Checkpoints of GUI-based Applications

Victor C. Zandy
University of Wisconsin – Madison
zandy@cs.wisc.edu
www.cs.wisc.edu/~zandy/guievict
Goal: Allow users to move with running programs to different computers.

How do we move a program that has a GUI?

• GUI state is hard to capture
Guievict

• Flexible, unpremeditated GUI mobility
• No advanced application preparation
  • *Hijack* your application when it is time to move
• Independent GUI mobility
  • Each application has access to its *window session*

• Complements recent GUI mobility activity
  • Fonts, rendering, device independence (XFree86)
GUI-based Applications

Desktop Host

Application

Window Server

Display

Keyboard

Mouse

User
GUI-based Applications

The application reference resources with resource identifiers

Application Window Resources
- Windows
- Pixmaps
- Fonts
- Cursors
- Graphics Contexts
- Colormaps

Server extension enumerates application’s resource identifiers
Hijacking the Application

Desktop Host

Application

Hijack

Evictor

Window Server

Window Session

Evict Extension
Hijacking the Application

Desktop Host

Application
Evict Library
Window Server
Window Session
Evict Extension

Hijack
Evictor
Detaching the GUI
Re-attaching the GUI

Desktop Host

- Application
  - Window Session
  - Evict Library

- Window Server
  - Evict Extension

Laptop Host

- Window Server
  - Evict Extension

Evictor

Re-attach
Application Migration

Desktop Host

Application
Window Session
Evict Library

Window Server
Evict Extension

Laptop Host

Application with detached GUI

Window Server
Evict Extension
GUI Replication

Bart’s Desktop Host

Evictor

Window Server

Evict Extension

Vic’s Laptop Host

Application

Window Session

Evict Library

Guimux

Window Session

Window Server

Evict Extension

Attach
Technical Issues

• Hijacking the application (Dyninst)
  • Find connection to server
  • Synchronize communication
  • Handle statically-linked executables

• Retrieve window session
  • Resource identifiers and resource state
  • Fonts are especially troublesome
Finding the Server Connection

Application
Evict Library

File Descriptor Table
0 1 2

Make a new connection to the port.
Does it quack like a window server?

NAT or Firewall
Window Server
Synchronizing the Connection

We must ensure that window state does not change as we capture the window session.

Changes are caused by messages exchanged between application and server.

Flush in-flight messages
  • Insert sync message

Allow partially sent messages to complete
  • Walk process stack
Retrieving Resources

1. Obtain resource identifiers
2. Obtain resource state

Application
Evict Library

X Requests
Resource State

Window Server
Window Session
Evict Extension
Fonts

Fonts are loaded by name.

There is no way to map a font identifier back to its name

• Not even the server has the information!

X provides:

• map from identifiers to font geometry
• enumeration of available fonts and their geometry

Expensive today, but client-side fonts are on the way.
Performance

Detach Latency (Netscape)
  • 21.7 sec with full font search
  • 0.6 sec with font geometry cached

Reattach Latency (Netscape)
  • 0.7 sec

Guimux overhead
  • 40 usec per round-trip message
  • Imperceptible
Summary

• Guievict enables flexible, unpremeditated GUI migration
• Key: Access to server-side application window session

www.cs.wisc.edu/~zandy/guievict

Come see the demo!