Resource Containers

Gaurav Banga Peter Druschel Jeffrey C. Mogul

Presented by Michael Roitzsch

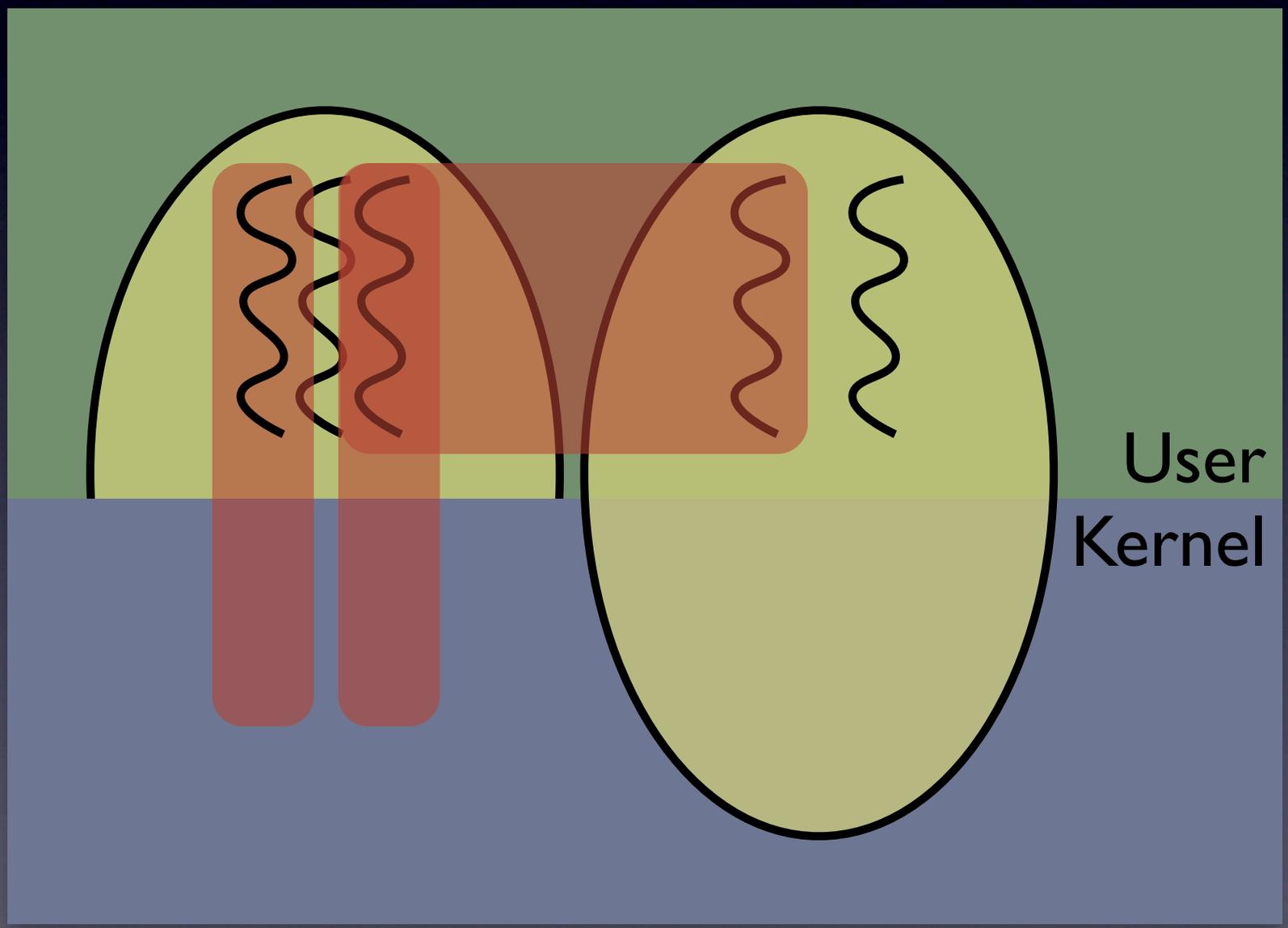
Observations

- mismatch: OS‘es resource management design assumptions and behavior of modern server applications
  - no control over resource consumption by the kernel on behalf of the application
  - difficulty to express priority policies, QoS
  - denial of service

