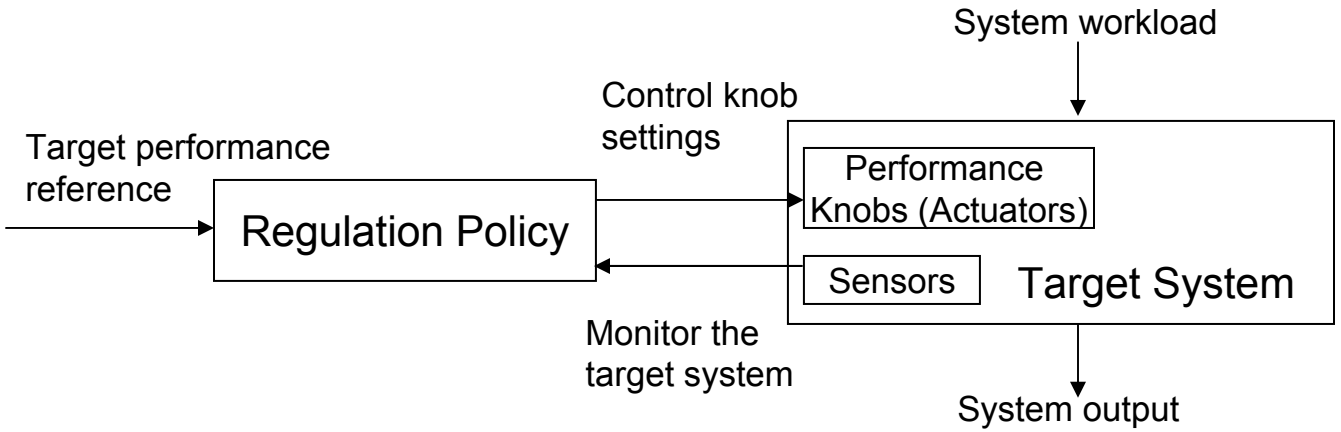
AdaptGuard: Guarding Adaptive Systems from Instability

Jin Heo and Tarek Abdelzaher

University of Illinois at Urbana-Champaign

Adaptive Systems and Performance Mgmt.



- Significant work on automatic performance management
  - Admission control
  - Power management
  - Queue management
- Regulation policy
  - Implements closed loop behavior: feedback control
  - Sensors → regulation policy → actuators
  - Assumes a model of system behavior
    - E.g., QoS-adaptive Web server:
      - Increased CPU utilization → Increased response time

Growing Challenges in Adaptive Systems

- Adaptive systems are getting more complex
  - More knobs and more distributed and dynamic environment
  - Assumptions made by regulation policies on target system may be violated
- Broken assumptions on system behavior → Respond to external stimuli in the wrong direction → poor performance

- This paper addresses the issue of detecting and recovering from instability of adaptive systems at runtime.

Contribution of AdaptGuard

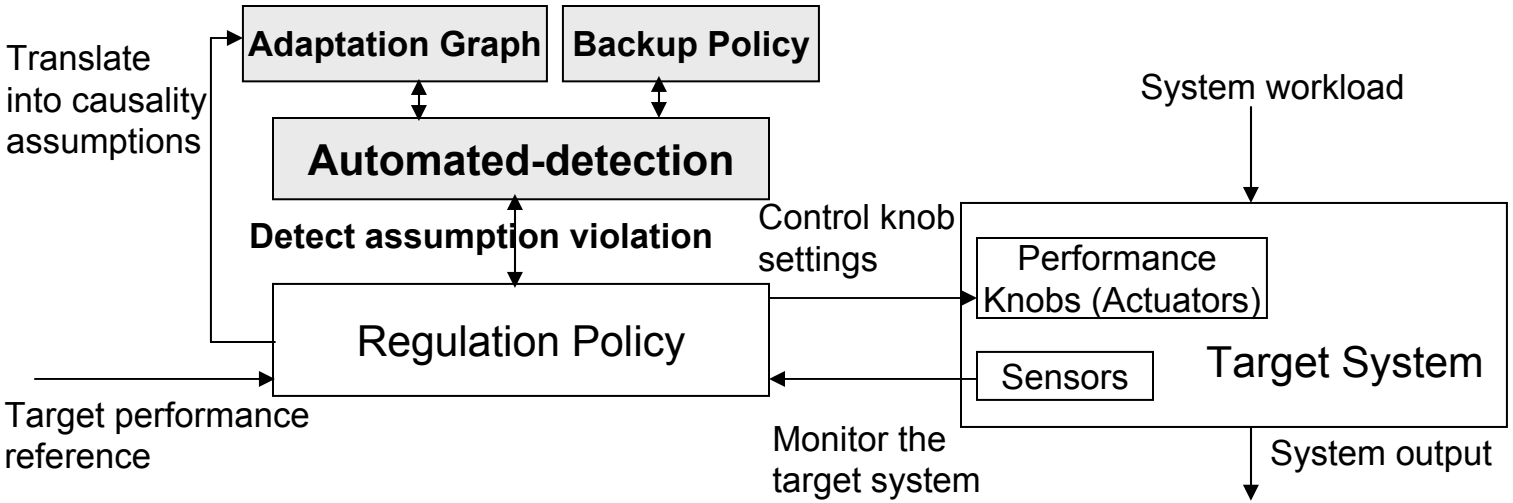
- A Software Service for guarding adaptive systems from instability
  - **Detect** system instability automatically
    - Specialized in closed loop behavior
    - Application-independent precondition of instability
    - Without *a priori* knowledge and full observabillity of system state
  - **Recover** from the instability when possible
    - Backup policy – open loop action
- Evaluation using fault injections and a case study on a QoS-adaptive Web server

