QoS for (Web) Applications
Velocity EU 2011

william.louth@jinspired.com
Self Adaptive Software

“Self Adaptive Software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.”

DARPA
Feedback Loops

Evidence, Relevance, Consequence, Action
Self Observation

Observe, Optimize, Control, Collect
Self Control

Control, Process, Sensor
DIRT - Data in Realtime

Remote, Past, Local, Present, Predicted
99.9999% => PP.PPPP

Profile, Protect, Police, Prioritize, Predict, Provide
Profile

instrument, measure, and collect performance data for a significant portion of an application code base
Protect

supervisory routines that are embedded into the runtime to govern the execution flow & consumption of resources
Police

watching for non-compliant behavior & taking corrective measures to enforce call flow contracts and agreements
Prioritize

differentiating the quality of service that is offered to applications, activities and actors
controllers that use past & current in-flight data & models to predict behavior & consumption and then to optimize
Provide resource capacity planning based on inflight activities & predicted their predicted requirements - in internet time.
QoS for Networks

“The ability of the network to provide better or “special” service to a set of packets/dataflows to the detriment of other packets/dataflows”
Traffic Characteristics

**Bandwidth**
the number of bits per second that can be expected to be transmitted

**Delay**
the elapse time between when a packet is first sent and when it arrives

**Jitter**
the variation in the arrival rate, or delay introduced, of packets sent at a uniform rate

**Packet loss**
all routers lose, drop of discard packets for a number of reasons
Traffic Planning

Identify
traffic and its requirements in terms of service levels

Divide and Associate
traffic into classes of service levels

Define
policies for each class of service level which cause some change in the traffic characteristic mentioned previously
Traffic Service Classification

differentiating one packet from another by examining fields inside of the packet’s header

in some cases this classification is performed automatically and adaptively based on traffic patterns and resource consumption behavior
Traffic Congestion Management

using queues and schedulers which split traffic across queues and schedulers based on service classification associated with a packet or flow
Traffic Policing and Shaping

**Policing**
traffic contracts which define how much data can be sent

**Shaping**
smoothing the peaks & troughs of data transmission in order to optimize or guarantee performance & bandwidth.
QoS for Applications

“The ability of the runtime to provide better or “special” service to a set of calls/threads to the detriment of other calls/threads”.
Call Characteristics

**Throughput**
the number of requests/calls per second that can expect to receive a valid response

**Response Time**
the time between a request/call is sent (received) & when it’s response is received (sent)

**Response Time Variation**
the variation in the response time of requests/calls

**Exceptions & Errors**
faults can occur in runtimes such as timeouts caused by contention and corruption
Call Planning

Identify
request/call patterns and their requirements in terms of service levels

Divide and Associate
request/call patterns into classes of service levels.

Define
policies for each class of service level which cause some change in the request/call characteristic mentioned previously
Call Service Classification

differentiation is typically done by examining the execution context associated with a request/call
Call Congestion Management

managed using concurrency and control constructs such as semaphores/latches/locks which are generally backed by implicit or explicit queues consisting of parked threads during contention
Call Policing & Shaping

**Policing**

call contracts which define how many calls can be sent

**Shaping**

it tries to conform the call rate to the call contract using some form of delay mechanism such as waiting and parking
## Network-to-Application Mapping

<table>
<thead>
<tr>
<th>Network</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>Call</td>
</tr>
<tr>
<td>Flow</td>
<td>Thread</td>
</tr>
<tr>
<td>Router</td>
<td>Method</td>
</tr>
<tr>
<td>Datalink</td>
<td>Call Site</td>
</tr>
<tr>
<td>Buffer</td>
<td>Resource</td>
</tr>
<tr>
<td>Queue</td>
<td>Semaphore</td>
</tr>
</tbody>
</table>
The Application is the Network
Activity Based (Resource) Metering

metering has the potential to revolutionize the software design, development, testing, and management

a metering standard would define a model, concepts and mechanism which can be used across languages and processing boundaries to control and cost consumption
# Methods of Measurement

<table>
<thead>
<tr>
<th>Method</th>
<th>Metering</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Causality</td>
<td>Correlation</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Event</td>
<td>Sampled</td>
</tr>
<tr>
<td>Allocation</td>
<td>Thread</td>
<td>Process</td>
</tr>
<tr>
<td>Assignment</td>
<td>Direct</td>
<td>Apportioned</td>
</tr>
</tbody>
</table>
# Metering Measurement Model

<table>
<thead>
<tr>
<th>Model</th>
<th>Activity</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Probe</td>
<td>Meter</td>
</tr>
<tr>
<td>Develop</td>
<td>Code</td>
<td>Counter</td>
</tr>
<tr>
<td>Design</td>
<td>Behavior</td>
<td>Usage</td>
</tr>
<tr>
<td>Data</td>
<td>Group</td>
<td>Metering</td>
</tr>
</tbody>
</table>
QoS Enablement

```java
public int func(...) {
    reserve(func);
    release(units);
}
```
Demo: Prioritization
Demo: Resource Reservation

Give Take Permit Pool
Demo: Rate Limiting

“Leaky” Bucket
QoS Reservation Lanes

resource pool

lane #1
not in range
within range
priority level
capacity limit

lane #2
within range

lane #N
within range

QoS Reservation Lanes
Fairer Web Page Servicing with Performance Credits
Fairer Web Page Servicing with Performance Credits
A Cloud Cortex

APM: Application Performance Management
CARS: Cost Aware Runtimes & Services
BI: Business Intelligence
QoS: Quality of Service
Cloud Services
More Information

http://opencore.jinspired.com