Scaling-up SLA Monitoring in Pervasive Environments

Engineering of Software Services for Pervasive Environments (ESSPE '07)

Dubrovnik, September 4, 2007

Antonia Bertolino*, Guglielmo De Angelis*, Sebastian Elbaum**, Antonino Sabetta*

* [bertolino,deangelis,sabetta]@isti.cnr.it
** elbaum@cse.unl.edu

ISTI-CNR, Pisa, Italy
Univ. of Nebraska, Lincoln, USA
Motivation and Context

» To enable QoS management in pervasive systems
» To check SLAs and to report violations in an efficient and timely manner
Example scenario

» Fraud detection services (FDS):
  » For online sellers: to detect suspicious transactions and illegitimate payments
  » For buyers: to verify that sellers can be trusted (e.g. items sold are authentic)
  » *Different types of service requests, depth of checks, users, locations*
Example scenario

- Services accessed through many different pervasive devices
- Clients may have different profiles and QoS requirements
- SLAs can be complex, and possibly involve application-specific conditions
- QoS level, which would otherwise be fine, might suffer just because we are monitoring it

Requests coming from users of class GOLD who have been registered for more than a year and have used the service less than 10 times in the last hour must be served in less than 1500ms.
Checking SLAs is not weightless!

Data series: plot-data

- "active+1-2-3-4.dat" using 1:2
- "active+1+2-3-4.dat" using 1:2
- "active+1+2+3-4.dat" using 1:2
- "active+1+2+3+4.dat" using 1:2

how many clients we can't afford serving because we are busy monitoring (irrelevant events)?
Different clients, different “distance from violation”
Different clients, different “distance from violation”
Different clients, different “distance from violation”
Different clients, different “distance from violation”
Different clients, different “distance from violation”
**Key idea**

Goal of monitoring: to reveal SLA violations

» Ideally, at a given instant:
  » Monitor only the interactions for which violations occur
  » Ignore (=don't log, don't check) all the others
A smarter way to do SLA checking

» Dedicate **more attention** to interactions that are **more likely to violate** an SLA

» Reduce checking activity for interactions that are far from violation

» Shift SLA-checking effort **dynamically and automatically** to save resources
Different clients, different “distance from violation”
How Opportunistic SLA Checking works

- Analyze every event
- Discard some events
- Discard more

Diagram:

- Client A: SLA constraint
- Client B: SLA constraint

- T1
- T2
Standard SLA checking infrastructure
Opportunistic SLA checking infrastructure
Prototype behaviour
Discussion

Approach assumes that:

» QoS fluctuations are slow enough to enable prediction

» There is enough variability among clients (service requested, usage profiles, SLAs)
Discussion

» Different optimization goals:
  » Save storage (!)
    » Always possible, with considerable gain if missing some violations is not a problem
  » Save CPU utilization (?)
    » Only if the checks are heavy (complex SLAs)
  » **Trade-off** between efficiency and accuracy
Challenges and opportunities

- The sampling mechanism does have an (albeit light) overhead
  - If just simple checks are needed, the overhead may exceed the optimization obtained by sampling

- Optimize resource consumption:
  - Approach reduces the use of storage
  - May also reduce cpu load

- Application-specific constraints can be heavier to verify; checking them may be well worth optimizing
Summary

» **Goal:** to scale-up the ability to check complex SLAs

» **Approach:** leverage users' variability to save resources by *shifting the attention* to the interactions that are more critical (i.e. closer to violation)

» Trade-off: observe as many violations as possible, saving as much as possible on resources
Open Issues

For the Opportunistic SLA Checking approach:

- Analyze the tradeoffs associated with the sampling overhead
- Identify classes of SLAs (or SLA clauses) for which an opportunistic approach is feasible/advantageous
- Develop support to leverage OSLAC for violation isolation and regression testing activities

For QoS monitoring in general:

- How to devise monitoring infrastructures that are effective and timely, but do not interfere with the very QoS of the services they are meant to monitor