Related Work

- Offline automated detection and diagnosis mechanisms
  - E.g., troubleshooting problems caused by misconfiguration, isolation of performance problems
  - Complex statistical methods
  - Root cause
- Run-time monitoring/verification techniques
  - Needs a priori knowledge
    - Safety properties explicitly specified
- Online automated detection and diagnosis mechanisms
  - Without a priori knowledge
  - Infers system behavior using quantitative models
    - Probabilistic models, linear models
  - More precise explanation
  - But false alarms when underlying model changes

Contents

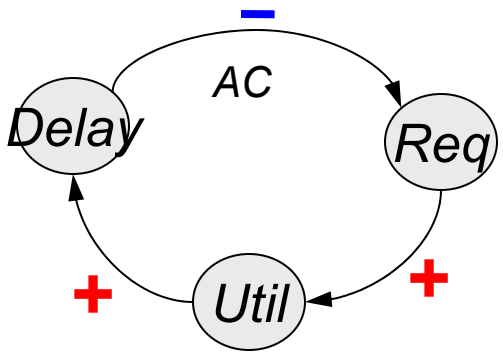
AdaptGuard: Overview



- Automatically learns "normal" behavior of closed loop actions
  - adaptation graphs - a set of assumptions
  - Doesn't use any qualitative models – robust to model changes.
- Monitors system state using adaptation graphs
- When instability detected, culprit loop is stopped

Detection of Potential Instability:
Introduction to Adaptation Graphs

- Causality assumptions: A affects B: $A \rightarrow B$
  - Changes in A cause changes in B
  - Direction of change (+, -)
  - Natural consequences or programmed behavior
- Adaptation graphs
  - Captures closed loop behavior
  - Graphical representation of a set of causal assumptions among performance control knobs and system performance metrics
  - The sign of a cycle: multiplication of the signs of all edges



Adaptation graph for QoS-adaptive Web Server

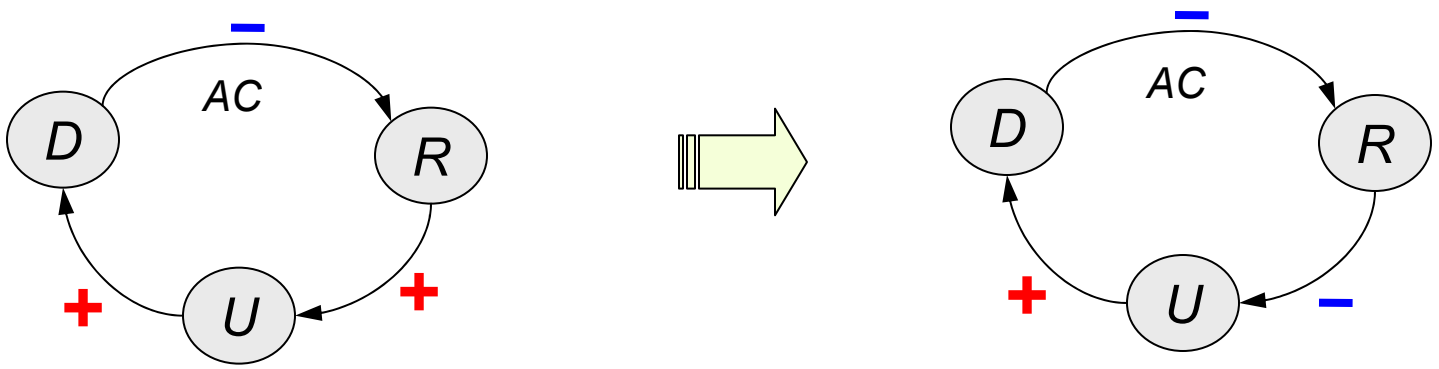
AdaptGuard automatically learns adaptation graphs

