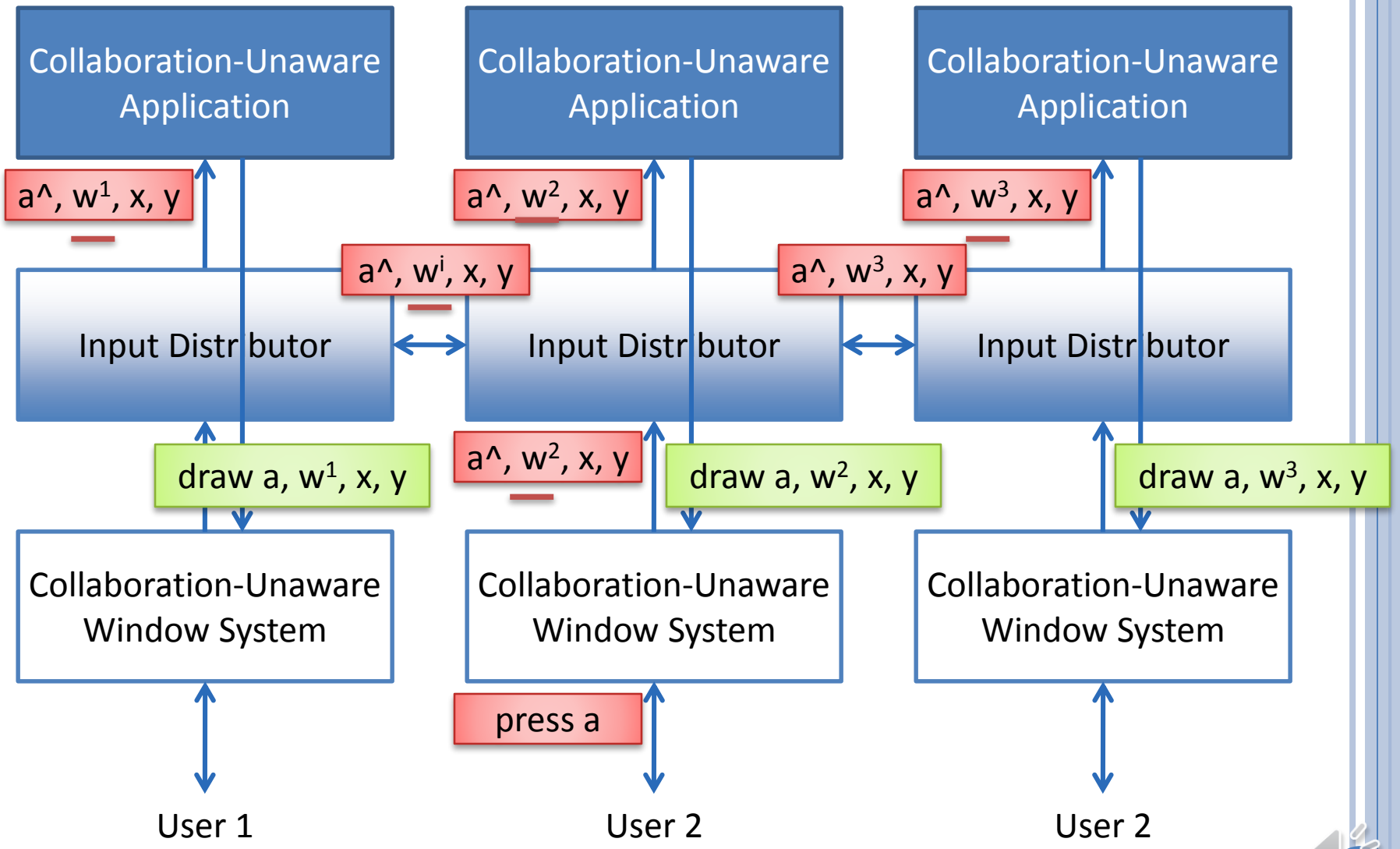
How to translate window IDs?

How to filter events?

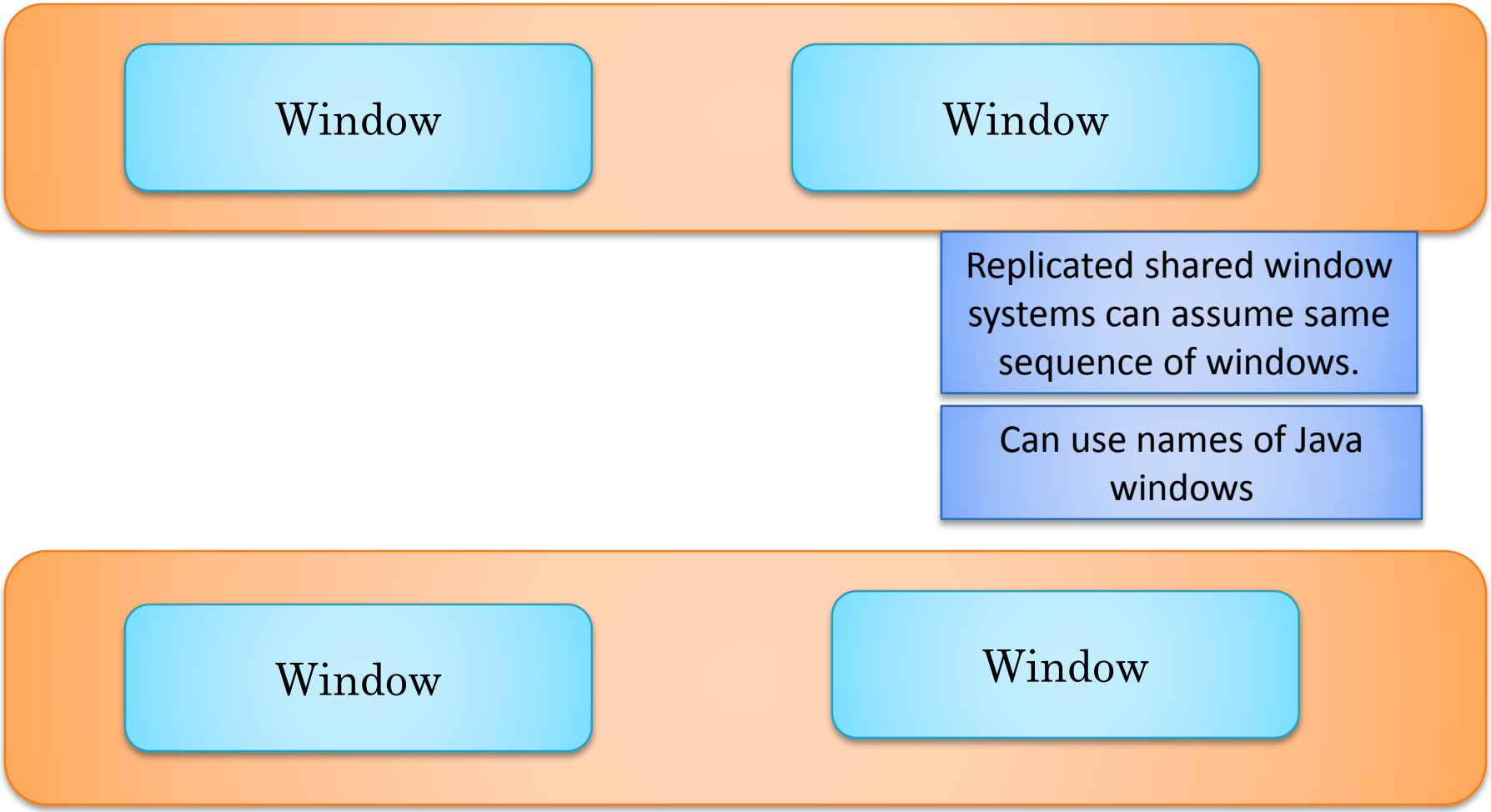
Translate Window IDs

How to find corresponding windows in different replicas?

Translating Window Ids



Connecting Replica Windows



How to replace local window ids (Component instances) in events with global ids (integer, string) and global ids in events with local ids?

Translation

Given AWTEvent e, component c

Send ASerializableAWTEvent (e, toID(c))

ASerializable AWTEvent

ASerializable AWTEvent (AWTEvent, String)

Given SerializableAWTEvent (event e, id i)

Dispatch toDispatchedEvent(e , toComponent(i)) to AnExtendibleAWTEventQueue

CommunicatedAWT EventSupport

AWTEvent toDispatchedEvent(SerializableAWTEvent , Component component);

```
AnExtendibleAWTEventQueue.getEventQueue().getCommunicatedEventSupport()
```

Serializable Event → Local Event

```
package util.awt;
public class ASerializableAWTEvent implements SerializableAWTEvent {
  public ASerializableAWTEvent(AWTEvent theEvent, String theComponentId) {
   …
  }
}
```

```
SerializableAWTEvent serializableEvent (SerializableAWTEvent) aMessage;
AWTEvent aDispatchedEvent =
   AnExtendibleAWTEventQueue.getEventQueue().
       getCommunicatedEventSupport().toDispatchedEvent(
           serializableEvent, toComponent(serializableEvent));
AnExtendibleAWTEventQueue.getEventQueue().dispatchReceivedEvent
            (aDispatchedEvent)
```

toComponent() written by programmer to translate between global id and local component

Translate Window IDs

How to find corresponding windows in different replicas?

How to find the windows and creation sequence in each replica?

Window Creation Event → Register Subtree

Once we find a window, we can recursively find all of its descendants

((Container) component).getComponents();

Resize event sent to EventQueue when it is created

AWTMisc.*isResizeEvent(event);*

If E is window creation (resize) event then register the global ids of its subtree if the subtree has not already been registered

A Tale of Two Resize Events

Resize event sent both when window resized and it is created.

Want to dispatch normal received resize events but not creation events

Connect queue listener before window tree created to get resize events

To prevent window creation events remotely broadcast

- broadcaster can be attached after window tree is created, which means two different listeners
- broadcaster can have a special mode to separate the two phases
- receive listener can be attached after local window tree created

System-Specific Issues

How to create a telepointer? ✓

How to intercept input for broadcast? ✓

How to inject received input? ✓

How to translate window IDs? ✓

How to filter events?

Transmit Function

Window Manager Events

Filter by Event Type

Done by library

Filter by (Top) Window Name

Object event.getSource();

SwingUtilities.getRoot (Component)

