Towards Integrated Runtime Solutions in QoS-aware Middleware

Multimedia Middleware Workshop '01

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Outline

- Overview of runtime middleware solutions
- Why an integrated solution?
- Insights about "triggering sources"
- A proposed integrated run-time architecture
- Concluding remarks

Applications in the future …

- Next generation multimedia applications
  - Very-low bit rate MPEG-4 streaming
  - Voice-over-IP
- Developed in an ad-hoc fashion
  - Tailored to a specific platform or OS
- What we need …
  - Highly scalable to a wide variety of heterogeneous devices
  - Highly available across wide-area distributed environments

Assumptions

- Key assumption: applications are componentized (roughly) to multimedia services and consumers
- As a result: various service configurations are possible
  - These may be selected by the middleware, based on the "environment", at **run-time**
    - What is the "environment" of an application?
  - The goal: adjust application behavior to accommodate to changes, providing better QoS

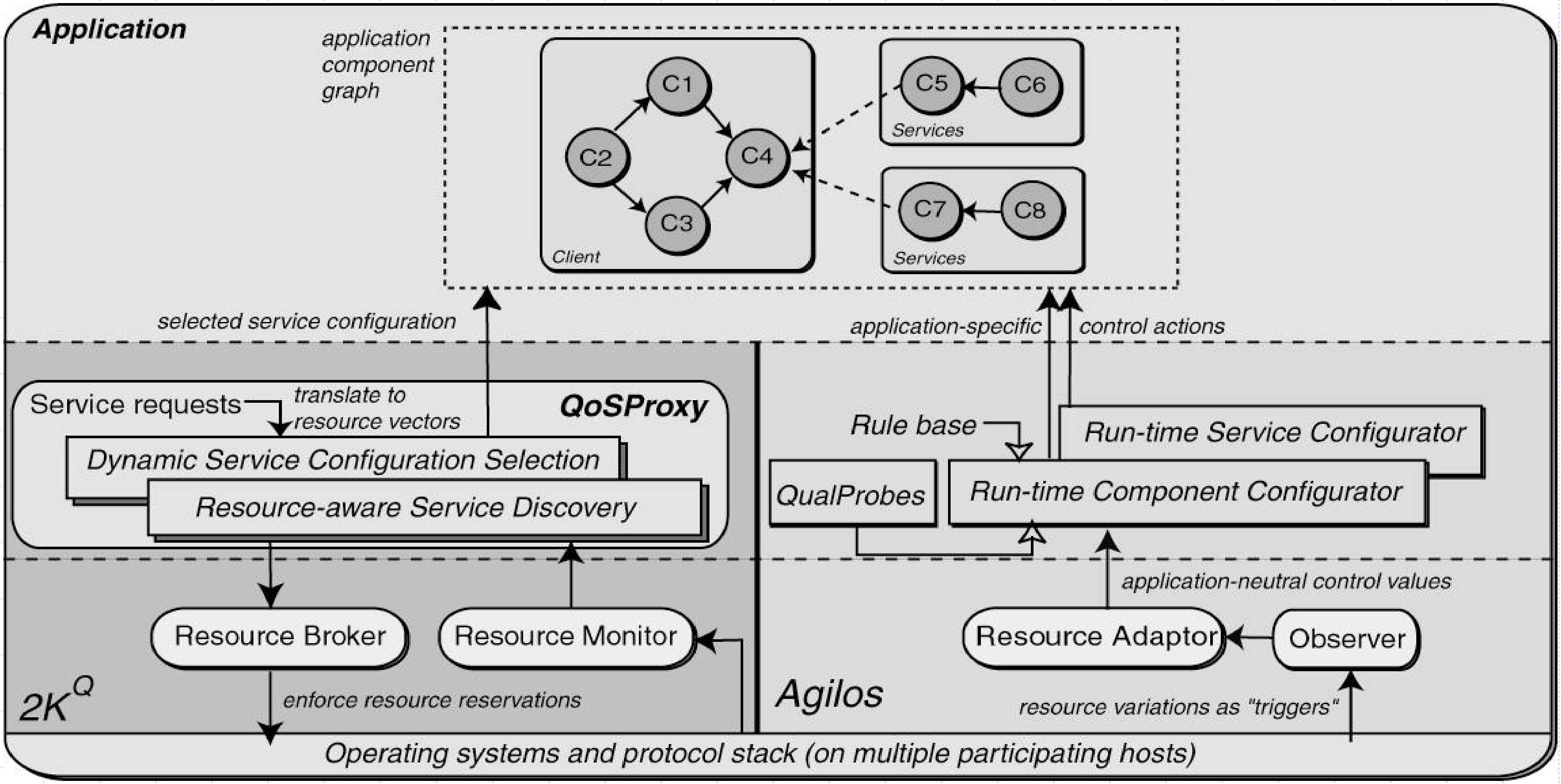
What kinds of run-time support?