Detection of Potential Instability: The Rules of Stability

- The sign of all cycles in the graph should be negative
  - A cycle represents a feedback control loop
  - Directly comes from control theory

- AdaptGuard detects system instability using this rule
  - Application independent
  - Performed without revealing root cause

Example 1: QoS-Adaptive Web Server



Sign of the loop: - * + * + = -

Sign of the loop: - * - * + = +

Left figure: stable since the sign is negative, right figure: unstable since the sign if positive

→AdaptGuard will monitor the sign of the cycles to anticipate instability

Contents

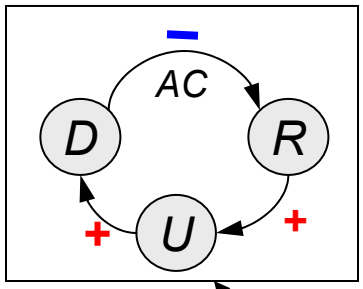
1. Infer causality assumptions

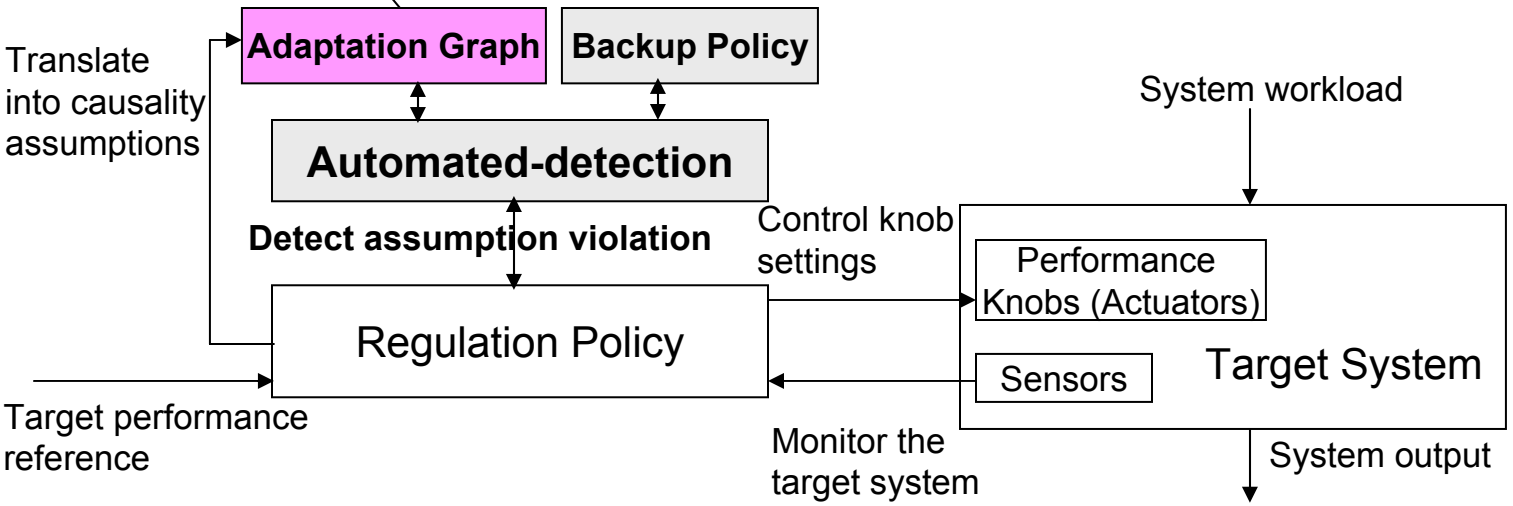