Need to cast source as Component

Mouse Drag

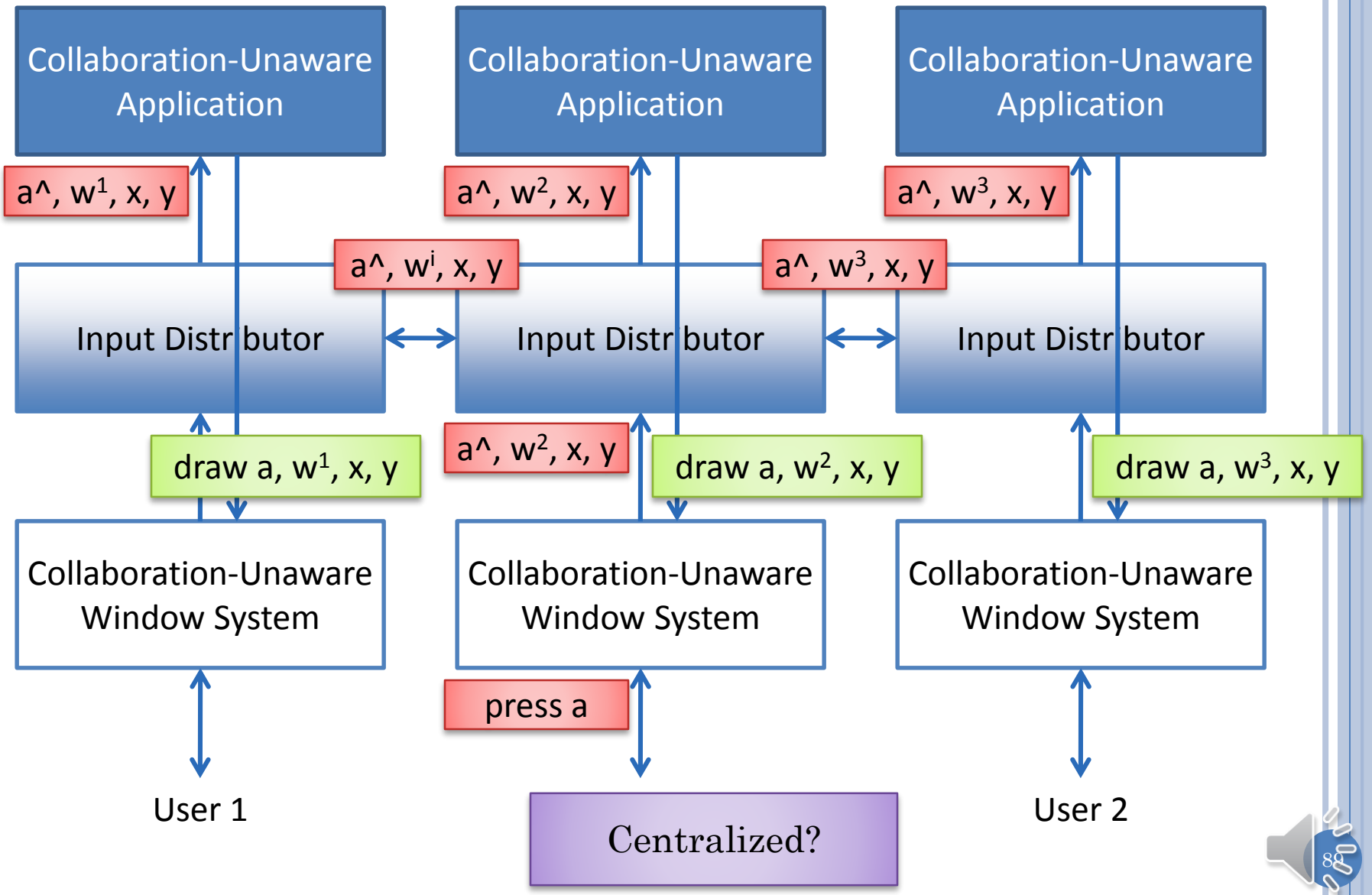
Filter by Event Time

System.*currentTimeMillis();*

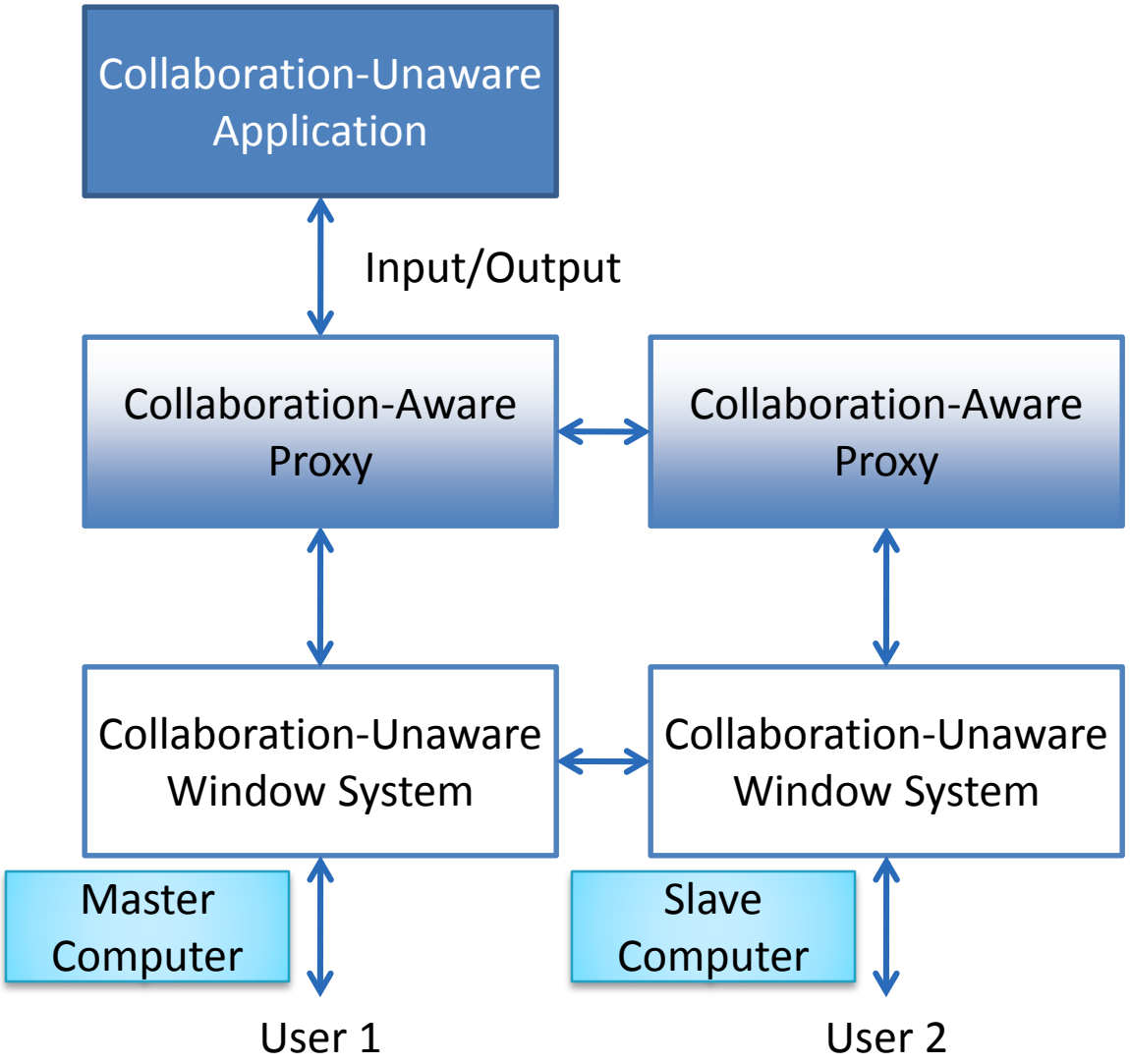
AWTMisc.*isMouseDragged Event(event);*

Need to ensure that last mouse drag event is sent

Replicated vs. Centralized Window System



Centralized Shared Window System



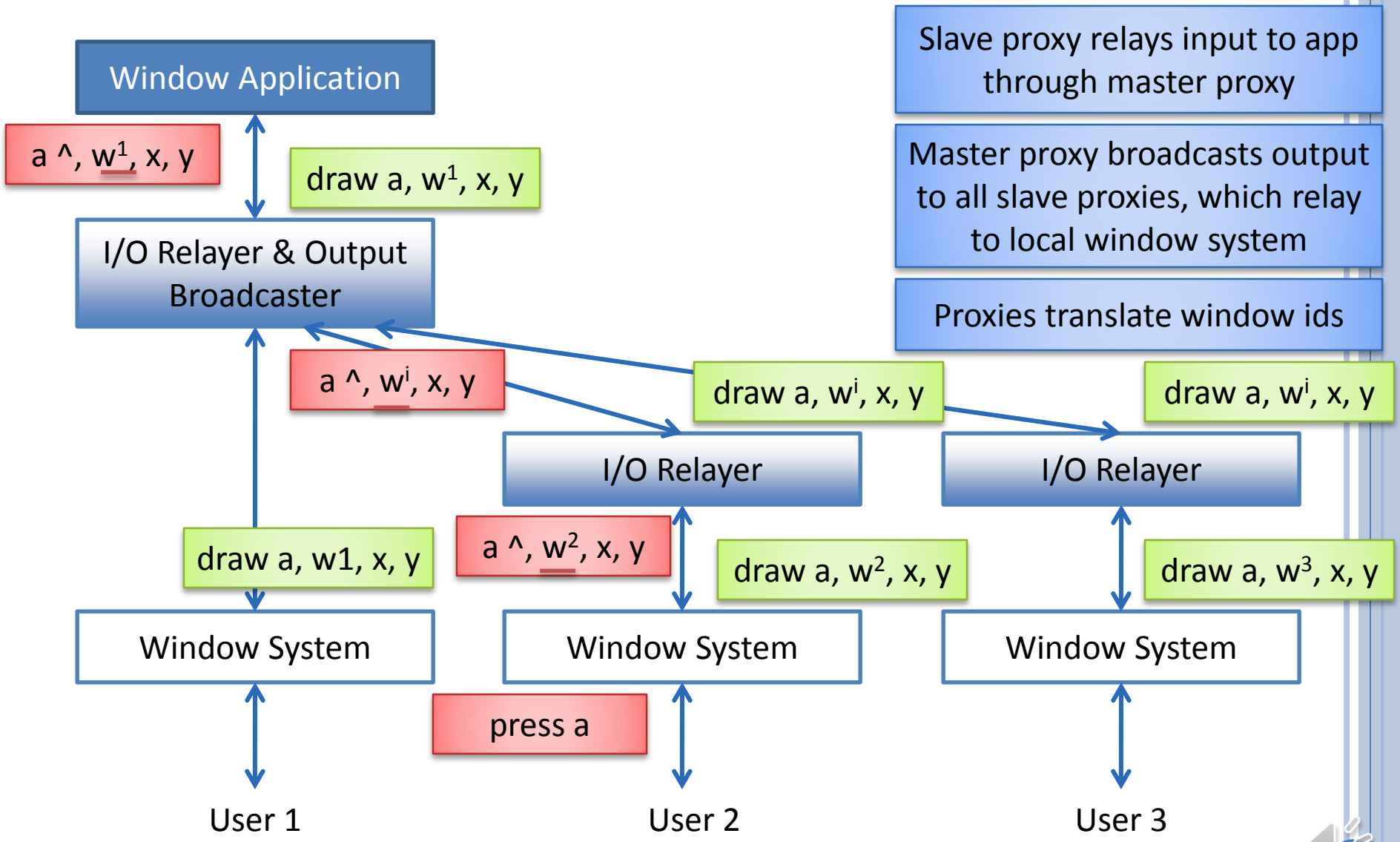
The shared application runs on the (master) computer of only one of the collaborators

The shared application runs on the (master) computer of only one of the collaborators

Each user's input relayed to application through proxies

Each output of application broadcast to all users through proxies

Proxy Functions



Replicated Window Algorithm

For each Window w, create Telepointer w^t

For each Window (including Telepointer) Input I

I should be followed by matching WindowEventDispatched

For each replica, if Transmit(I) then I should be followed by matching WindowEventSent

How to change it to centralized?

For each WindowEventReceived R

R should be followed by matching WIndowEventDispatched

For each replica, R should be followed by matching WindowEventSent

Centralized Window Algorithm

Master and Slave

For each Window w, create Telepointer w^t

For each Window (including Telepointer) Event I

I should be followed by matching WindowEventDispatched (including Telepointer)

If isSlave() and Transmit(I) then I should be followed by matching WindowEventSent to Master

Master Receiver

For each WindowEventReceived R at Master, R should be followed by matching WindowEventDispatched

For each output call O, O should be followed by WindowRequestMade and WindowRequestSent to all Slaves

Slave Receiver

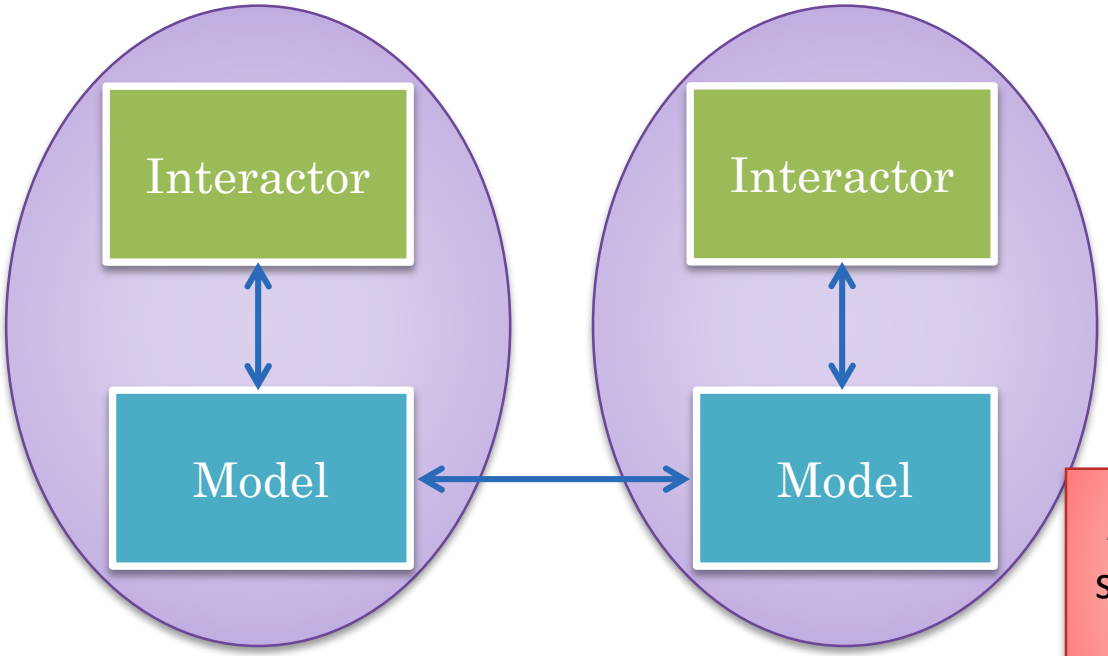
For each WindowRequestReceived R at Slave, R should be followed by WindowRequestMade

Centralized vs. Replicated Window Systems

Centralized vs. Replicated Shared Window Systems

~ Centralized vs. Replicated Shared Model Systems

Replicated Model: Issues

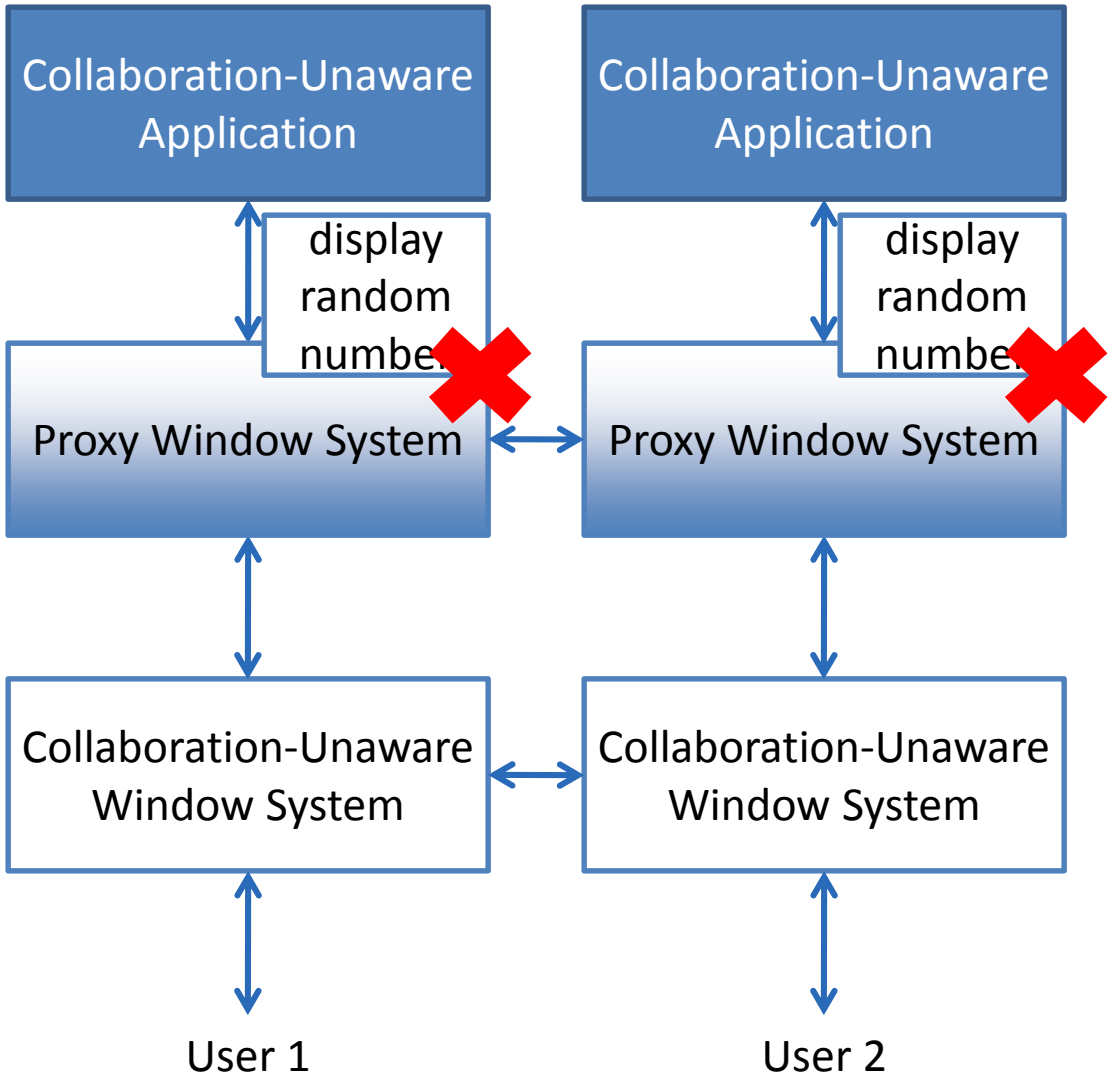


All of these problems still occur in replicated window systems

Consistency issues of causality and concurrent operations (to be addressed later)

Correctness and performance issues when model is non deterministic, accesses central resources, and has side effects

Example of Non Determinism and Other Replication Problems (Review)



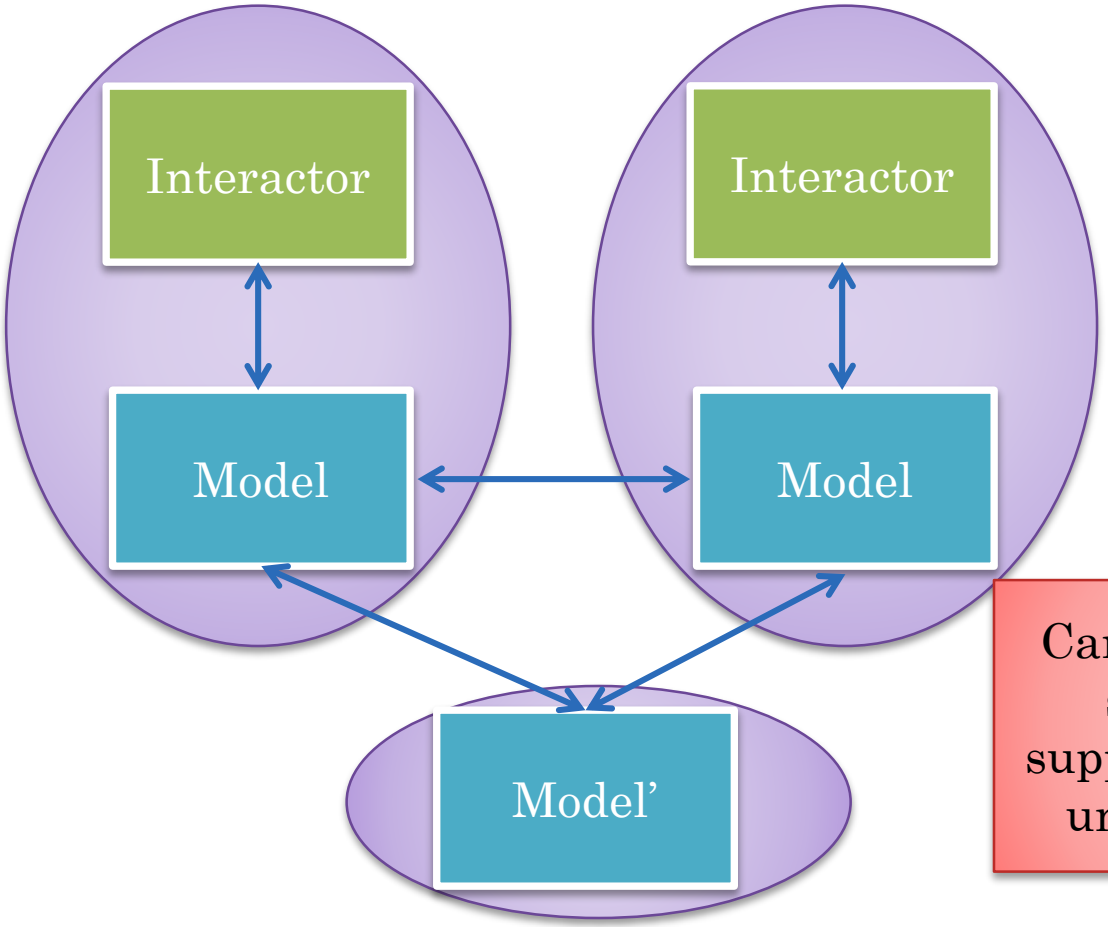
Different users will see different output

Behavior of centralized and replicated different

Assumption: Output should be only a function of input

Non determinism!

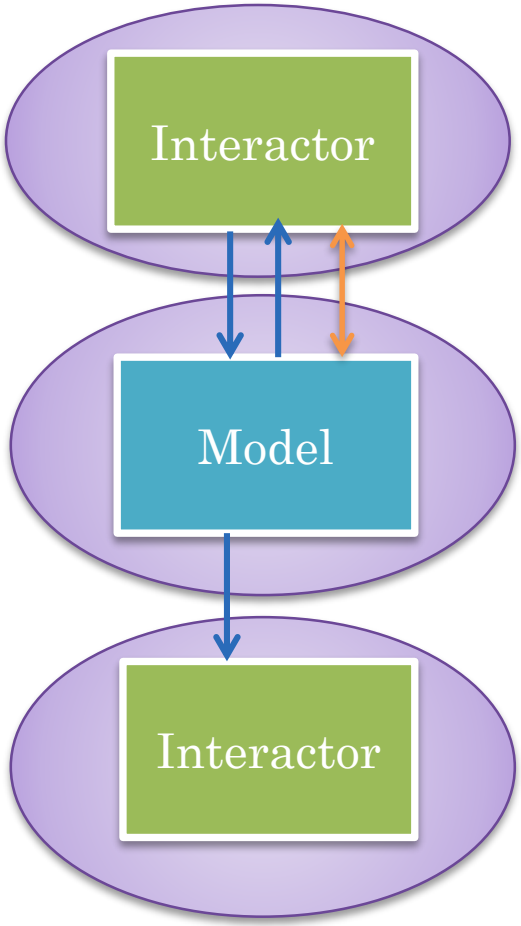
Distinguished "Replica" Model Solution



Cannot use application-specific solution if supporting collaboration-unaware applications

A central model executes operations that are expensive, non-idempotent, or access central resources

Centralized vs. Replicated Model



None of the replication issues

Feedback times involve round trip delays

Feed through incurs extra hop (beyond relaying)

Refresh and query operations also involve round trip delays (e.g. searching history)

Can we fix the last problem?