Web Server Example

HTTP Connection Handling	CGI Handling
process forking	CGI process forking
pre-forked processes	persistent CGI processes
event driven	library based
multi-threaded	

Current Resource Management



Dual Use of Processes

Process

Resource Container

Protection Domain

Resource Principal

Isolation

Accounting

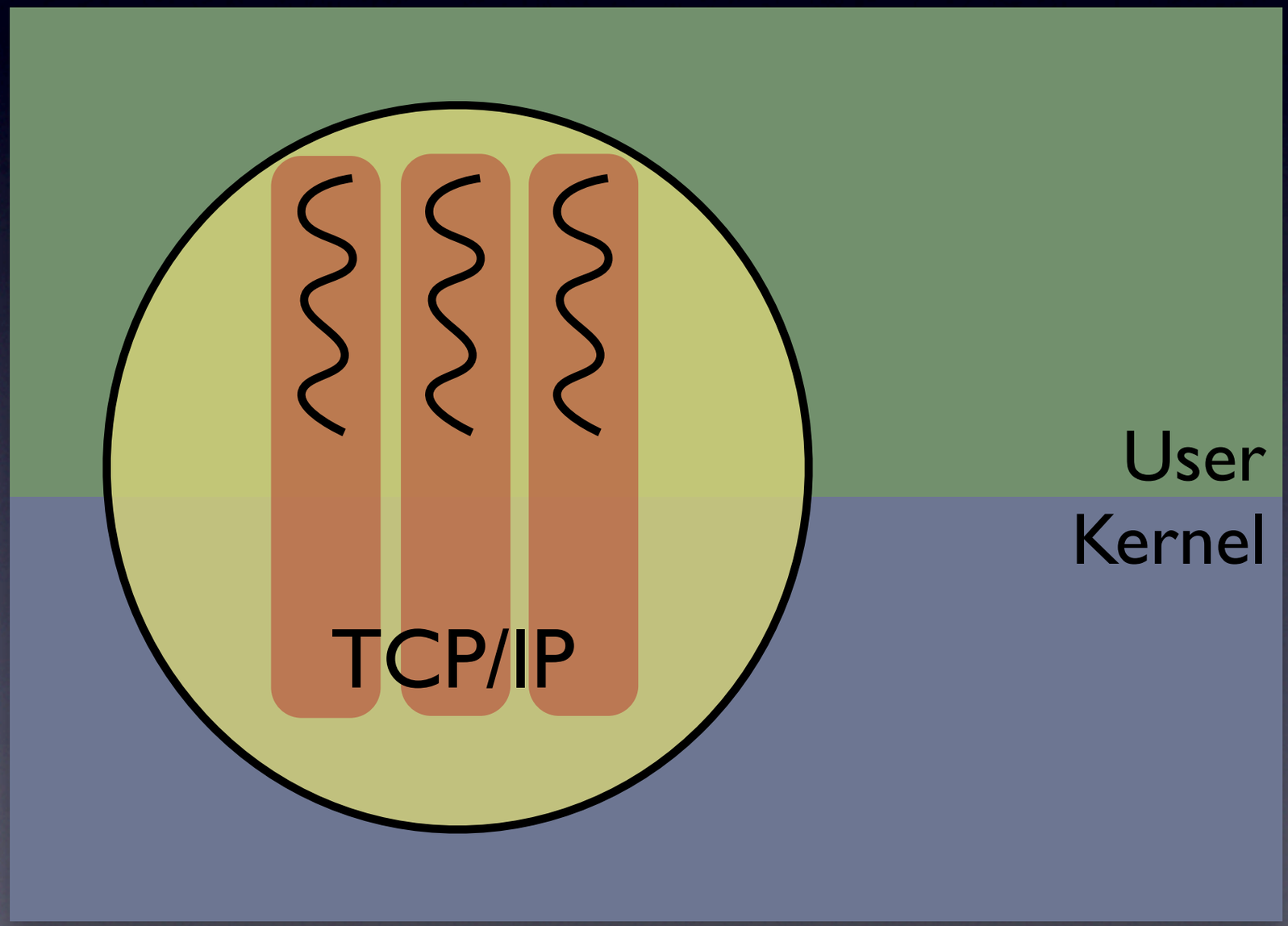
Resource Containers

- abstract operating system entity
- contains resources used for an activity, kernel accounts against resource containers
- arbitrary relationships between protection domains, threads, resource containers
- mechanism to be used with resource management policy
- form a hierarchy