Construct an adaptation graph

Considers only the variables used in the control loops

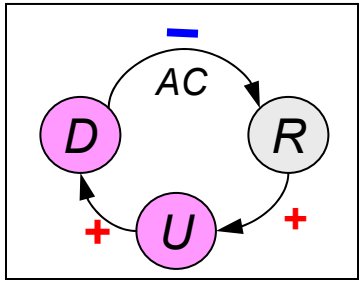
Only a small number of variables

All variables are known: AdaptGuard provides interfaces between the objects





1. Infer causality assumptions Cont.



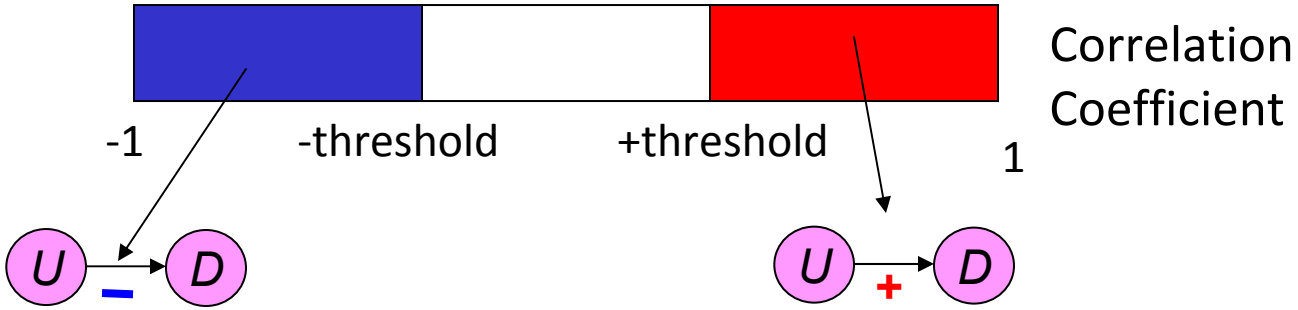
Pair-wise causality assumptions

To determine U→D:

Calculate (time-displaced) correlation coefficient of U and D

The most recent *N* values of *U* at time *t-k* and *D* at time *k*

Map the value [-1,1] →[+,-] using a threshold

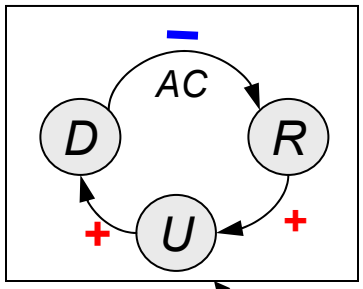


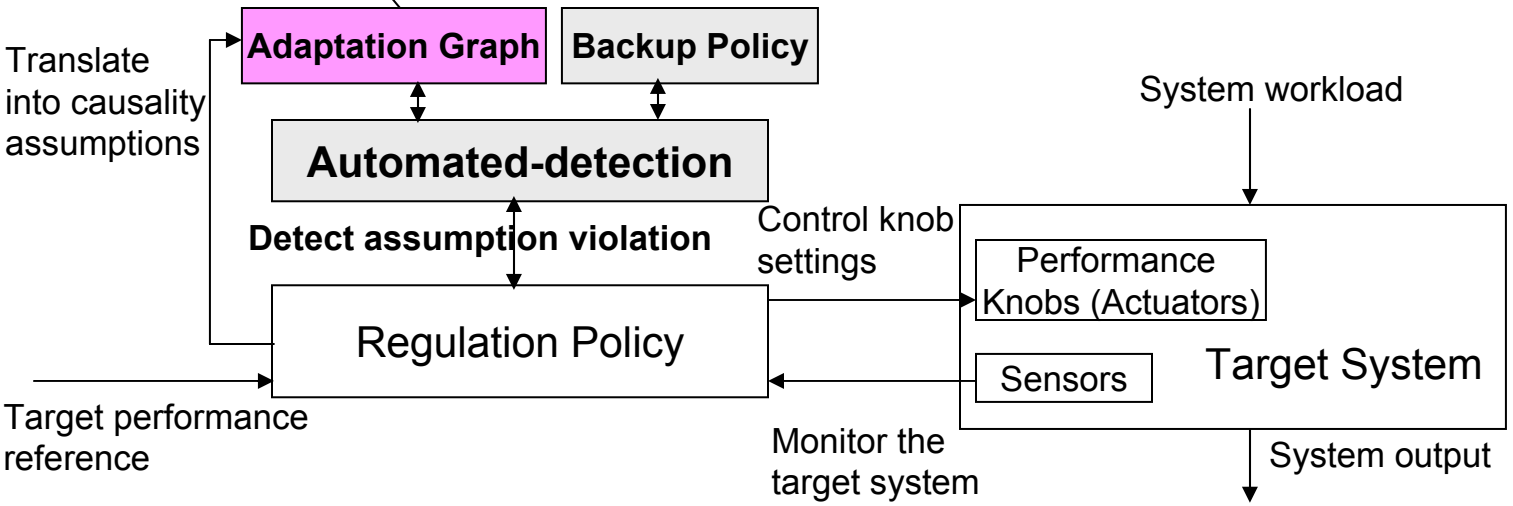
Apply the same procedure for *D→U* (*[D(t-k), U(t)]*)