Cannot use caching of high-level state, no local non window state

Refresh, scrolling involves round trips

Shared Window Systems

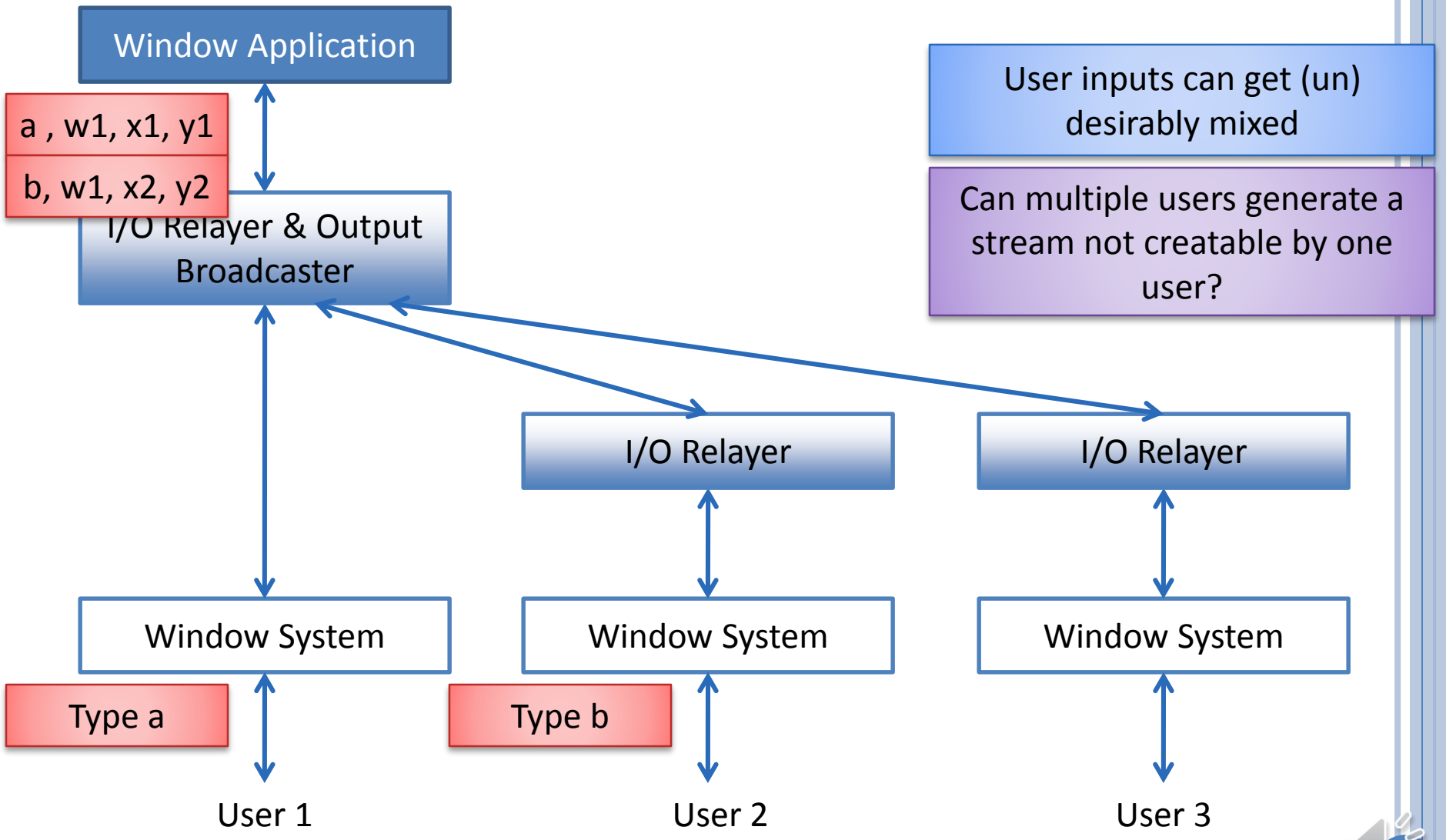
Problems of centralization and replication get aggravated

Collaboration-awareness required for distinguished process in replicated systems

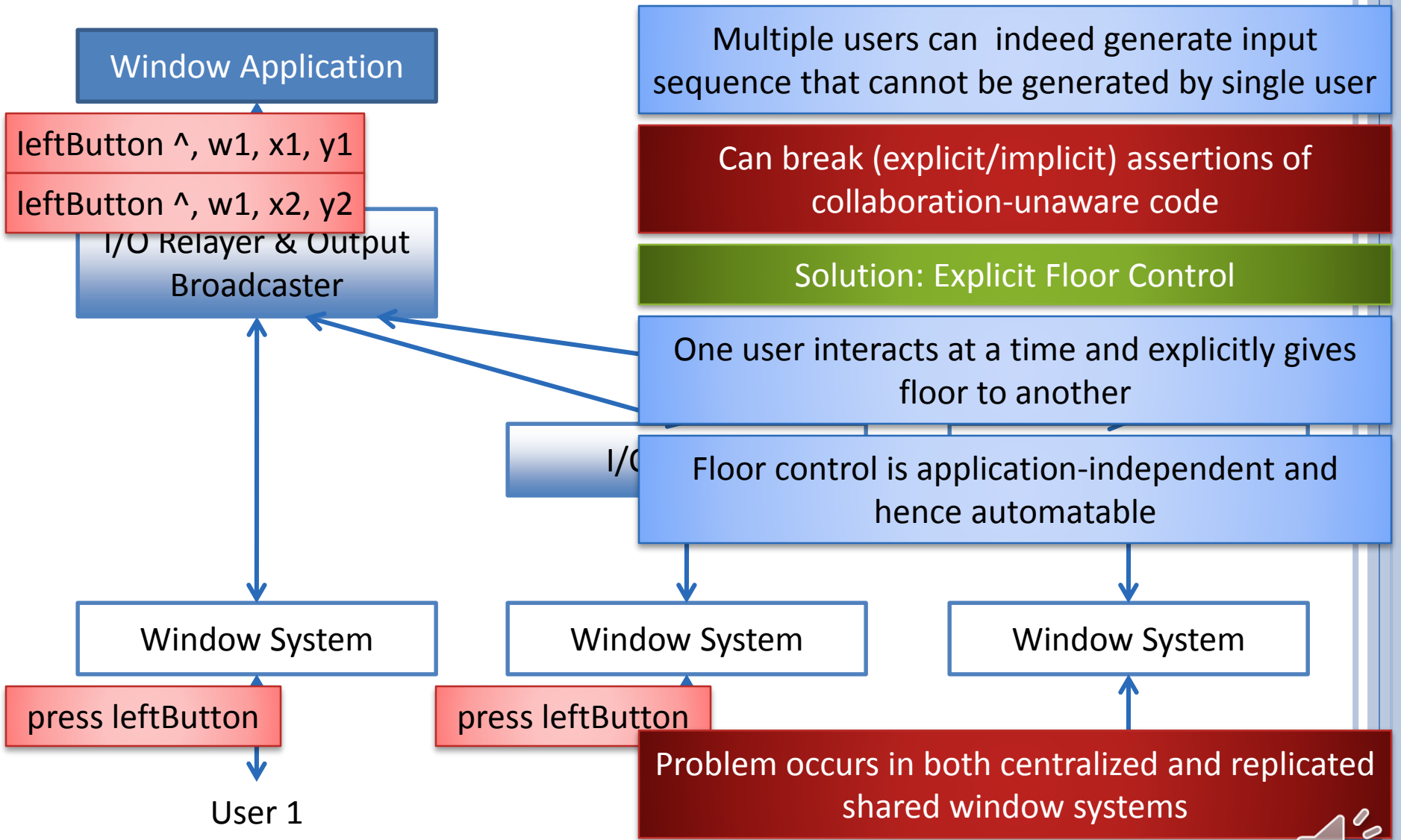
In central systems, round trip for readable model state

Plus other problems

Concurrent/Interleaved interaction



Explicit Floor Control

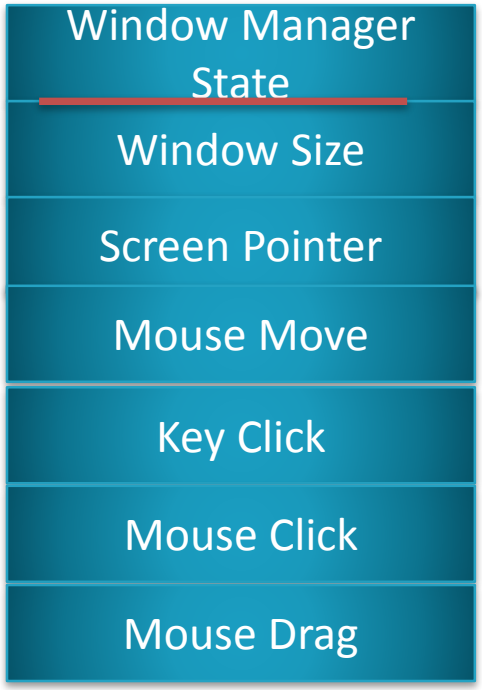


Shared Window Systems

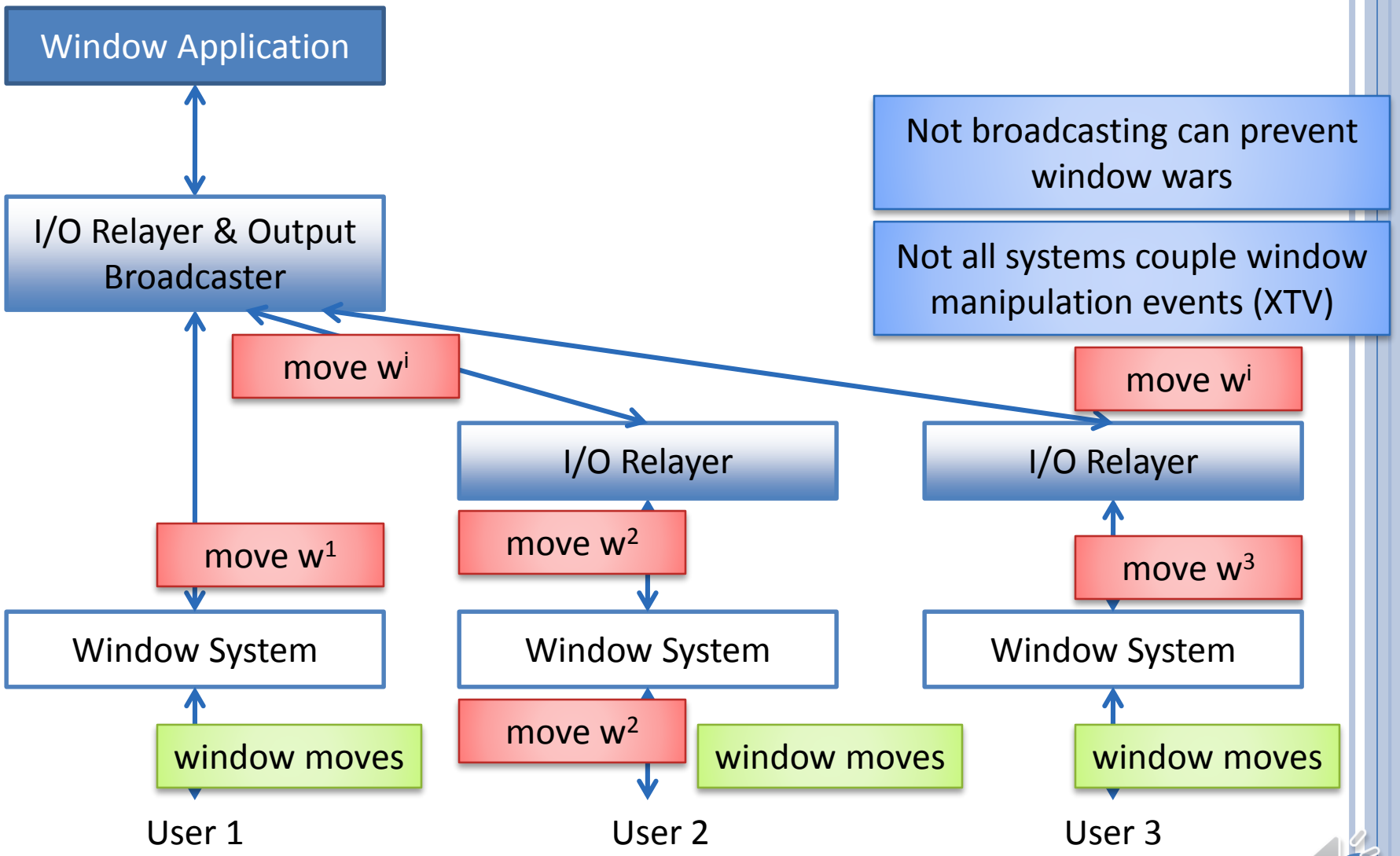
Problems of centralization and replication get aggravated
Collaboration-awareness required for distinguished process in replicated systems
In central systems, round trip for readable state

Plus other problems
Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events

Problem with Relaxed Wysiwis in Centralized Systems

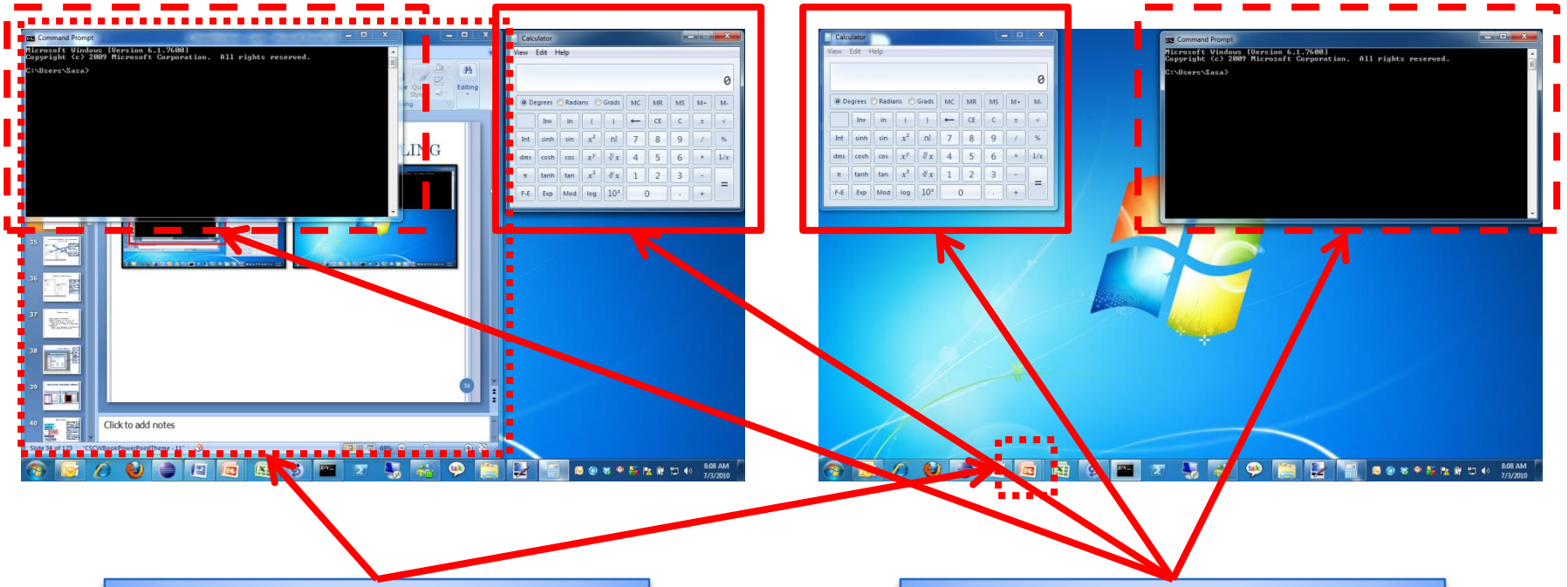


Sharing Window manager State



RELAXED WYSIWIS COUPLING

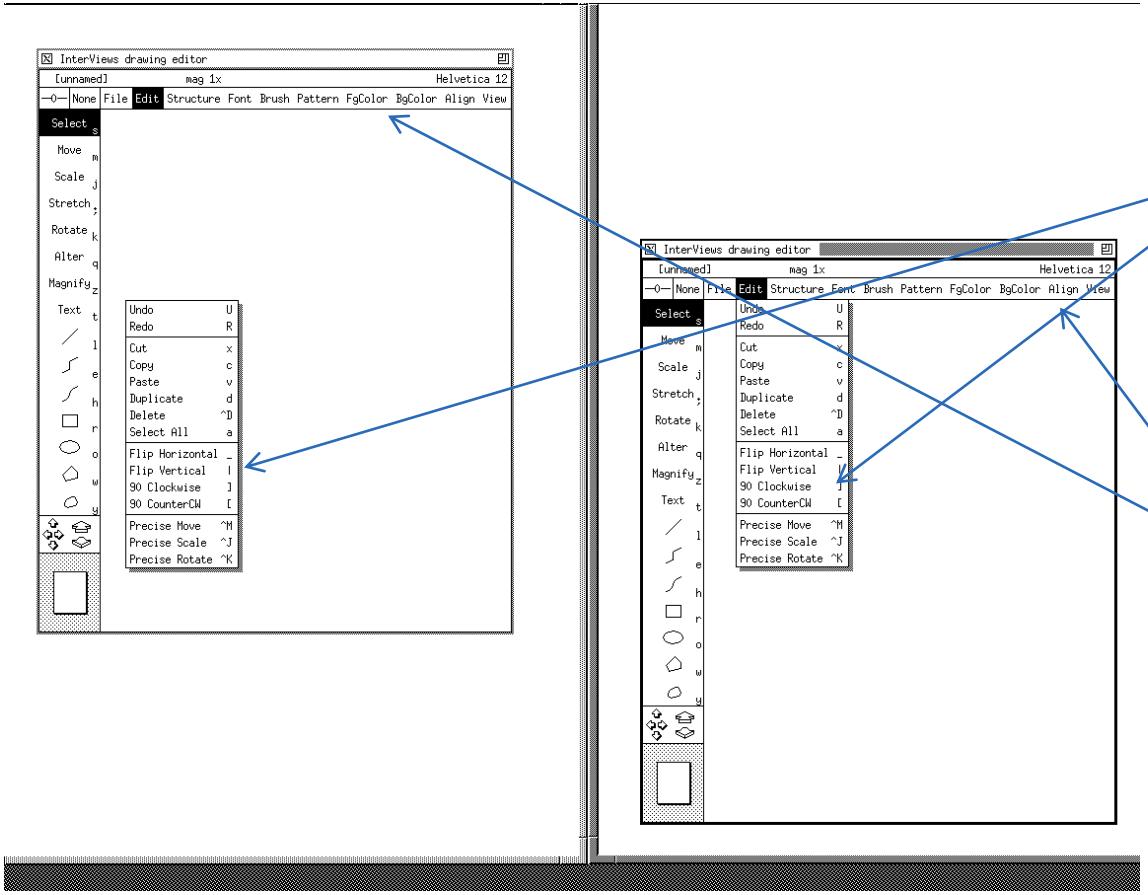
Implemented in XTV



Independent Window Minimization

Independent Window Positions

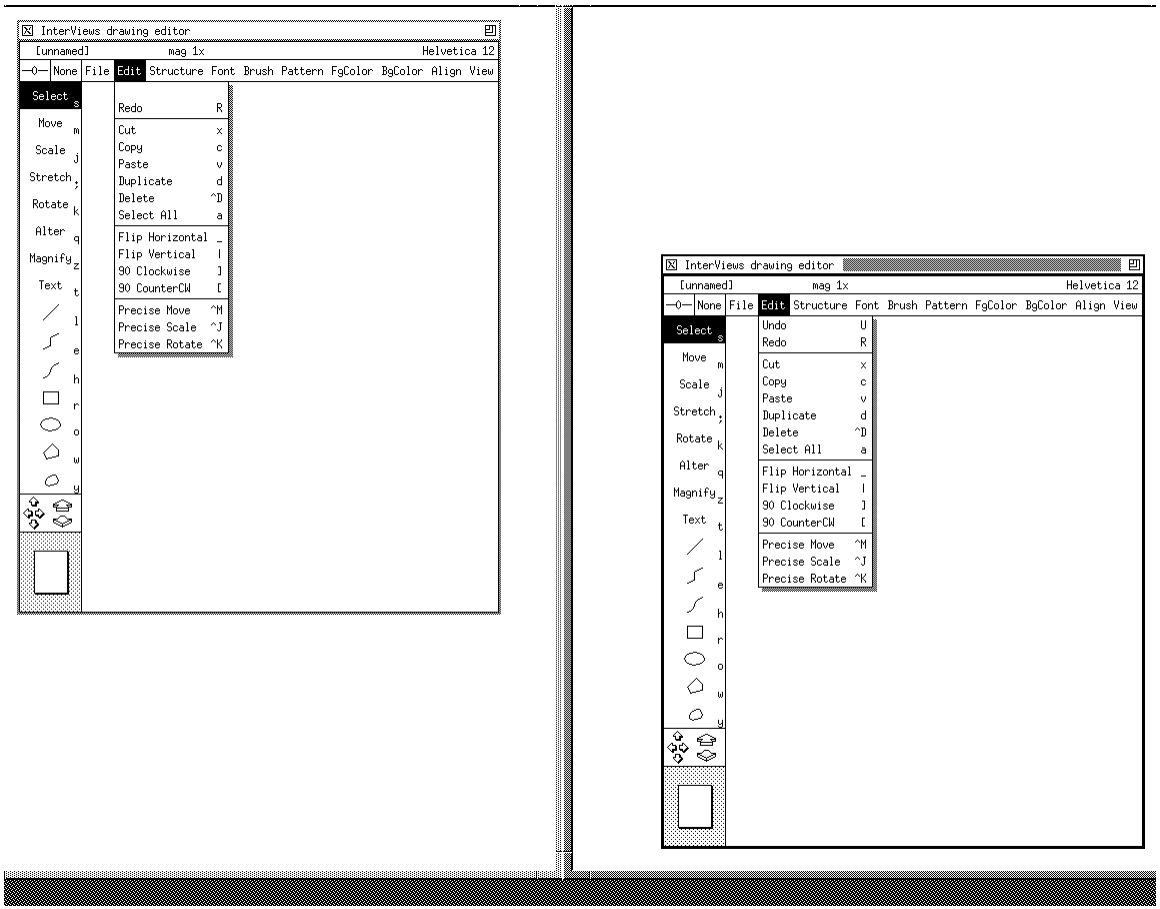
Pop-Up Menus Problem with Uncoupled Window State



Pop-up menus are top-level windows drawn at absolute positions by application.