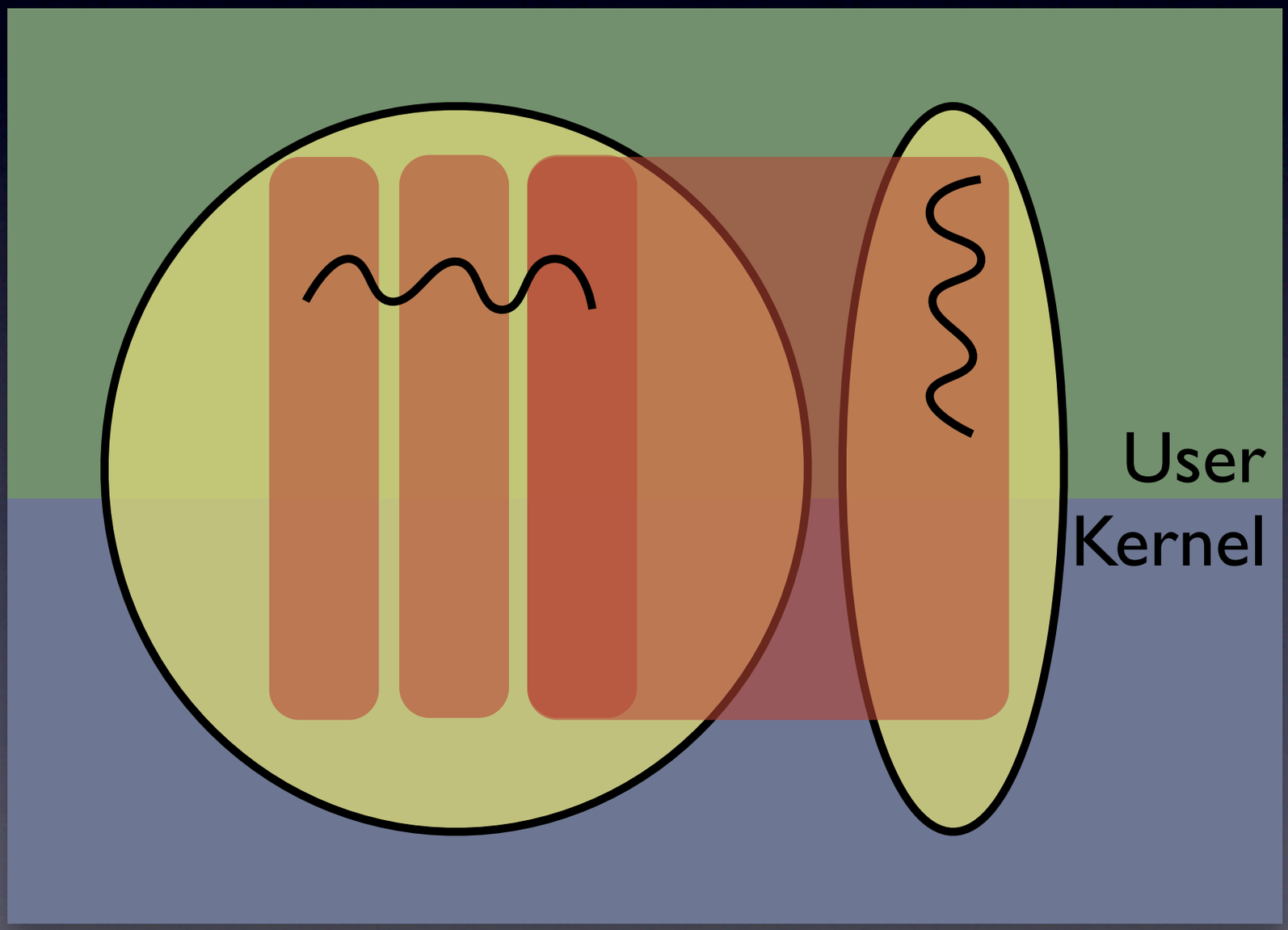
Operations

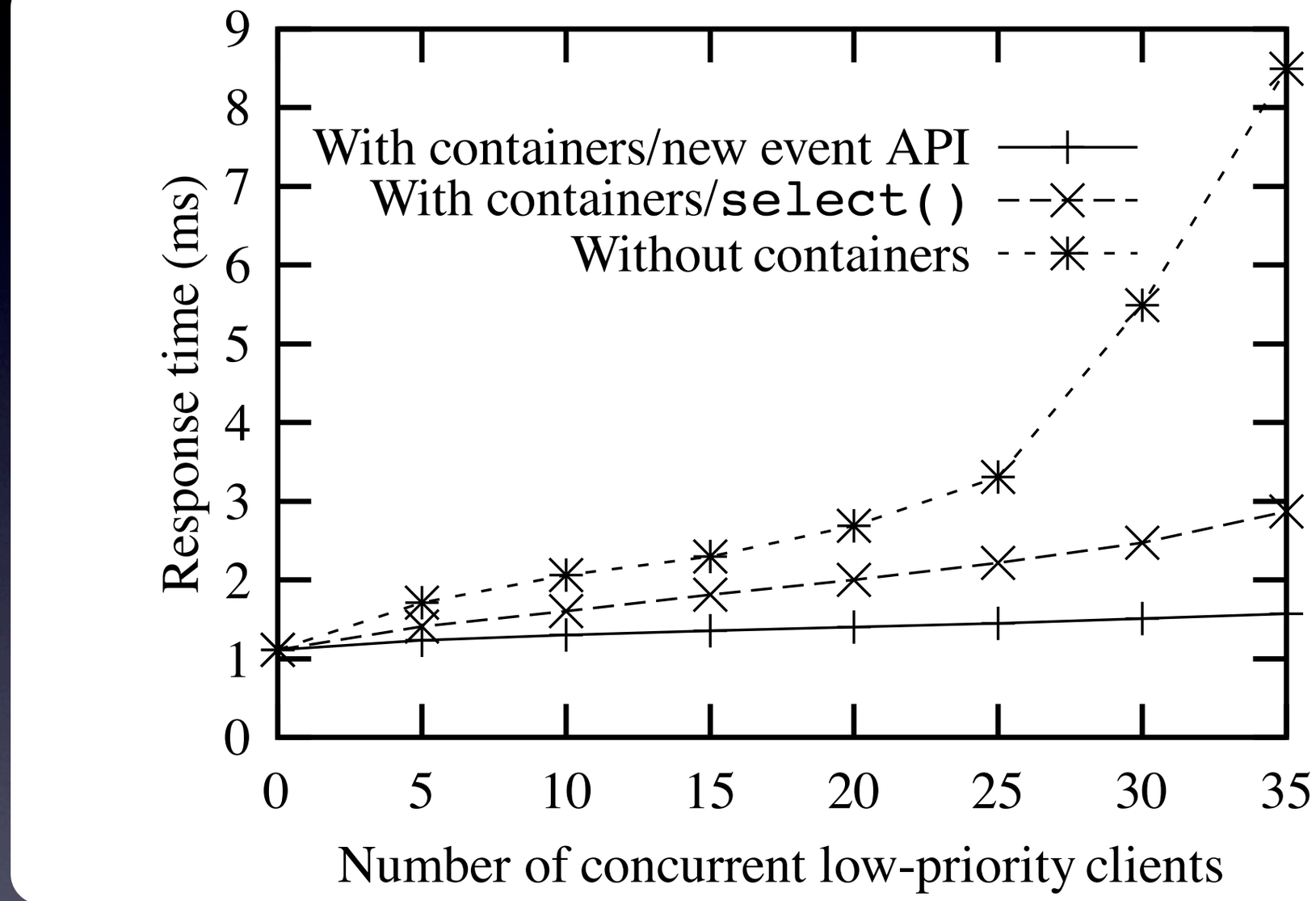
- creating a new container
- set a container‘s parent
- container release
- sharing containers between processes
- container attributes
- container usage information
- binding a thread to a container
- reset the scheduler binding
- binding a socket or a file to a container