1. Infer causality assumptions Cont.

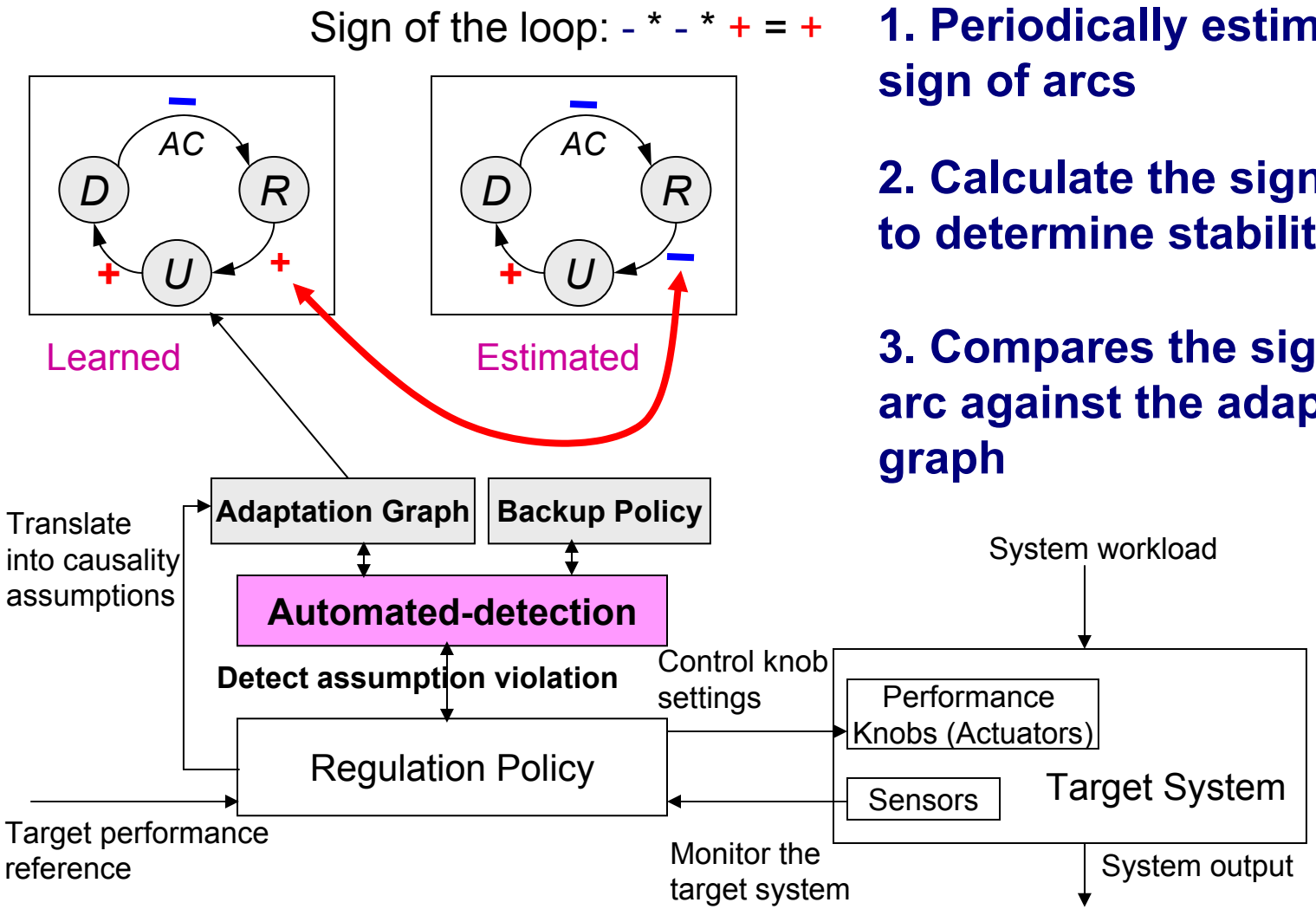
Simple hence fast process: adequate for online detection

No qualitative model: less prone to false alarms due to benign changes in underlying models