Inner windows drawn relative to containing window.

Correct Pop-Up Menus



Proxy keeps track of root window and translates.

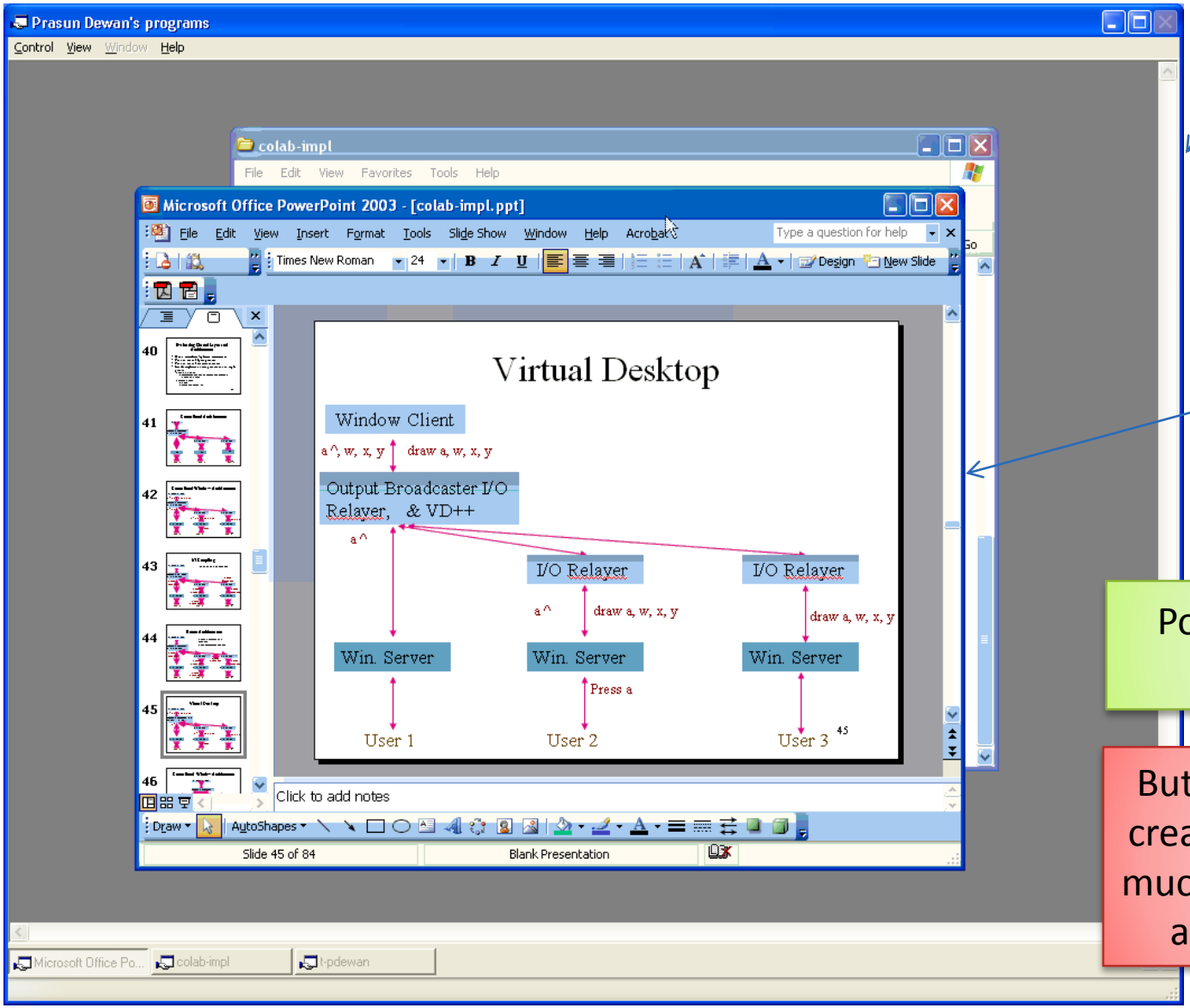
More than a proxy – understands underlying window system.

Not uncommon in proxies.

Semantic Issue

- Should window state be coupled?
- Coupled → window wars (Stefik et al '85)
- Uncoupled → no referential transparency
  - Cannot refer to the "upper left" shared window
  - Problems in centralized systems
- Compromise for centralized system
  - Create a virtual desktop on a slave computer for physical desktop of the master user

Virtual Desktop



Privately scrollable , movable window representing master screen

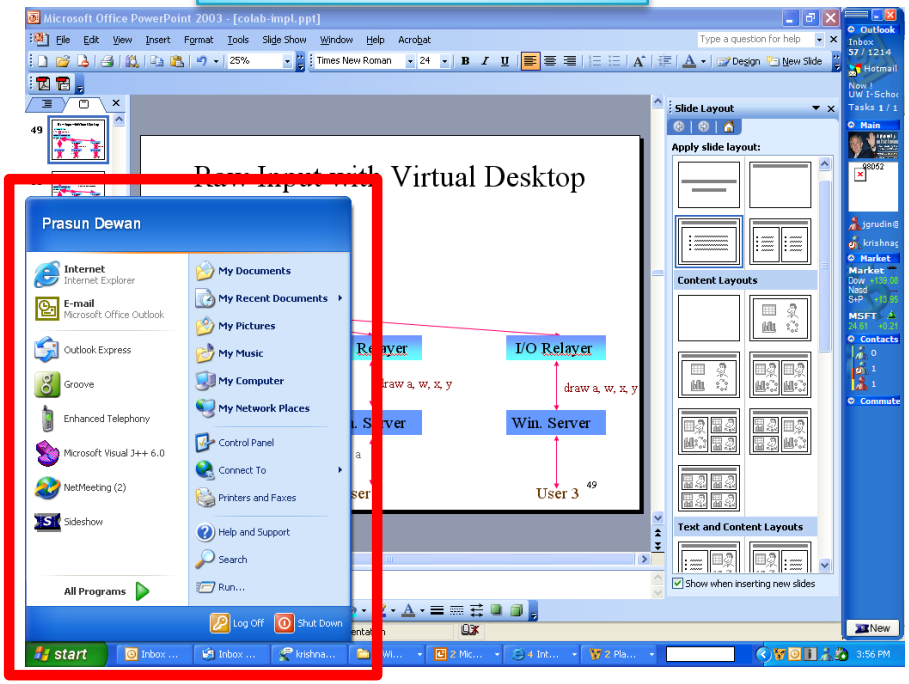
Nested, shared , WYSIWIS master window

Pop up menus not a problem

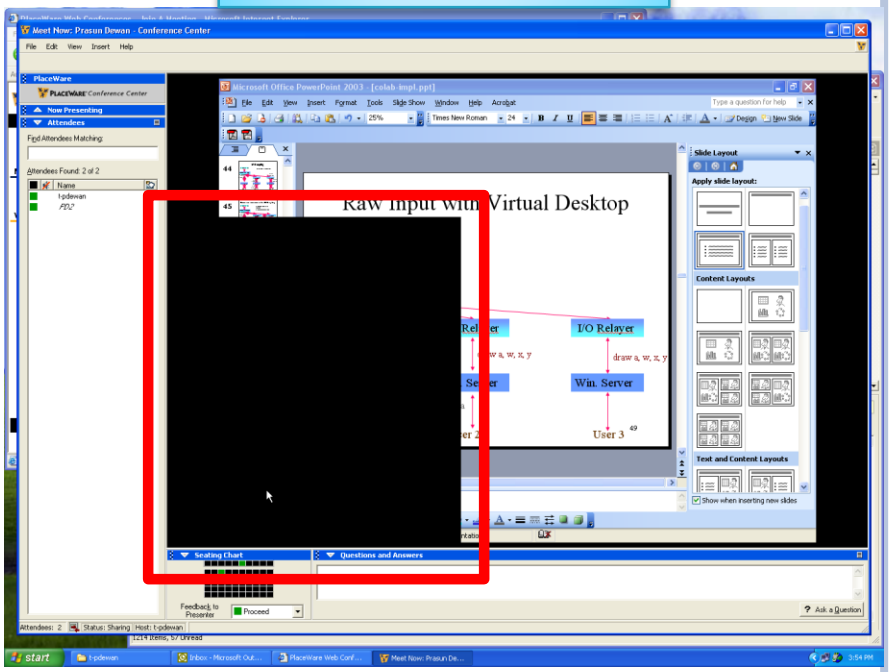
But slave system has to create a virtual desktop, much more than a proxy, a window manager

ANOTHER ISSUE: COUPLING OF UNEXPOSED REGIONS?

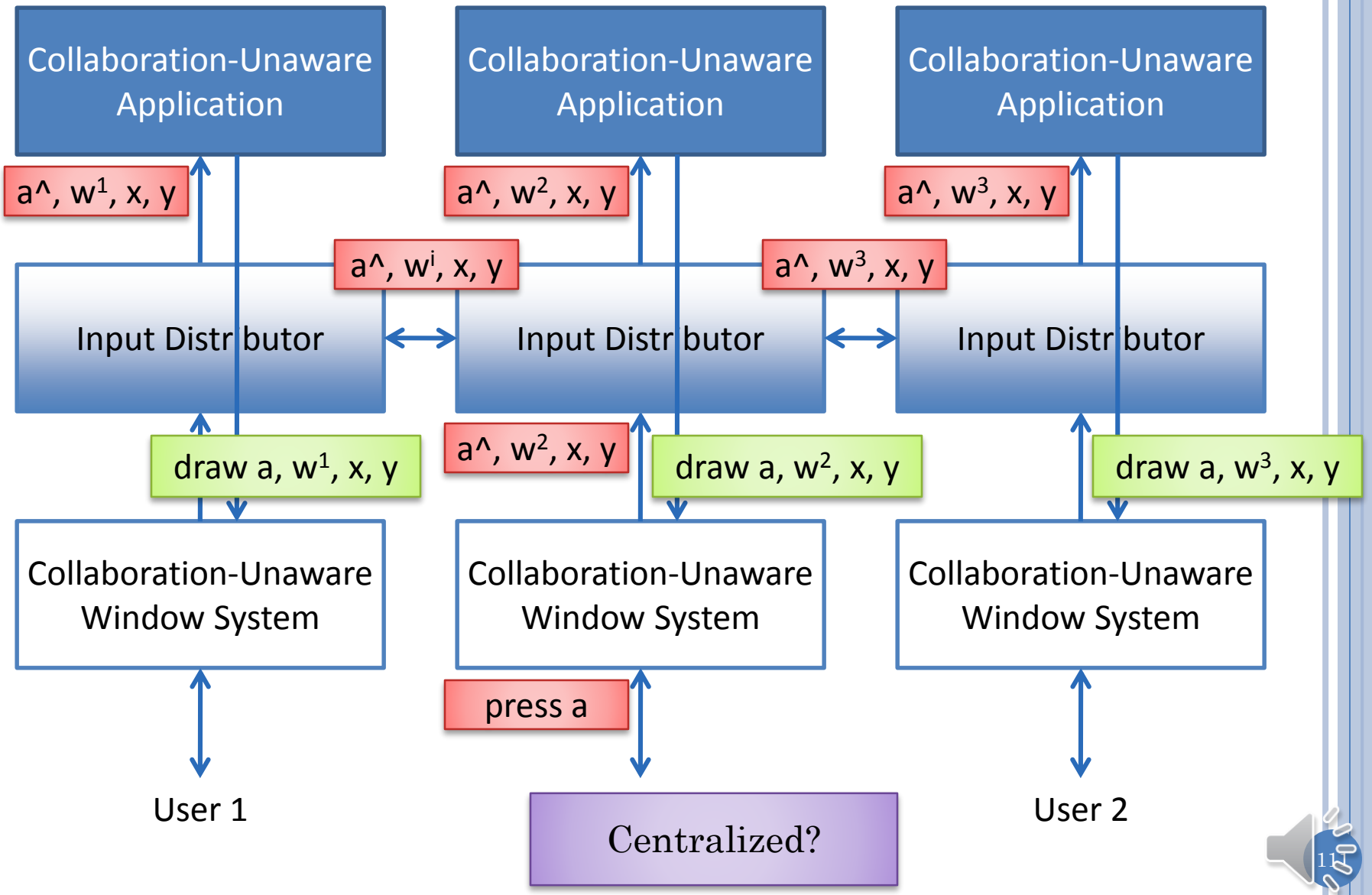
Real desktop



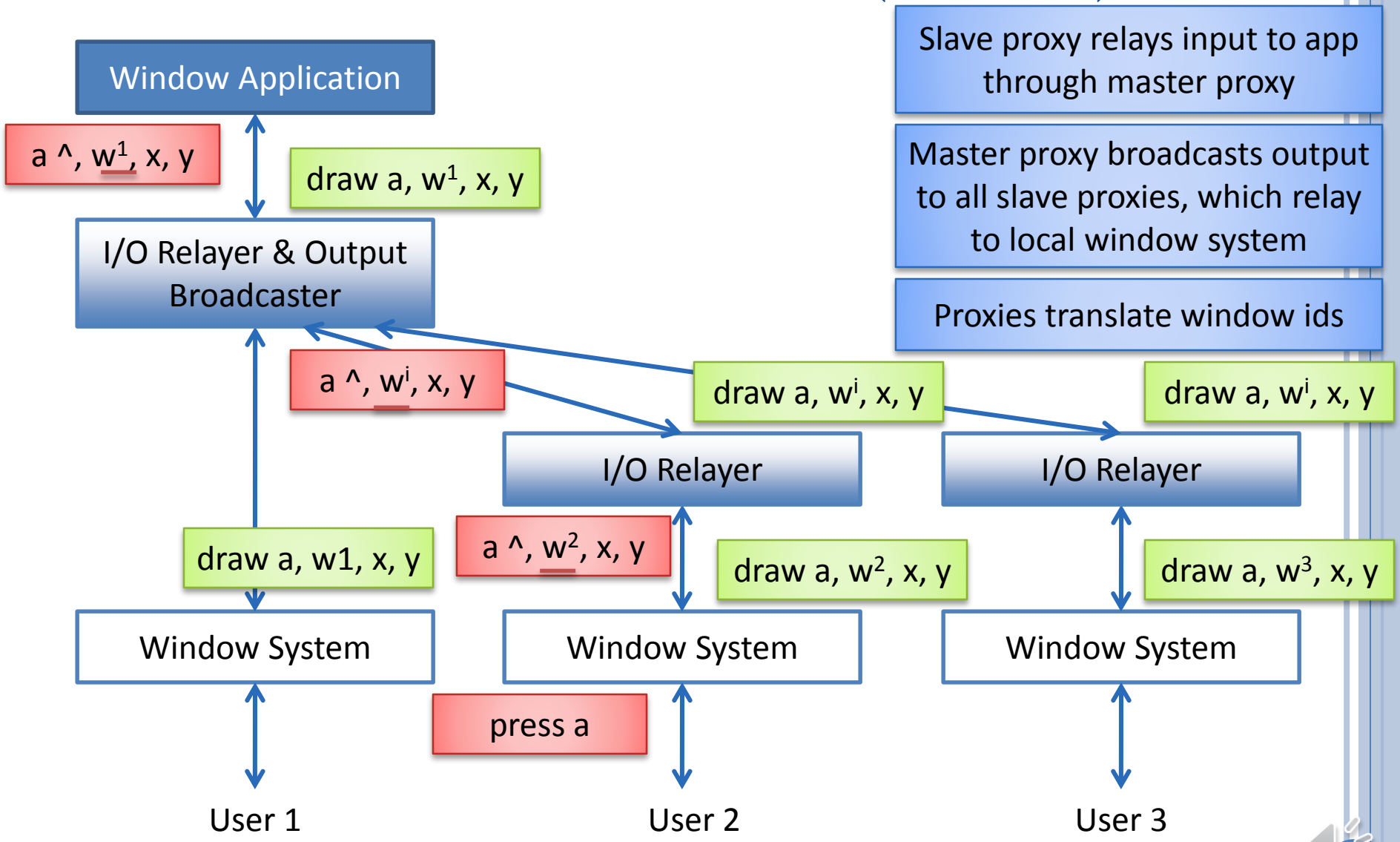
Virtual desktop



Replicated Systems (Review)