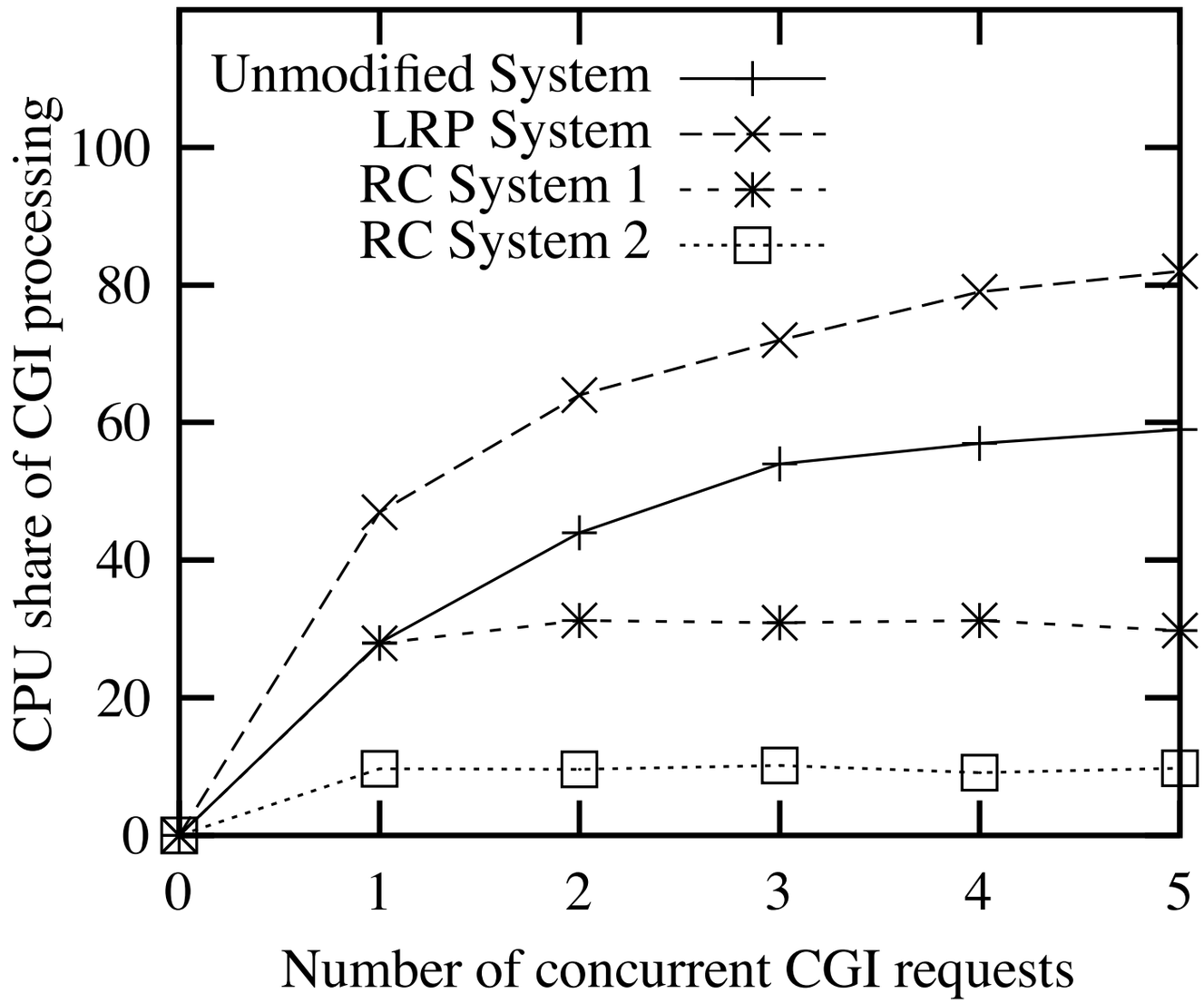
Multi-Threaded Server

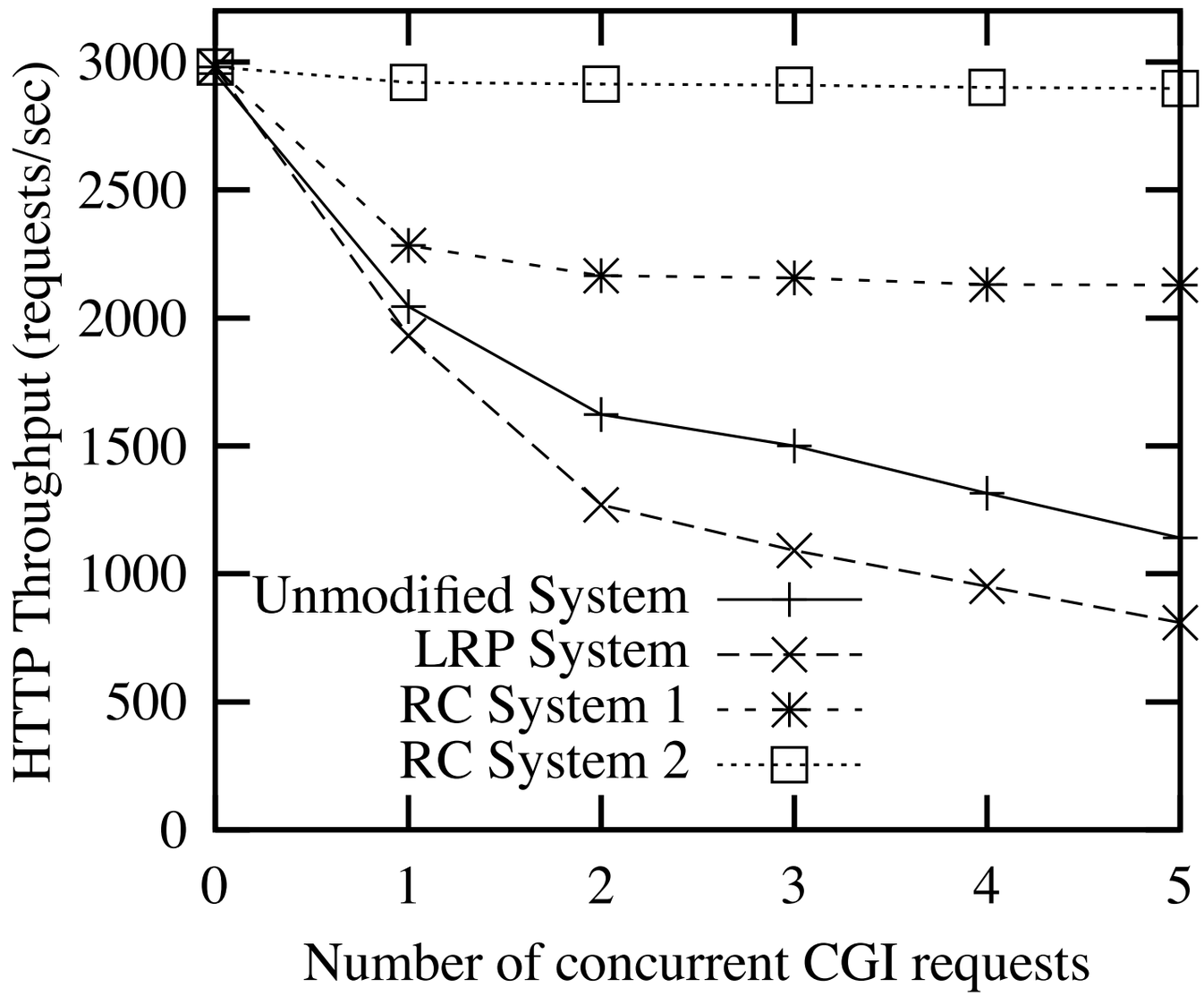


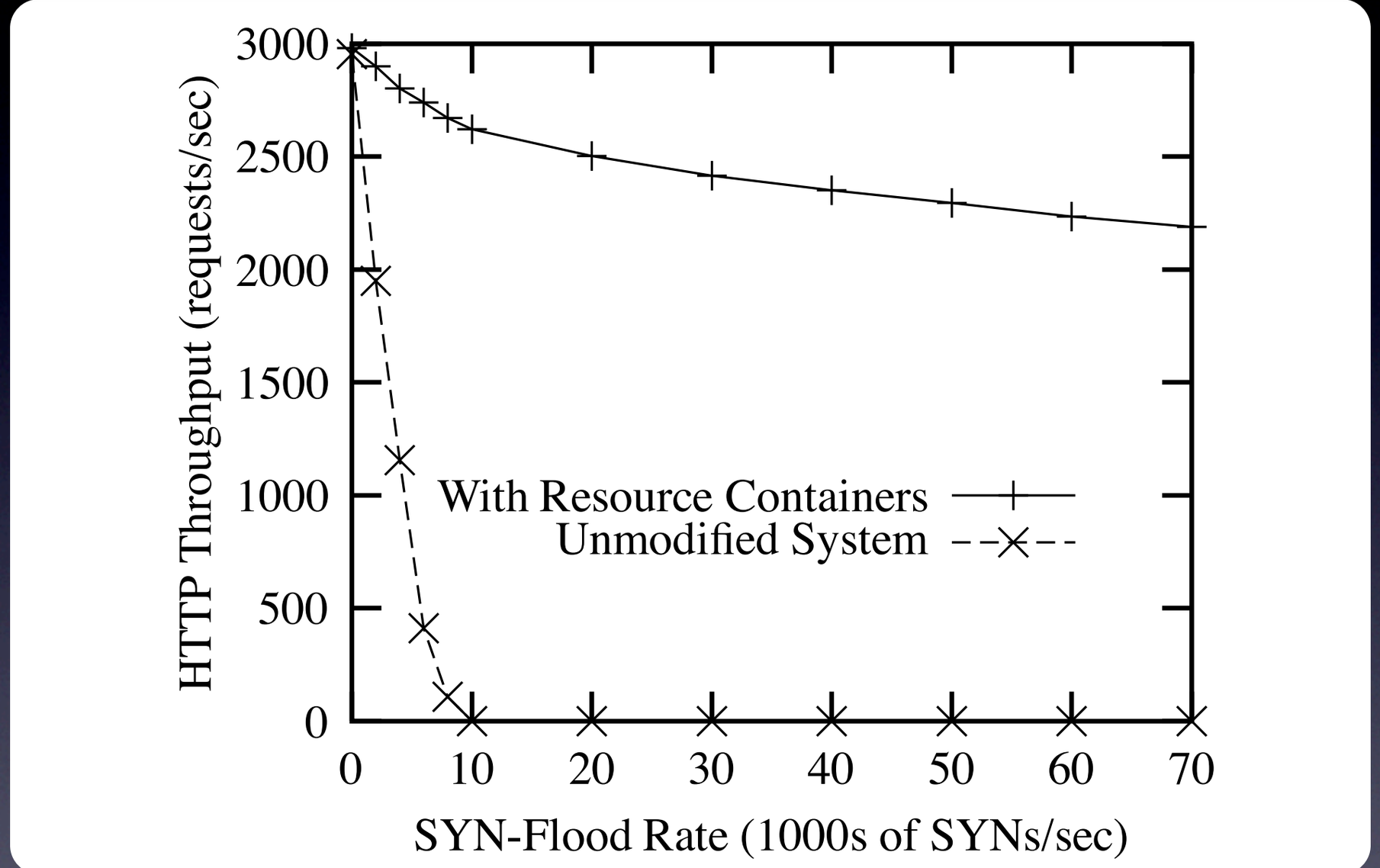
Event-Driven Server




With containers/new event API
With containers/select()
Without containers
Response time (ms)
Number of concurrent low-priority clients
0
1
2
3
4
5
6
7
8
9
0
5
10
15
20
25
30
35


Unmodified System
LRP System
RC System 1
RC System 2
CPU share of CGI processing
0
20
40
60
80
100
0
1
2
3
4
5
Number of concurrent CGI requests


HTTP Throughput (requests/sec)
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
Number of concurrent CGI requests
Unmodified System
LRP System
RC System 1
RC System 2


HTTP Throughput (requests/sec)
3000
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
SYN-Flood Rate (1000s of SYNs/sec)
With Resource Containers
Unmodified System

Discussion

- How do microkernels solve such problems?
- Why can a container be reparented or even made parent-less?
- How to account kernel resource consumption properly?
- Is the motivation for the problem still valid?