2. Detect Instability



1. Periodically estimates the sign of arcs

2. Calculate the sign of cycles to determine stability

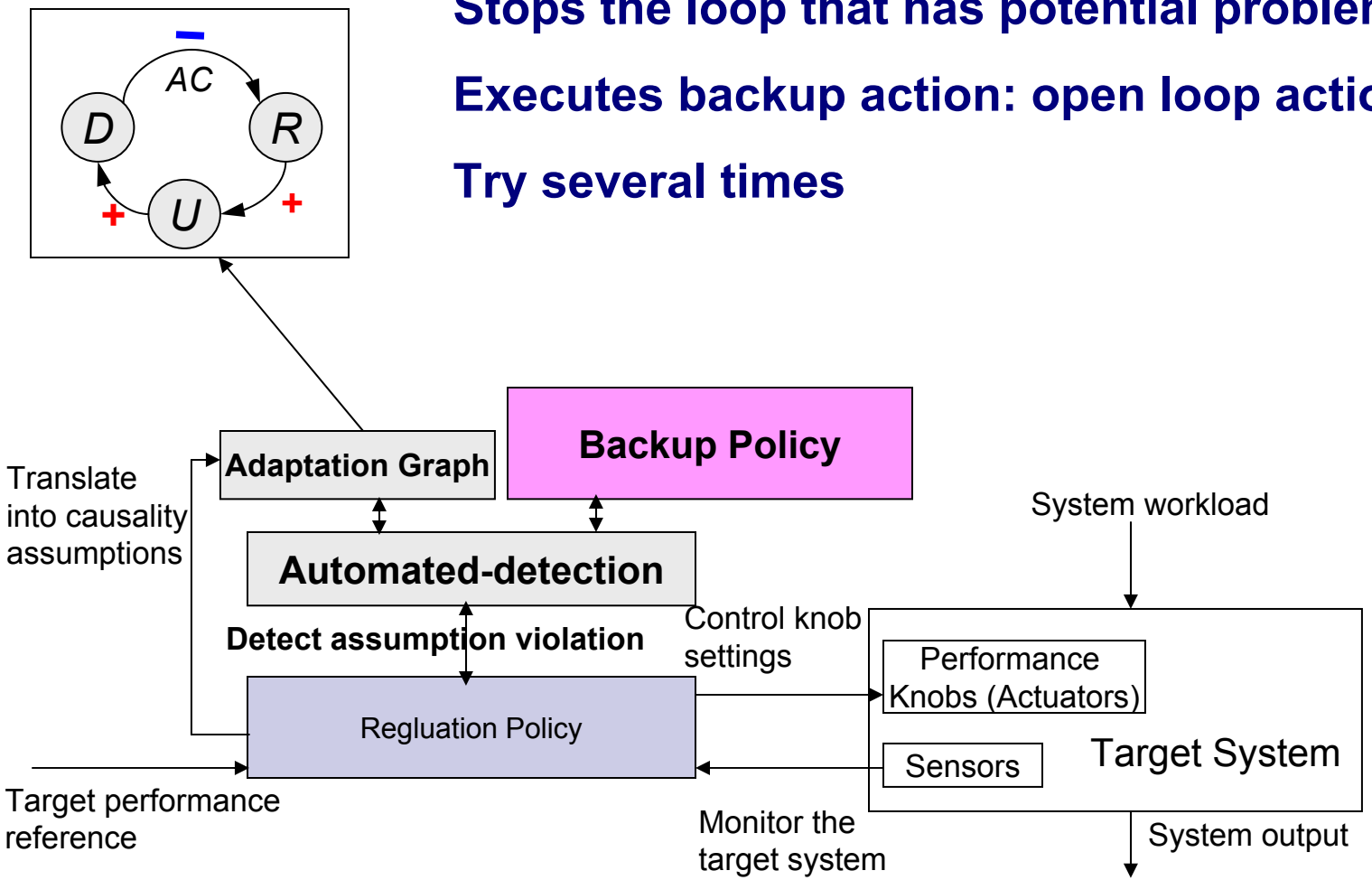
3. Compares the sign of each arc against the adaptation graph

