Hybrid Resource Control for Active Extensions

Parveen Patel
Jay Lepreau

University of Utah
The Problem

- Resource-greedy active code
- Resource control of untrusted code
  - CPU, memory, network bandwidth

- Context: Active Extensions
  - Code downloaded via the control channel
  - Examples: Application Layer Gateways, Multicast scoping agents
Current Solution #1: Dynamic

- “Sandbox” the active code
- Run-time checks in the critical path
- Asynchronous termination
  - Requires checks at the “user-kernel” boundary to protect integrity of the “kernel” code
- Flexible
- Examples: Janos, Smart Packets, RCANE, OKE Corral
Current Solution #2: Static Analysis

- Constrained programming model bounds resource consumption
- Admission control == Resource control
- Examples: PLAN, SNAP, PCC

Issue: Existing work does not yet address the problem with pessimistic estimates, valid code gets rejected.
Current Solutions - Summary

- **dynamic checking**
  - run-time overhead
  - asynchronous termination

- **static checking is very conservative**
Hybrid Resource Control #1

- Static checking
  - Constrained programming model to bound the resources and guarantee termination
  - Static analysis rejects resource greedy code from the “kernel” fast-path environment
  - Liberal resource limits
Hybrid resource control #2

- Dynamic resource accounting
  - Detects misbehavior
    - Misbehaving code is detected and unloaded only when idle (between packets)
  - Limits overall resource consumption
Poll points

- Extension could cause packet drops at device input queue

- Split the active extension code and poll network interfaces

- Adds some runtime cost
Merits of Hybrid Resource Control

- No asynchronous termination
  - Implies no runtime checks at the “user-kernel” boundary

- Reduced runtime overhead
  - Runtime accounting checks are inexpensive

- Flexibility via “poll points”

- DoS prevention
Outline

- Prototype: resource bounded Click or RBClick
  - Building blocks
  - The big picture
  - Preliminary evaluation
Cyclone

- Cyclone: typesafe C-like language from Cornell and AT&T
  - Region-based memory management
  - Control over data-representation
  - Easy to interface with C
  - Namespaces
Resource-bounded Cyclone

- Namespace control

- Restricted programming constructs (bounded loops)

- Memory management via 4 distinct dynamic regions
  - Per-packet
  - Packet-cache
  - Inter-packet
  - Global memory
Click

- Modular router toolkit from MIT
- Data-flow programming model
- Has an increasingly large base of router extensions
Prototype: Architecture

- An active extension is a special Click graph
  - Mix of trusted and untrusted elements
  - Statically analyzed

- Admitted to kernel fast-path
An Active Extension

Trusted C++ Element

Untrusted RBCyclone Element
The big picture

Code Analysis Tool

Element Resource bounds
The big picture

Poll Element

Graph Analysis Tool

Element Resource bounds

Overall Resource bounds
Loop configuration

Element Resource bounds

Overall Resource bounds
Evaluation

- Flexibility of programming model
- Experimental performance gains
Classification of Click elements

- Categorized all 234 Click v1.2.1 elements into 7 different classes based on their resource use
  - E1 - *Constant* resource consumption
  - E2 - ~ *length of the packet*
  - E3 - ~ *length of some protocol header*
  - E4 - ~ *length of element configuration*
  - E5 - ~ *some value in the configuration* of an element.
  - E6 - ~ *field in a protocol header*
  - E7 - Potentially *unbounded*
Evaluation: flexibility

- Results:
  - 88% resource-bounded
  - The rest can be easily rewritten to be bounded
- Demonstrates that RBClick can reuse a rich set of Click elements
- Strongly suggests that RBCyclone programming model is sufficiently expressive
Prototype Context

- Janos

![Diagram of Janos components]

- AA1
- AA2
- AA3
- AA4

- ANTS2/Bees
- Janos Java NodeOS Bindings
- Janos Virtual Machine
- Moab
- OSKit

- Active Applications (AAs)
- Execution Environment (EE)
- NodeOS
Evaluation – experiment configurations
Evaluation: performance

- Moab Fast path
- RBClick
- RBCyclone
- Click
Conclusion

- Hybrid resource control
  - Static analysis reduces runtime overhead
  - Dynamic accounting allows liberal admission control

- RBCyclone is expressive and practical
  (“tastes great”)

- RBClick doubles forwarding rate in Janos
  (“less filling”)