Centralized Systems (Review)



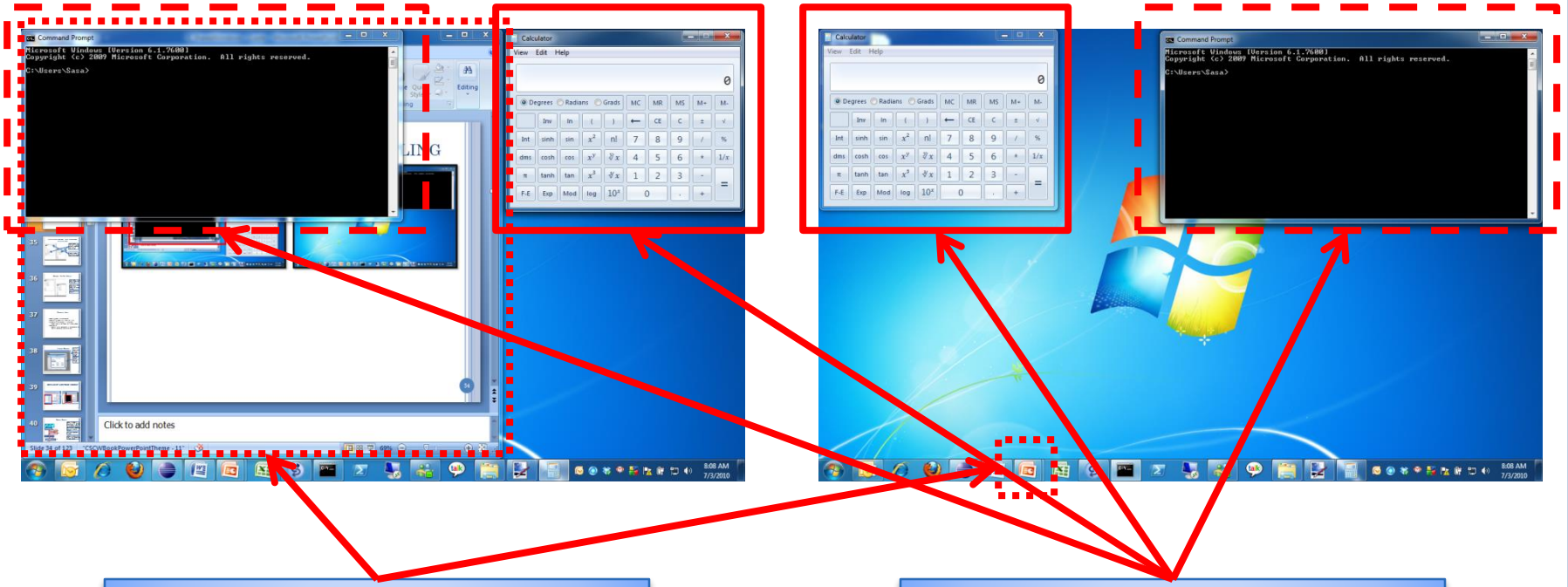
Shared Window Systems

Problems of centralization and replication get aggravated
Collaboration-awareness required for distinguished process in replicated systems
In central systems, round trip for readable state

Plus other problems
Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events
In centralized systems with relaxed WYSIWIS, pop up menus and obscured master windows can create problems

Relaxed WYSIWIS Coupling (First-Class Remote Windows))

Implemented in XTV

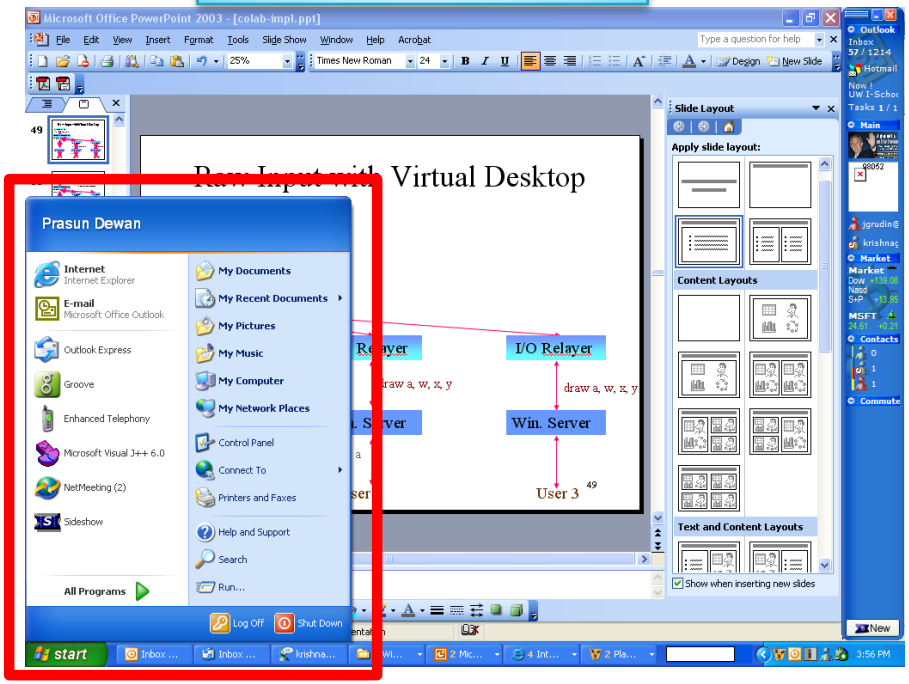


Independent Window Minimization

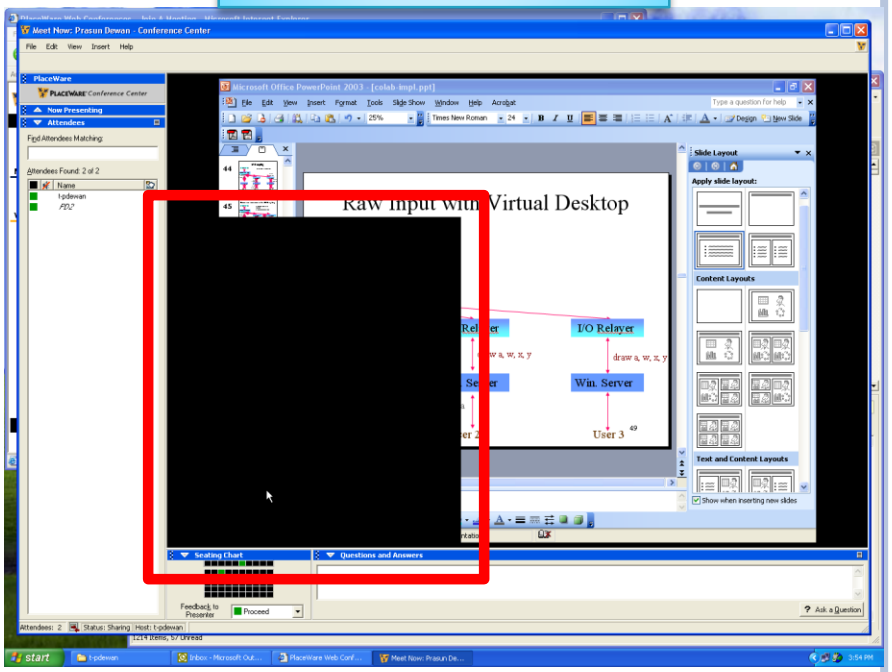
Independent Window Positions

Virtual Desktop

Real desktop

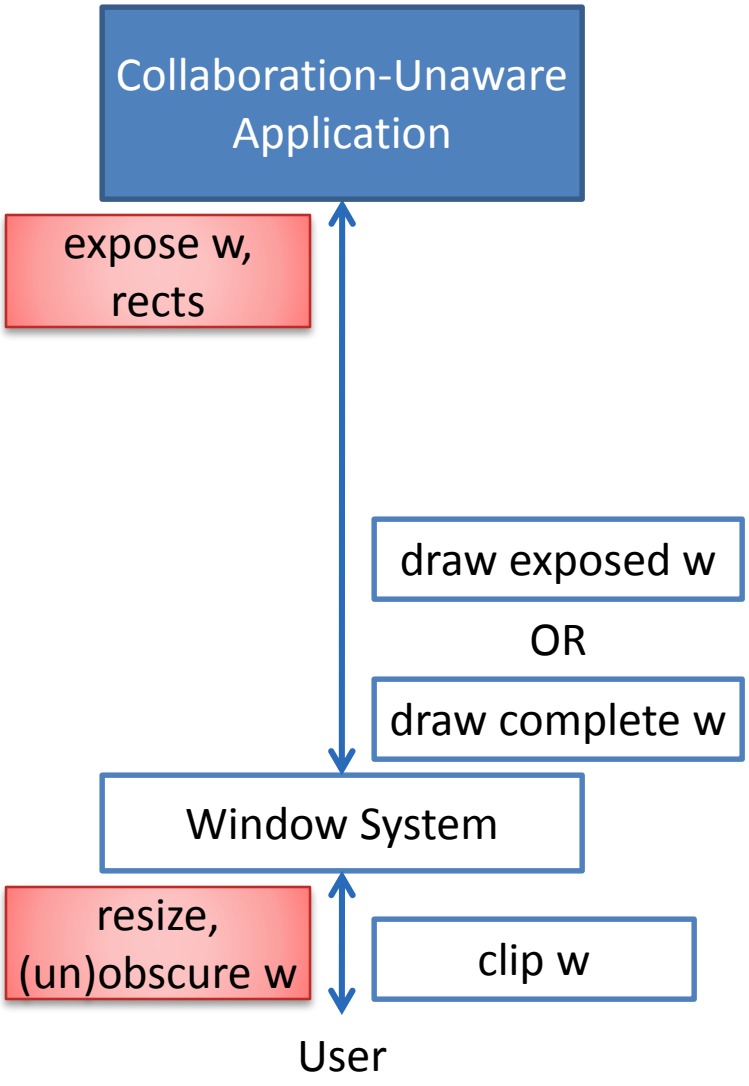


Virtual desktop



Corresponding windows in first-class remote windows and virtual desktops may be exposed differently

Expose Events



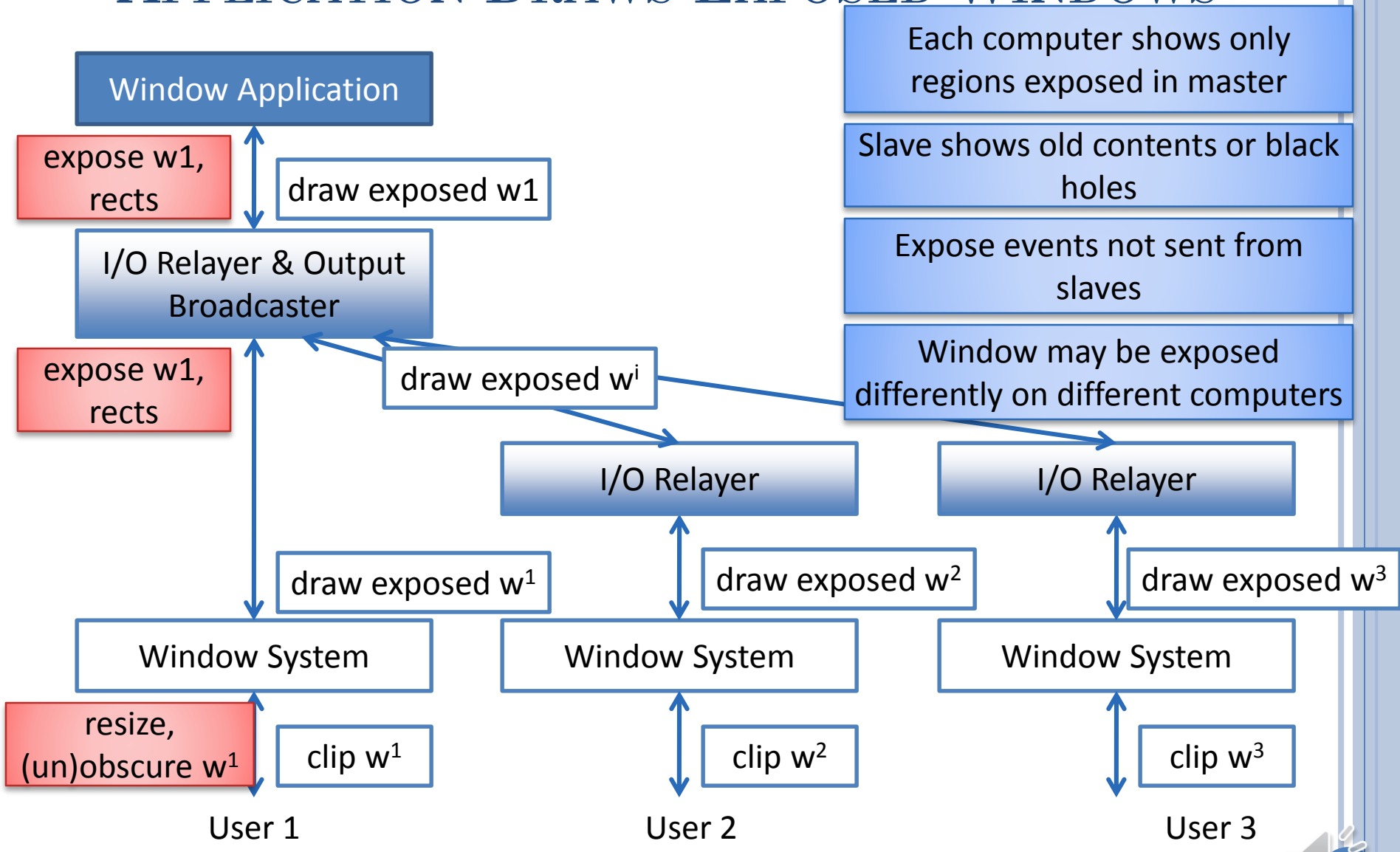
When a window is resized, or (un) obscured expose event is sent to it with exposed regions