3. Recover from Instability

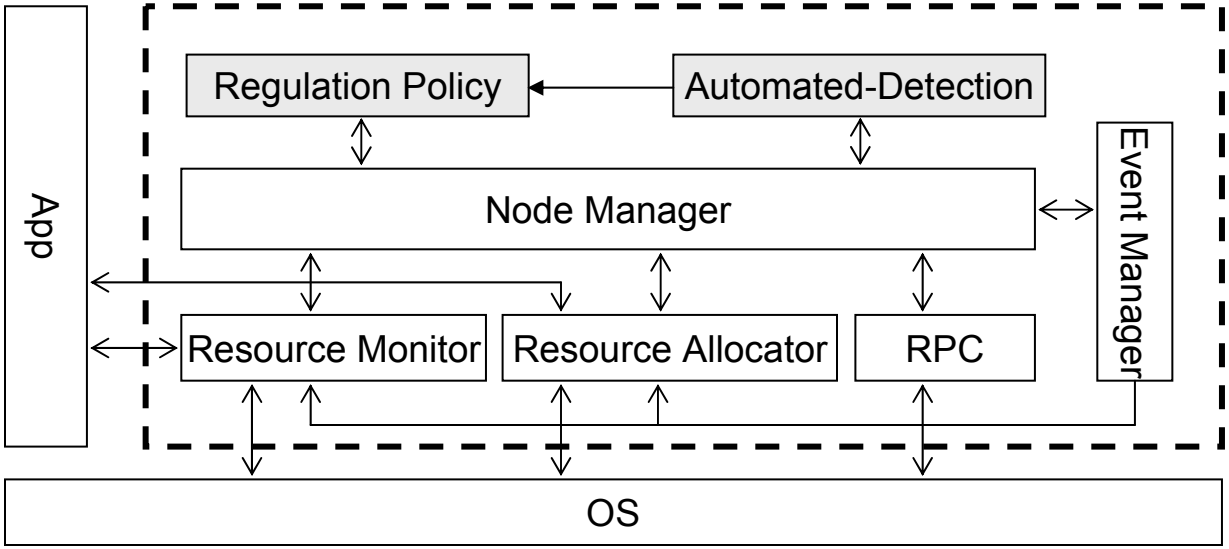
Stops the loop that has potential problems

Executes backup action: open loop action

Try several times



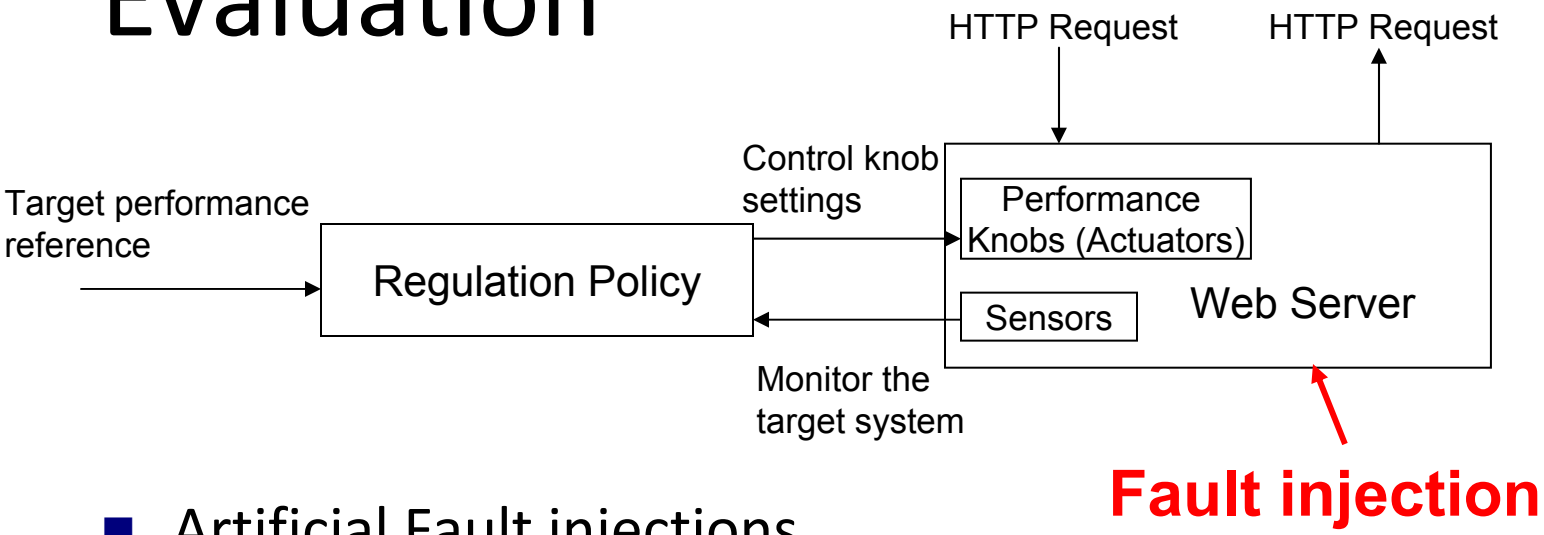
Implementation



- Separate software service
- Simple object-based API interfaces
  - Regulation policy, actuator, sensor object classes
  - Objects as well as API
- Services
  - Node manager, Event manager, RPC
- Written in python