- Probing (or monitoring)
  - off-line or on-line
- Instantiating
  - Choosing a specific service configuration
- Adapting
  - Making changes to the application at run-time

Our Past Experiences

- 2KQ: focusing on
  - Translates between application-level and OS-level QoS parameters
    - Before instantiation
  - Instantiates a specific service configuration
    - During application run-time
- Agilos: focusing on
  - Probing
  - Adapting
    - Using a processing engine to generate adaptation decisions

Existing Run-time Support


Application
application component graph
C1
C2
C3
C4
Client
C5
C6
Services
C7
C8
Services
selected service configuration
application-specific
control actions
Service requests
translate to resource vectors
QoSProxy
Dynamic Service Configuration Selection
Resource-aware Service Discovery
Rule base
Run-time Service Configurator
QualProbes
Run-time Component Configurator
application-neutral control values
Resource Broker
Resource Monitor
Resource Adaptor
Observer
$2K^Q$
enforce resource reservations
Agilos
resource variations as "triggers"
Operating systems and protocol stack (on multiple participating hosts)

Key Observation

- The "driving force", or "triggering sources", are identical
  - For probing, instantiating and adapting the applications
- Can we make the implementation of the run-time support as simple as possible?
  - By providing a unified decision-making process to configure and adapt the applications in a coherent fashion

What are the triggering sources?

- Variations in resource availability
  - Client-side parameters (CPU, bandwidth)
  - Server-side parameters
  - Network performance
- Represented as a “label” in the application component graph
  - So that they may be coherently monitored by on-line QoS probing components
  - Inbound, outbound, core and edge

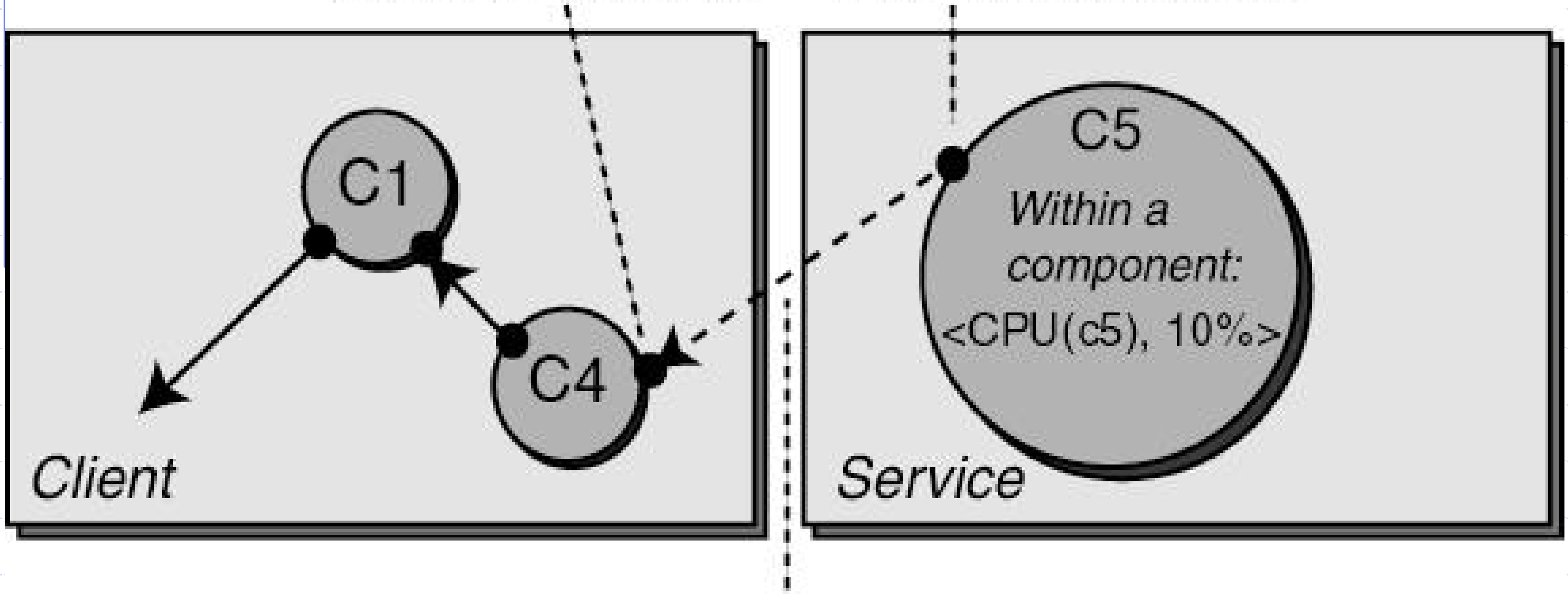
Application Component Graph

At inbound interface:

<BW^{in}(c4), 1800Kbps>

At outbound interface:

<BW^{out}(c5), 1800Kbps>



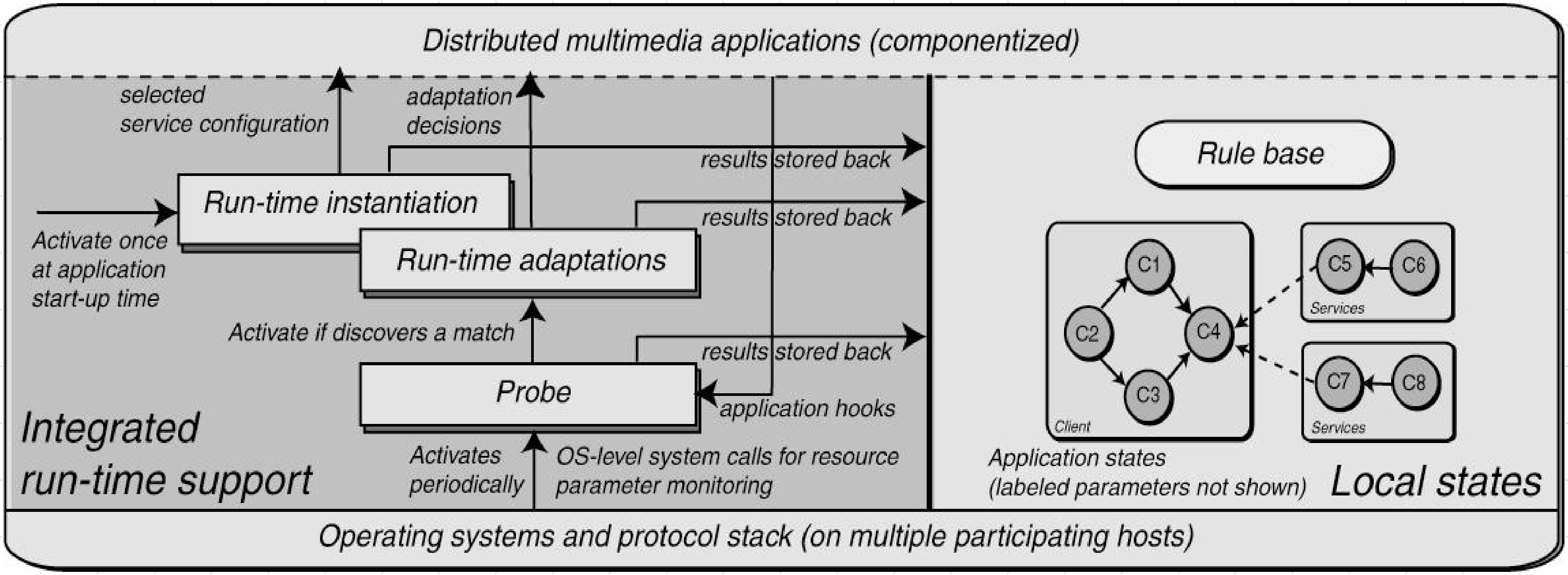
Between components from different hosts:

<Delay(c4, c5), 200ms>

What are the triggering sources?

- User preferences
  - Tradeoff policy among parameters
  - Level of satisfaction (QoP)
- User mobility
  - Treated as the rule, not an exception
  - Application component graph may be disrupted (changed)
  - Location of component: may be treated as a core component

An Integrated Run-time Architecture


Distributed multimedia applications (componentized)
selected service configuration
adaptation decisions
results stored back
Run-time instantiation
results stored back
Activate once at application start-up time
Run-time adaptations
Activate if discovers a match
results stored back
Probe
application hooks
Integrated run-time support
Activates periodically
OS-level system calls for resource parameter monitoring
Rule base
C1
C2
C3
C4
Client
C5
C6
Services
C7
C8
Services
Application states (labeled parameters not shown)
Local states
Operating systems and protocol stack (on multiple participating hosts)

Concluding remarks

- Key point: make the run-time support at middleware level as simple as possible
  - While still trying to be effective for as many applications as possible
- Route taken:
  - Identify a series of identical triggering sources
  - Propose a simpler framework

Questions?

http://www.eecg.toronto.edu/~bli/

http://cairo.cs.uiuc.edu/