Application is expected to draw only exposed regions

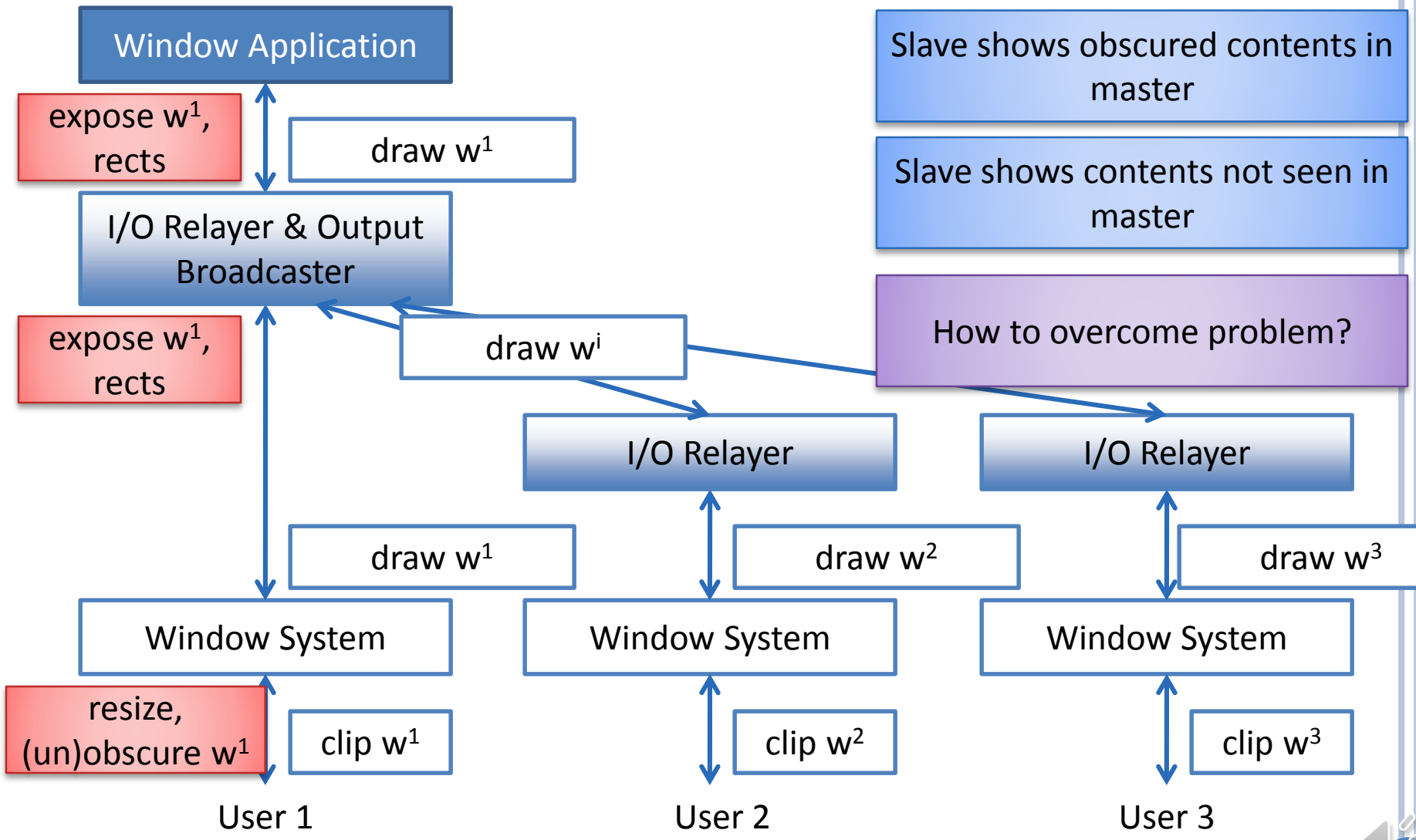
May draw complete window

Window system clips in case it app. drawn outside window bounds

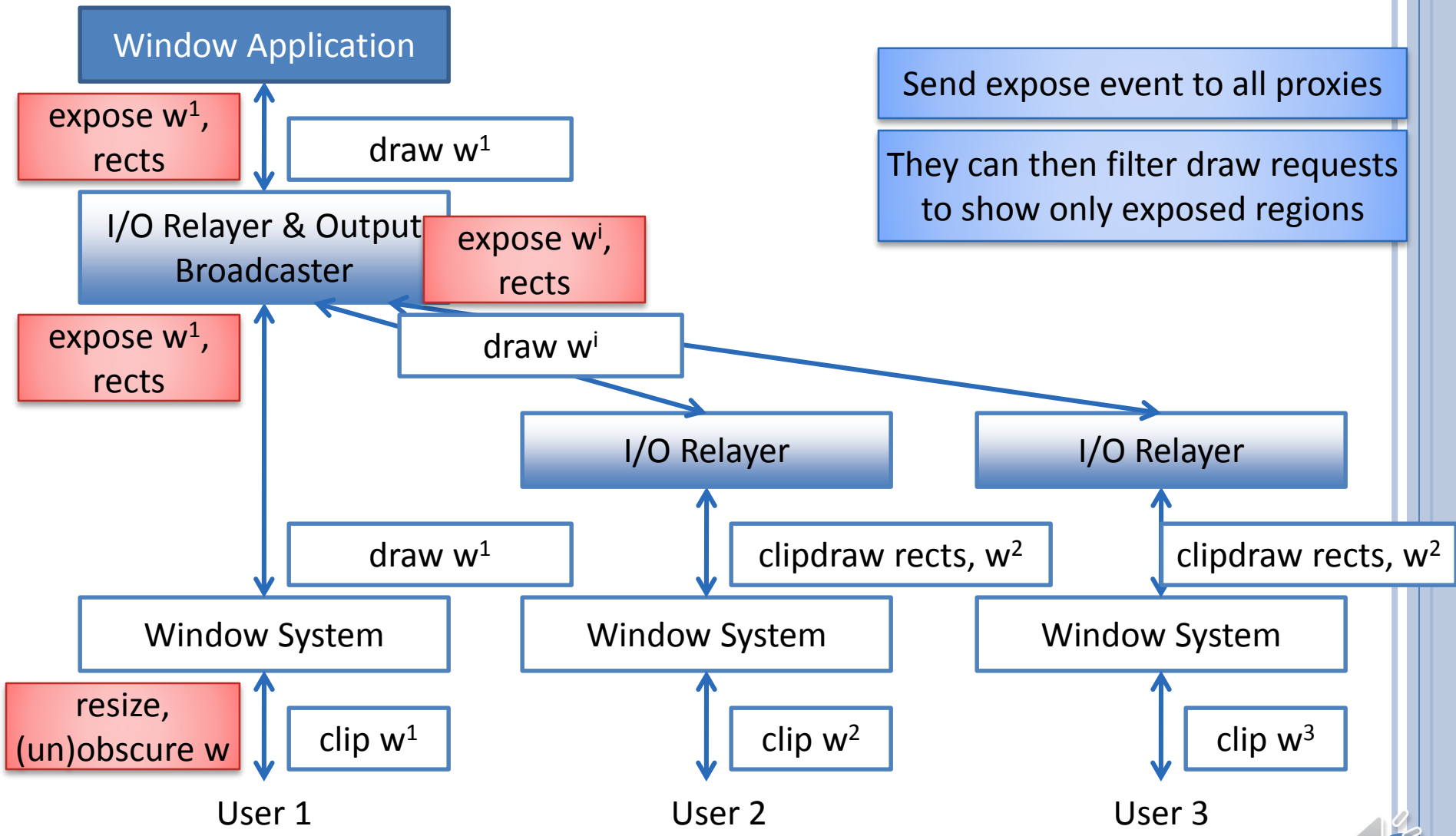
Application Draws Exposed Windows



Application Draws Complete Window



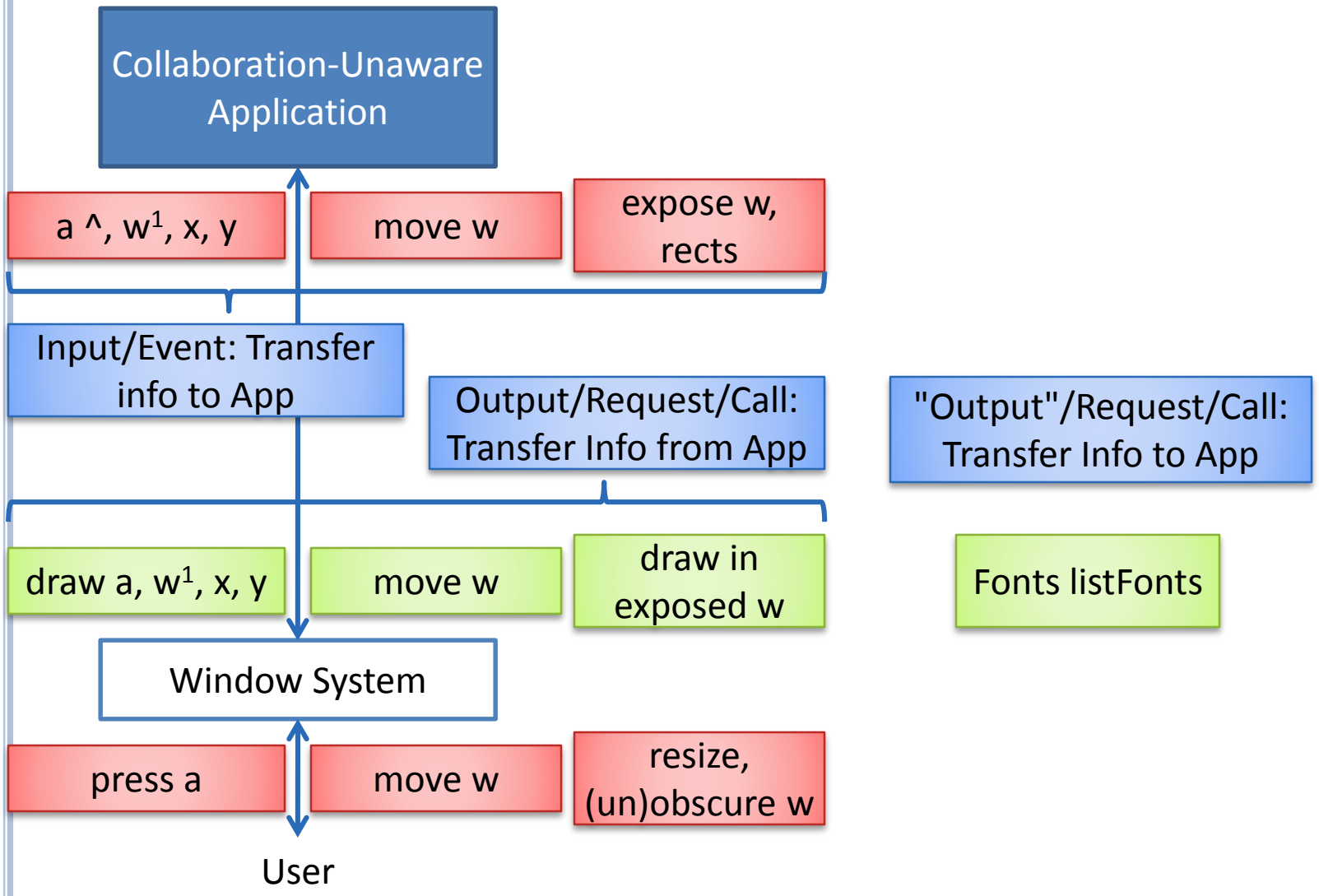
Preventing Privacy Issues: Expose Coupling



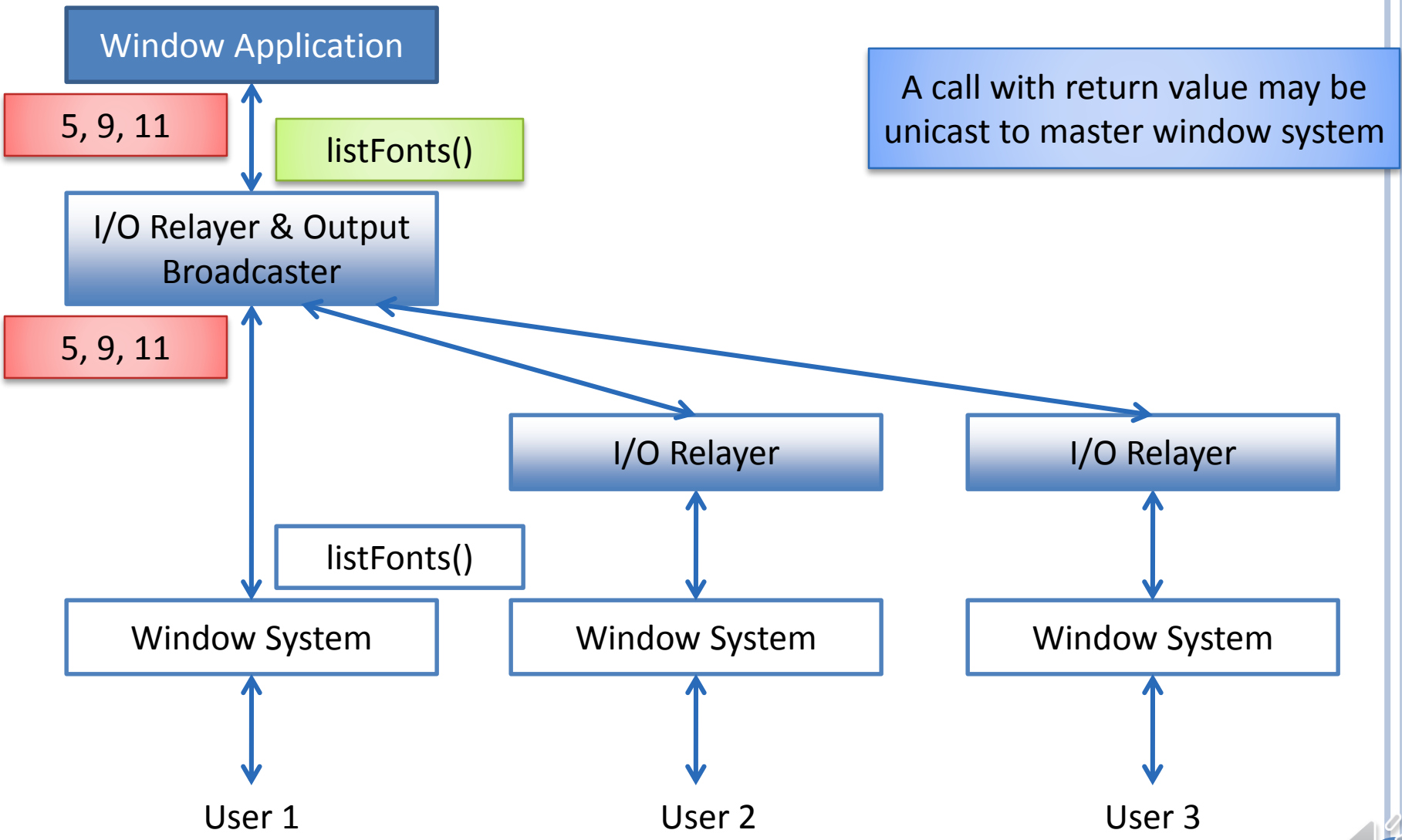
Window-based Coupling

- Mandatory
  - Window sizes
  - Window contents
- Optional
  - Window positions
  - Window stacking order
  - Window exposed regions
- Optional can be done with or without virtual desktop
  - Without virtual desktop, problems of pop up menus
- In both cases, expose events are an issue that proxies must address

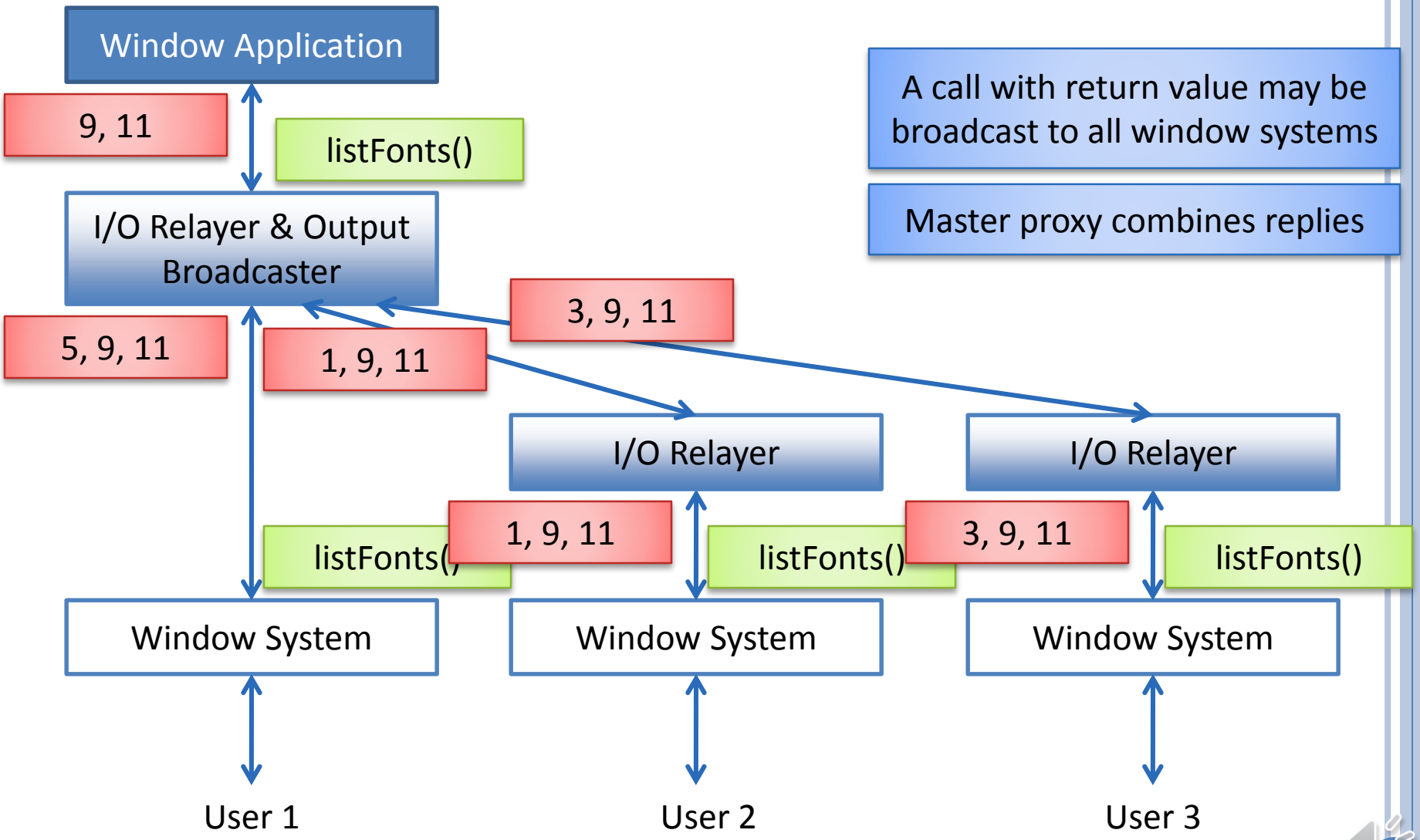
"Output" Calls with Return Values



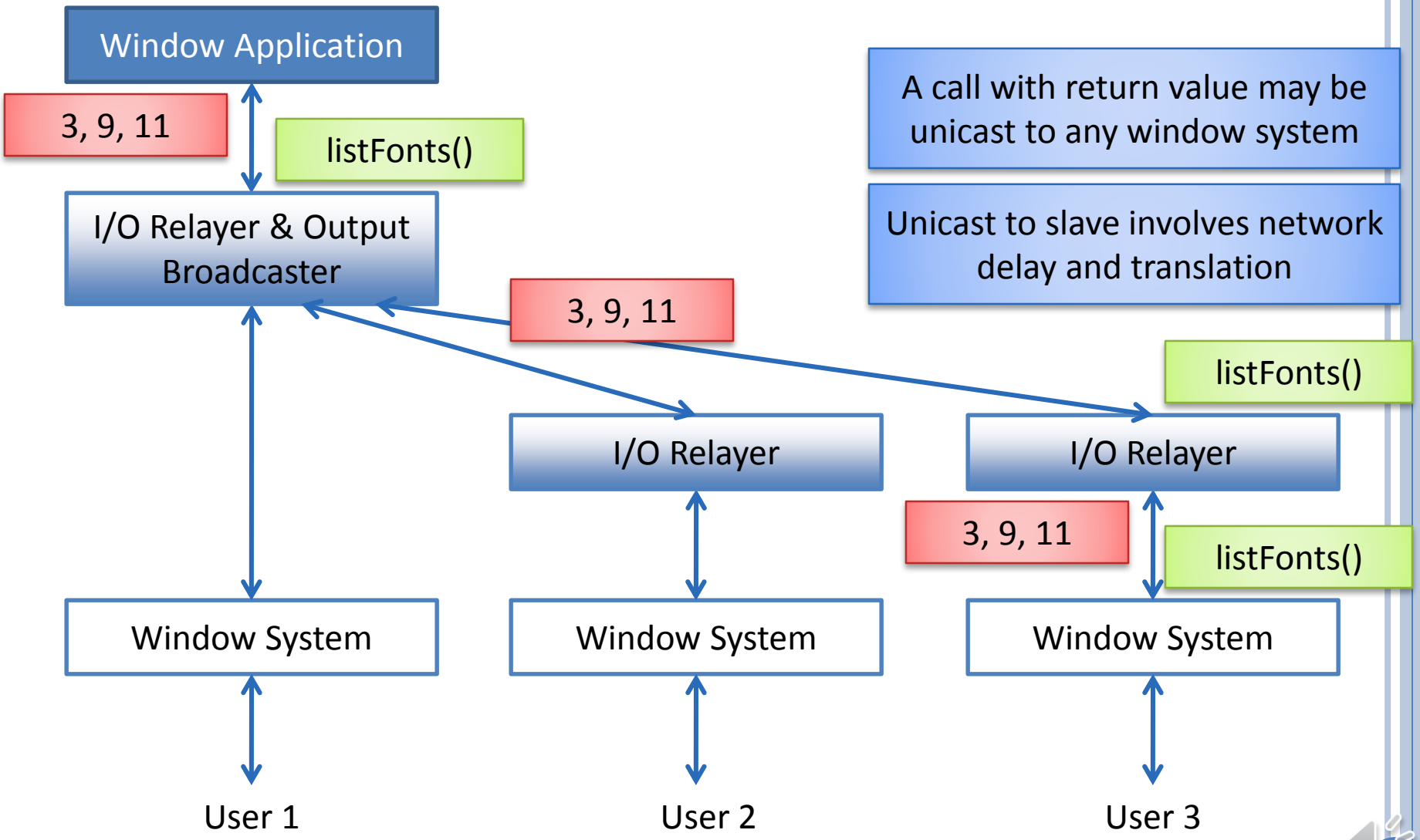
Unicasting to Master



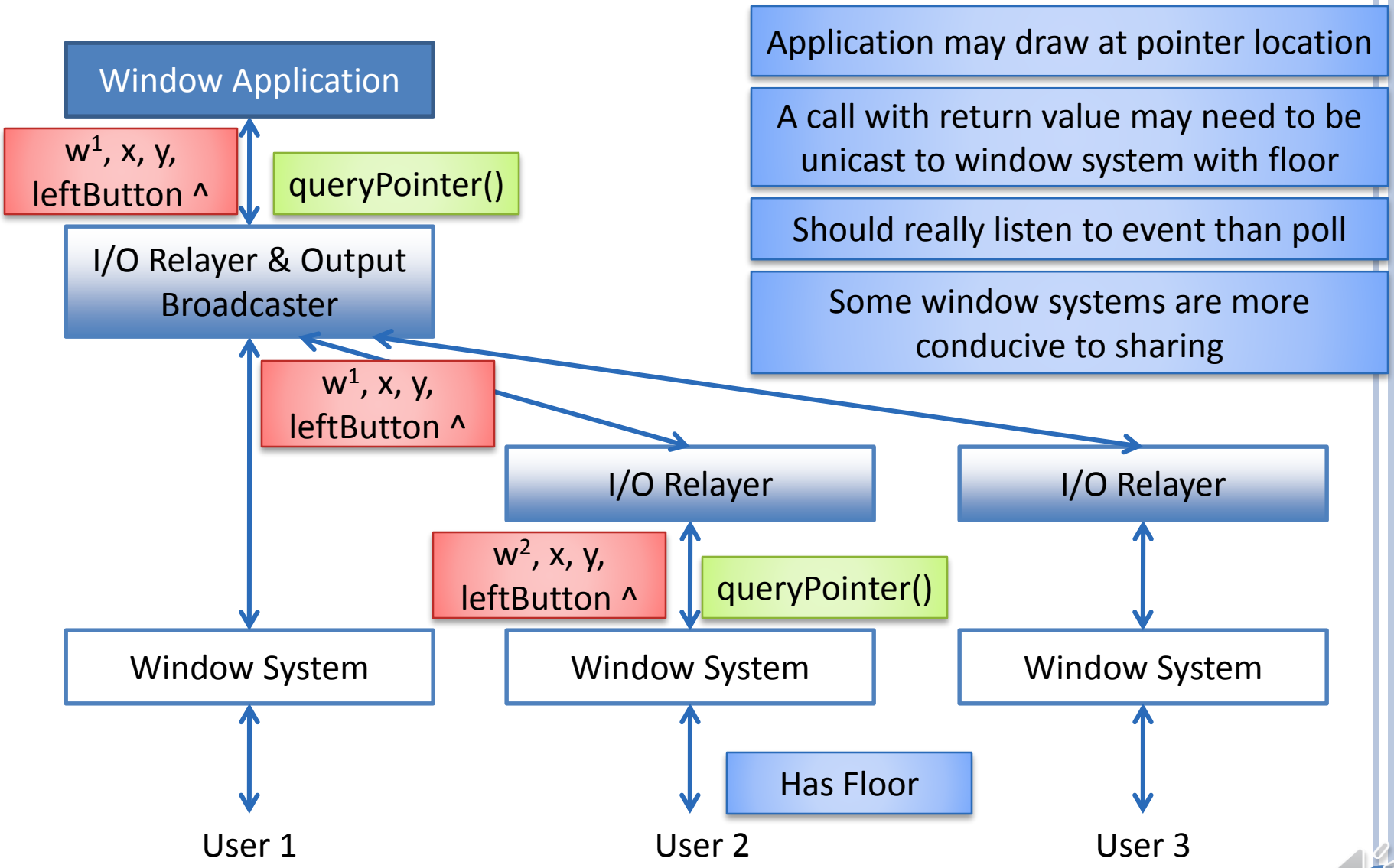
Broadcasting Calls with Return Values



Unicast to Some Slave



Unicast to Active Slave



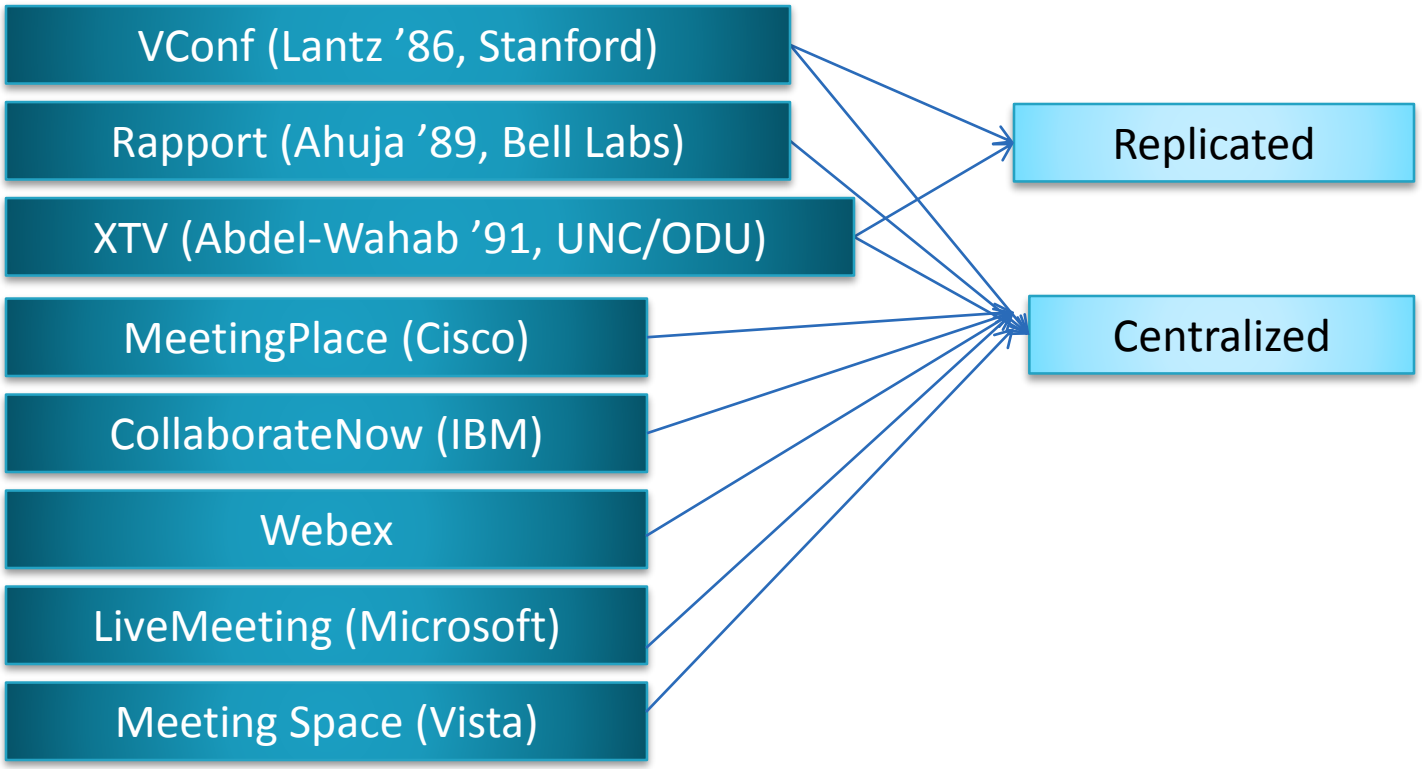
Shared Window Systems

Problems of centralization and replication get aggravated
Collaboration-awareness required for distinguished process in replicated systems
In central systems, round trip for readable state

Plus other problems
Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events
In centralized systems with relaxed WYSIWIS, pop up menus and obscured master windows can create problems
In centralized systems with query calls, the target of these calls is an issue

Replicated vs. Centralized Infrastructures

Research systems tried both architectures.

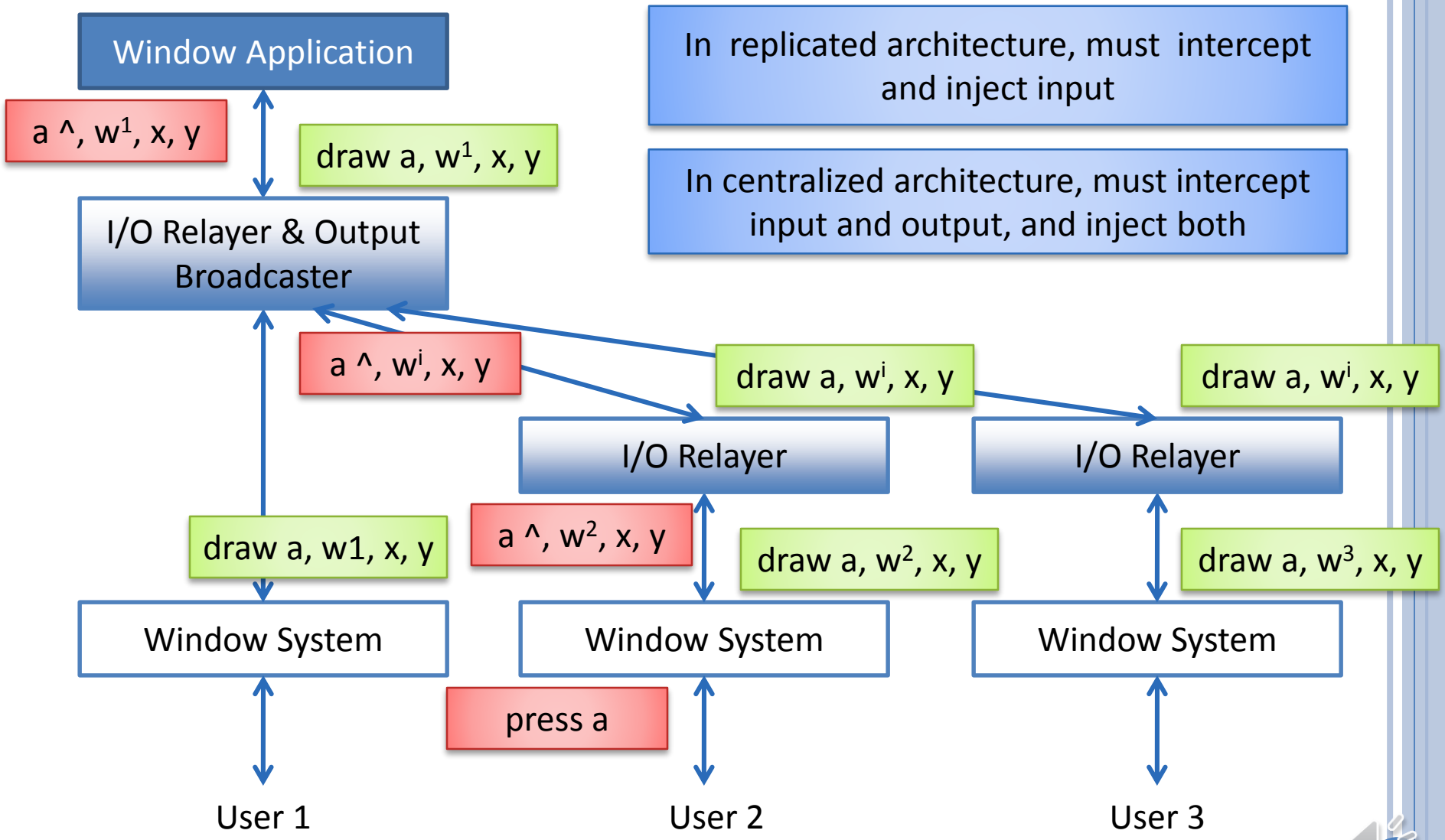


Commercial ones implement centralized

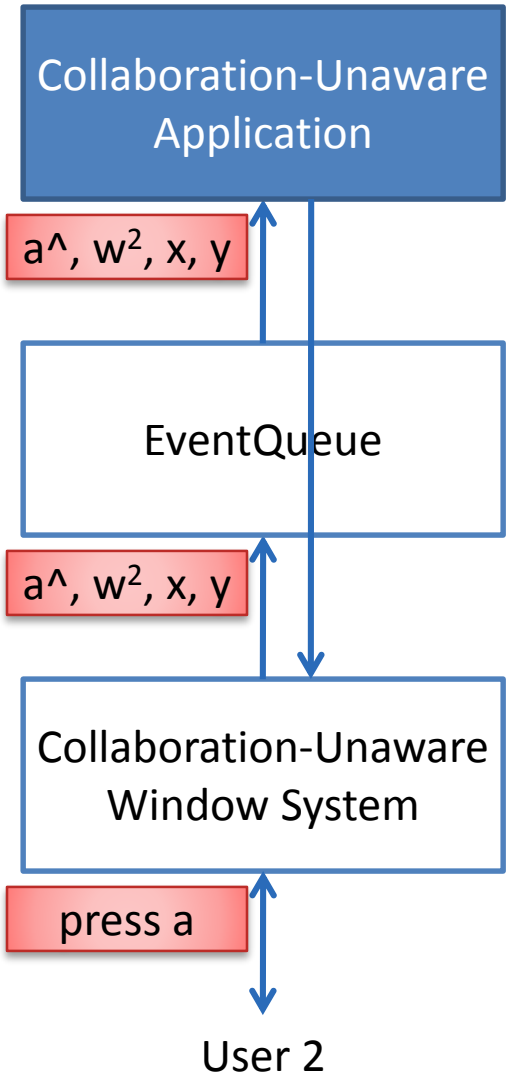
Case Study: Java AWT

Why is centralized harder based on what you know about Java?

Interception Differences



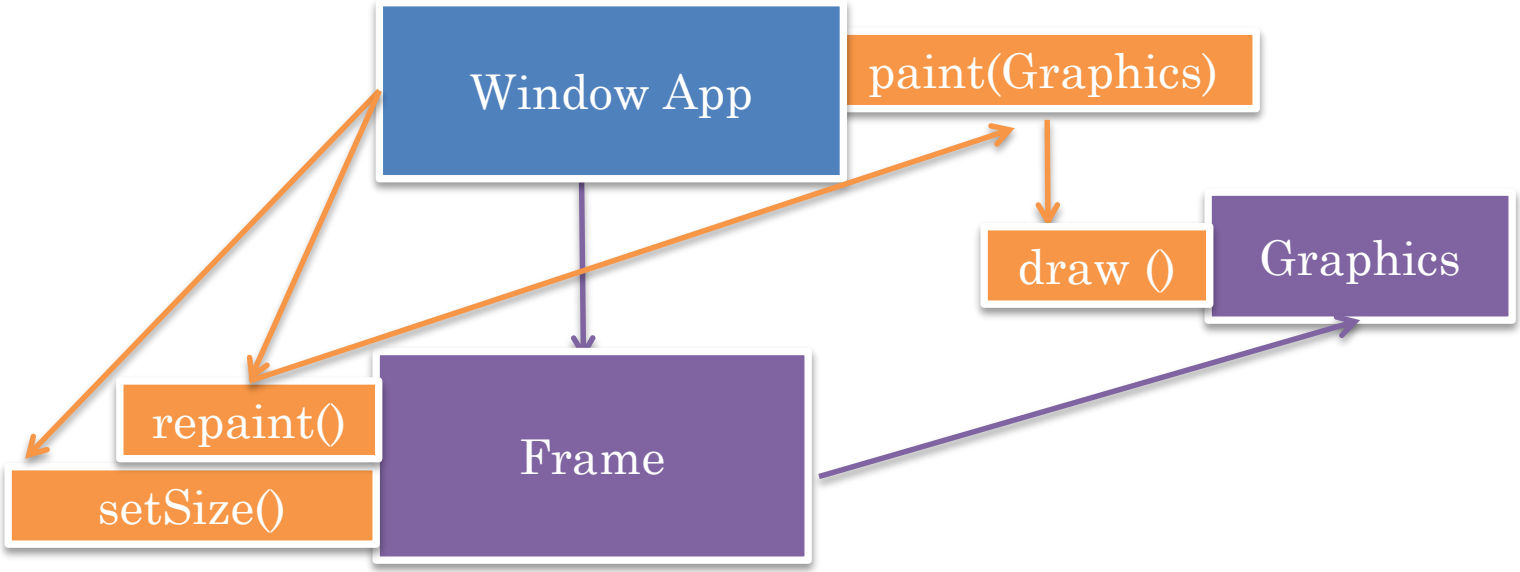
Input vs. Output Differences



All input defined by a single event type

Funneled through a single replaceable object

How to Intercept Output Calls?



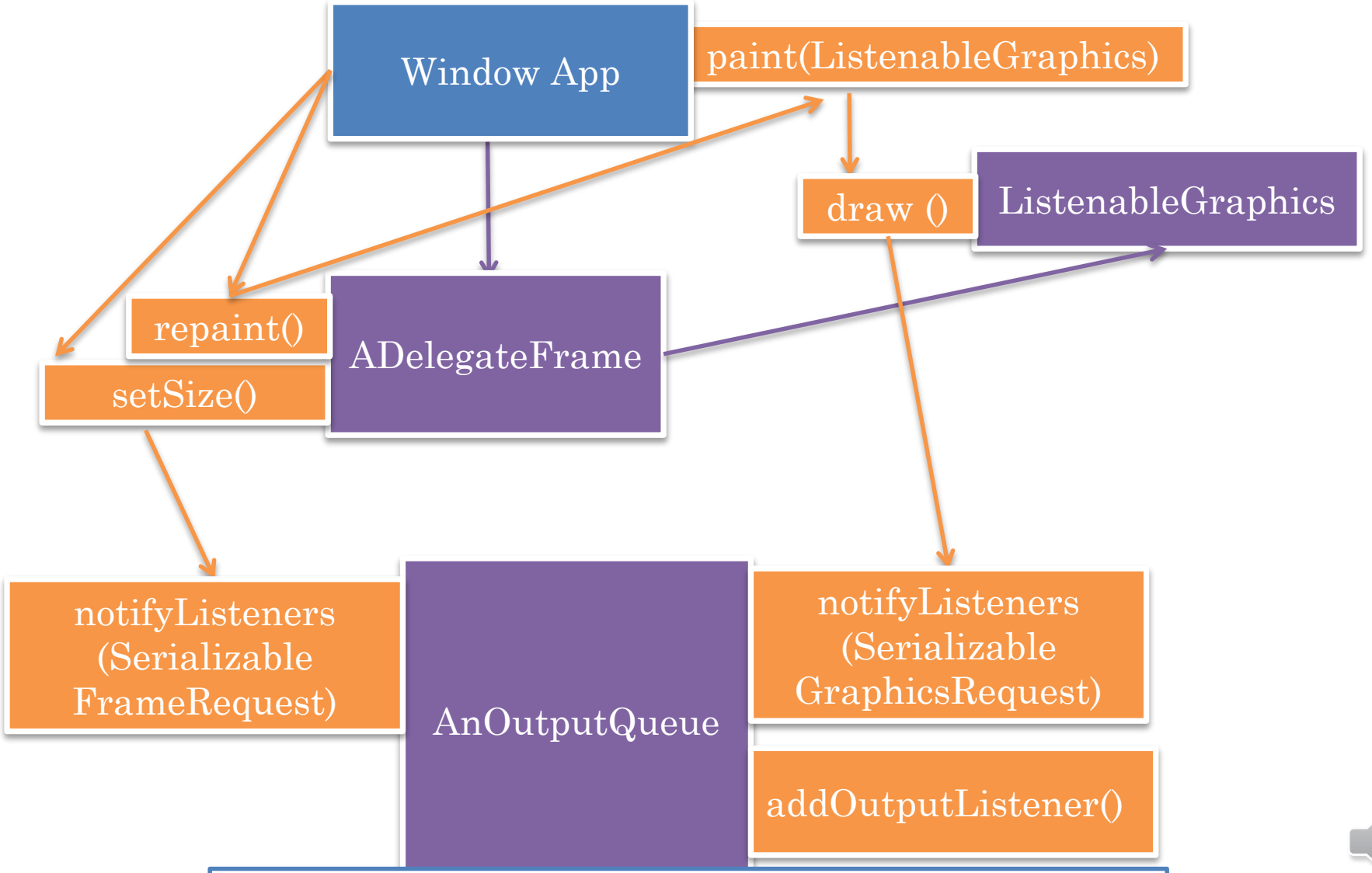
Calls to both frame and graphics must be intercepted

Java not designed to allow output interception

Calls are made by making different invocations, not passing a unifying data structure

How to change AWT/Swing?

Library Global Queue




```
AnOutputQueue.addOutputListener(this);
```

Global Queue

All frame calls and listenable graphics calls sent to output queue

AnOutputQueue, like AnExtendibleAWTQueue, allows listeners

Master Painter

```
public class AListenableCharacterDrawer
        extends ACursorTrackerOfDelegateFrame
        implements ListenablePainter {
    final static int CARAT_LENGTH = 10;
    public AListenableCharacterDrawer(ADelegateFrame theDelegateFrame) {
        super(theDelegateFrame);
        delegateFrame.addPainter(this);
    }
    public void paint(
            ADelegateFrame theDelegateFrame, ListenableGraphics g) {
        g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
        g.drawString("" + lastChar, charX, charY);
    }
    public void mousePressed(MouseEvent event) {
        super.mousePressed(event);
        delegateFrame.repaint();
    }
    public void keyTyped(KeyEvent event) {

    }
}
```

For each frame, need to track graphics requests issued in last paint call so we can make transactions at the other end

For each frame, need to track graphics requests issued in last paint call

SerializableRequest

```
public interface SerializableRequest extends Serializable {
  public int getFrameId();
  public String getName();
  public Object[] getArgs();
}
```

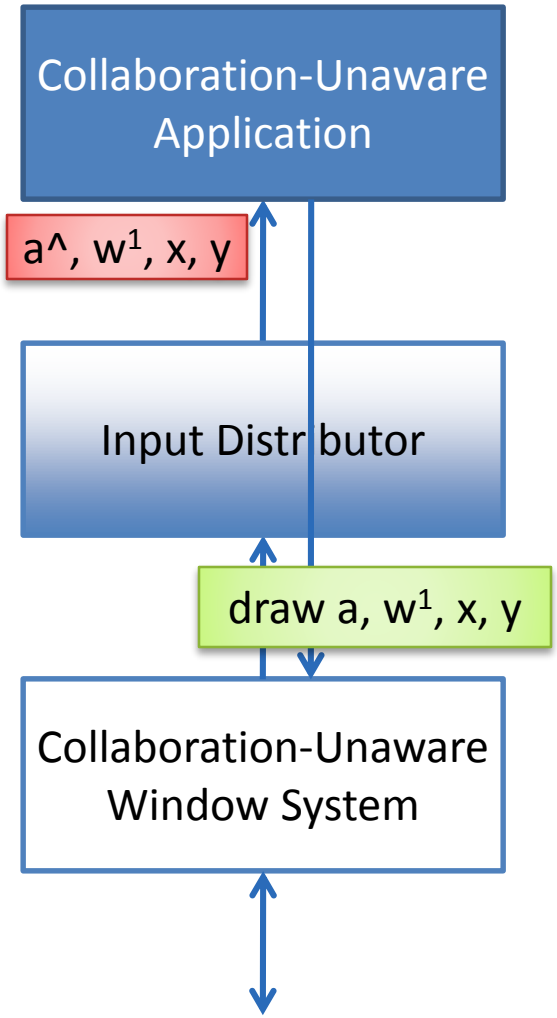
SERIALIZABLEGRAPHICSREQUEST

```
public interface SerializableGraphicsRequest extends SerializableRequest{
    public static final String DRAW_OVAL = "drawOval";
    public static final String DRAW_RECT = "drawRect";
    public static final String DRAW_STRING = "drawString";
    public static final String DRAW_LINE = "drawLine";
    public static final String PAINT_START = "paintStart";
    public static final String PAINT_END = "paintEnd";
    public Rectangle getClipBounds();
    public void setClipBounds(Rectangle theRectangle);
}
```

SerializableFrameRequest

```
public interface SerializableFrameRequest extends SerializableRequest{
  public static final String CREATE_FRAME = "createFrame";
  public static final String SET_SIZE = "setSize";
}
```

Intercepting Output in Real world



Call in the window system/widget usually does not result in notifications

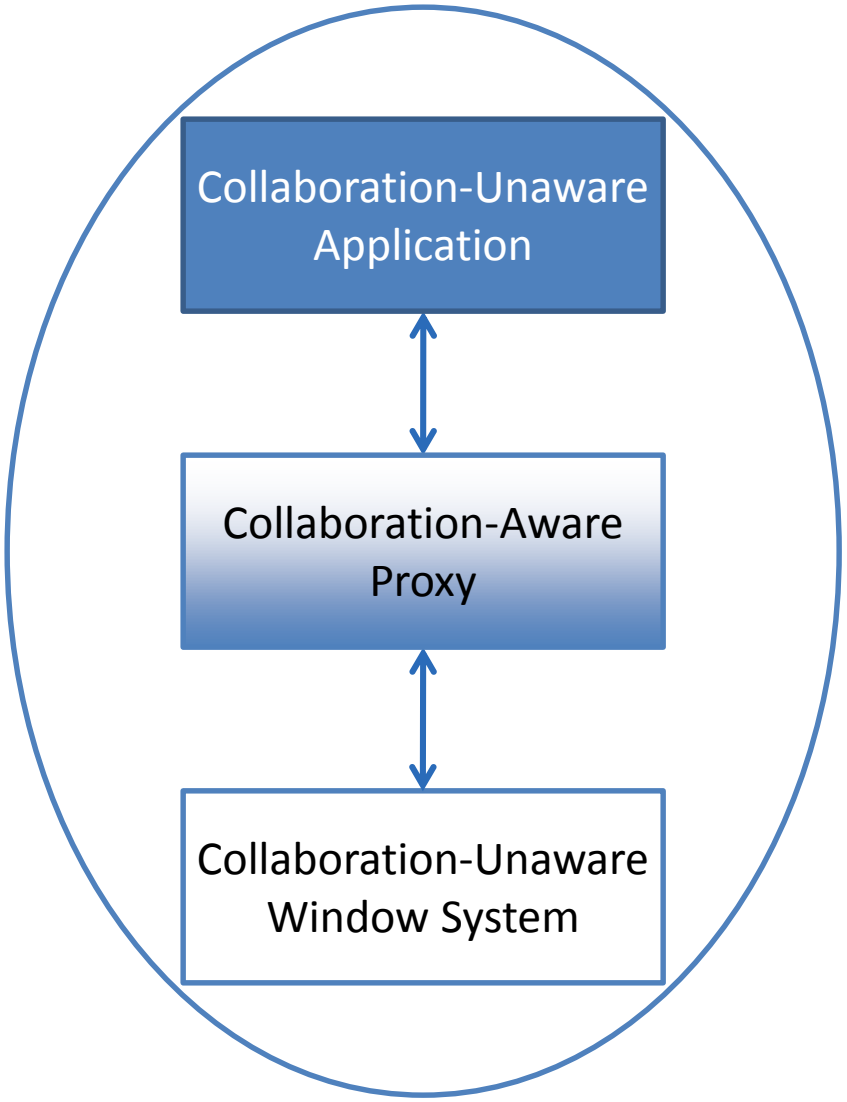
Local infrastructure module can send screen diffs to other computers

Must poll and send periodically

draw $pixrect_1$, $pixrect_n$

Intercepting output needed only in centralized architectures

Window System as A Library



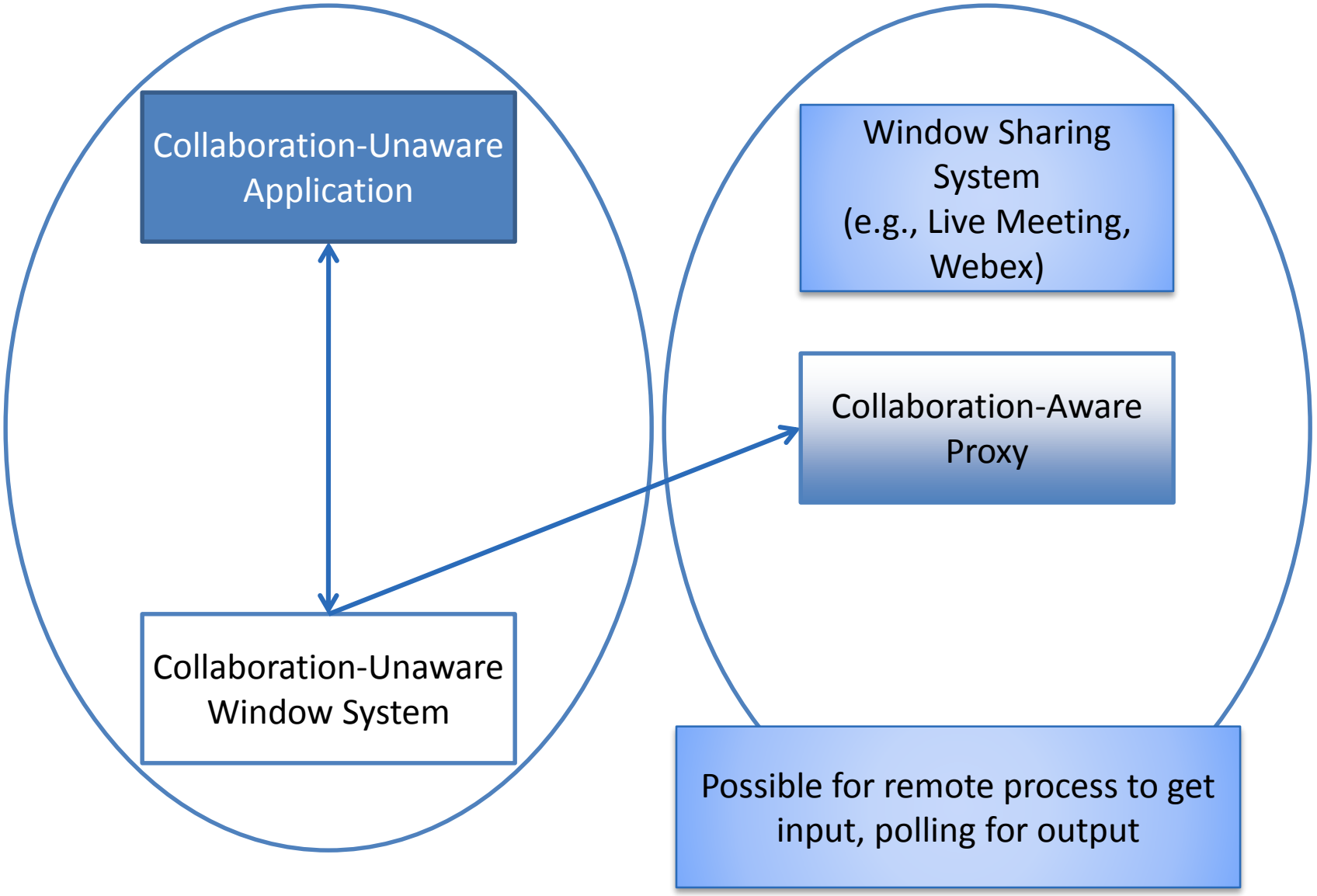
In Java window system is a library and not a separate process

Microsoft Windows

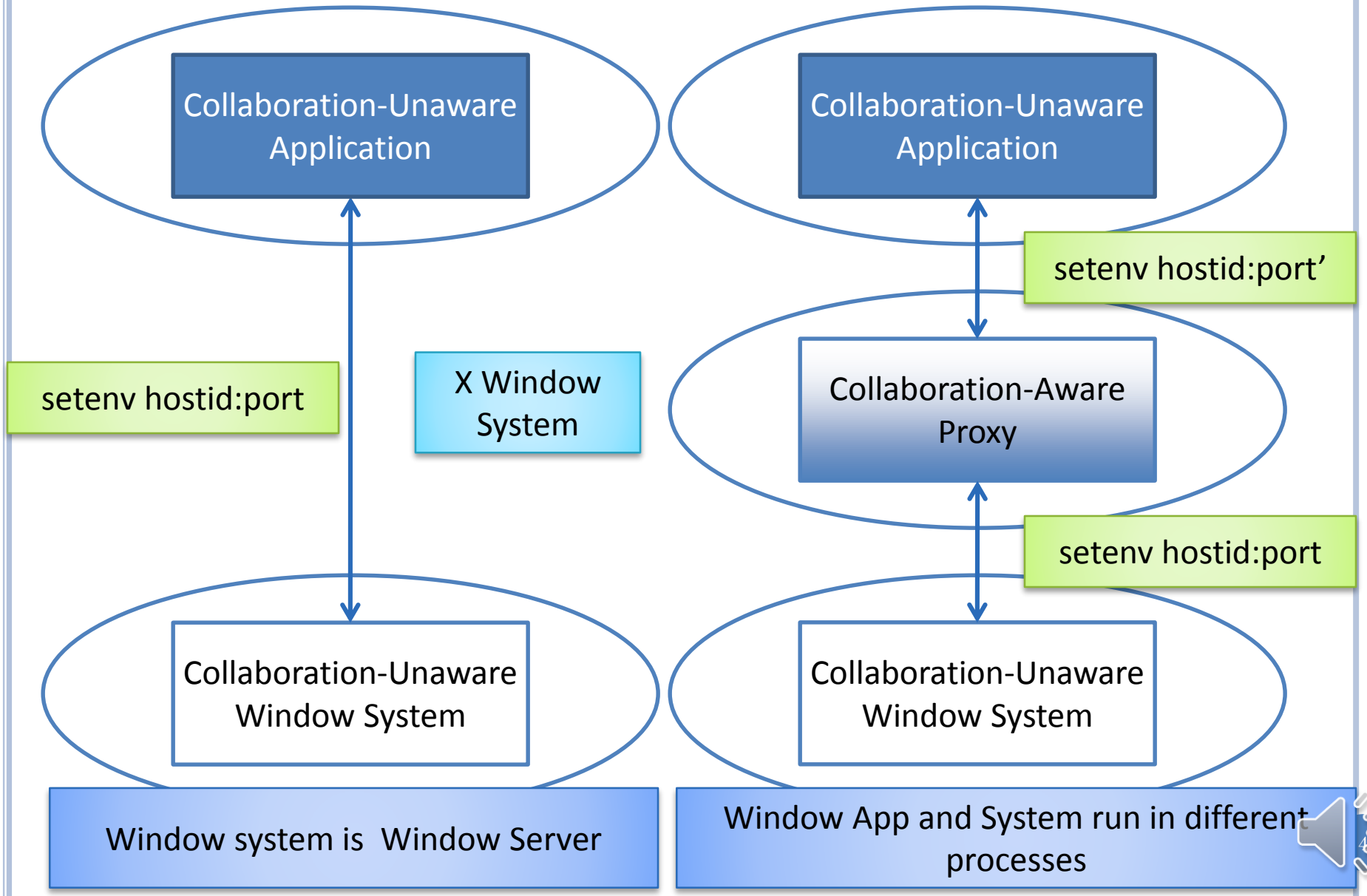
Window App and System run in same process

Library proxy possible but not common, have to run proxy code for some compiled process

Window System proxy



Proxy Addition in Client-Server Architecture



Shared Window Systems

- Coupling
  - WYSIWIS, Relaxed WYSIWIS
- Shared Window Systems
  - Constraints on input sequences can be violated
  - Layering for telepointers
- Replicated
  - Input broadcast
    - Input interception, injection
  - All problems of replicated models and no awareness
- Centralized
  - Input relay and output broadcast
    - Input interception, injection
    - Output interception, injection
  - All problems of centralized plus pop up menus, different exposed areas, output query calls
  - Virtual desktop solves problem of referential transparency vs. per user window configuration
- Java Implementation
  - Input interception and injection possible
    - Implementation problems as demo shows
  - Output interception and injection not directly supported
- Address space of proxies
  - Different from shared application in reality
  - Easier to attach proxies if distributed window systems such as X