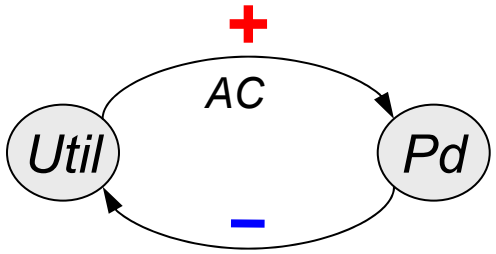
Contents

Evaluation

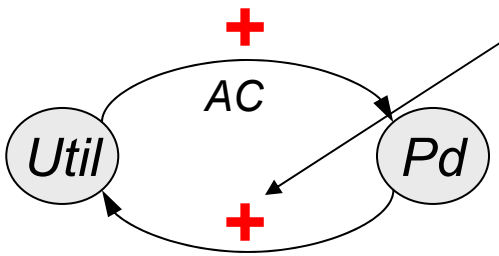


- Artificial Fault injections
  - Two regulation policies in QoS-Adaptive Web server
    - Admission control policy
    - DVS Policy
    - Apache Web server
  - Two faults on QoS-adaptive Web server
    - Missing files: util becomes zero
    - Busy loop: util becomes close to one
  - Combined regulation policies
- Case study

Simple Fault Injections

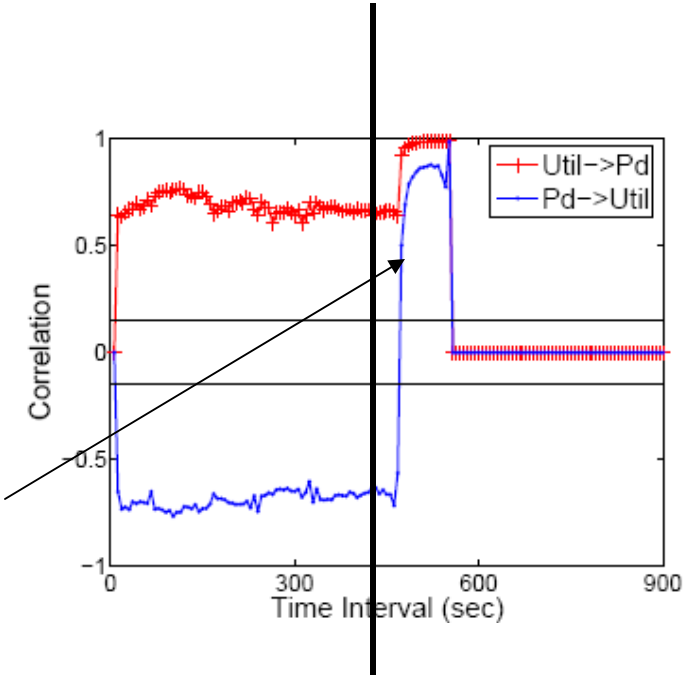


Admission control policy – before fault - **negative**



Admission control policy – **after fault - positive**

Pd→Util is violated



Correlations – busy loop fault

Busy loop fault into a QoS-adaptive Web server with admission control policy

Faults are injected at 450th sec

Pd → Util is violated (- to +), hence positive feedback loop

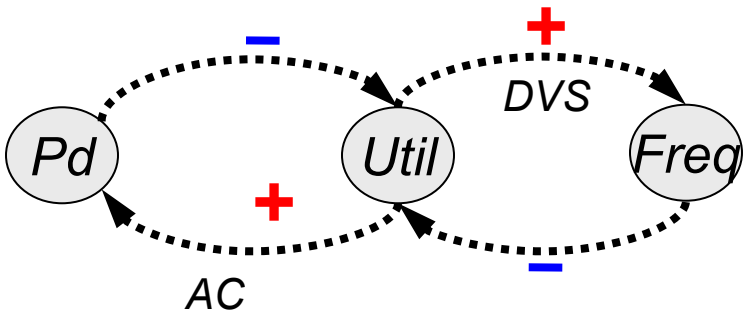
AdaptGuard catches this to detect instability

Fault Injections – Combined policies

DVS policy runs first .

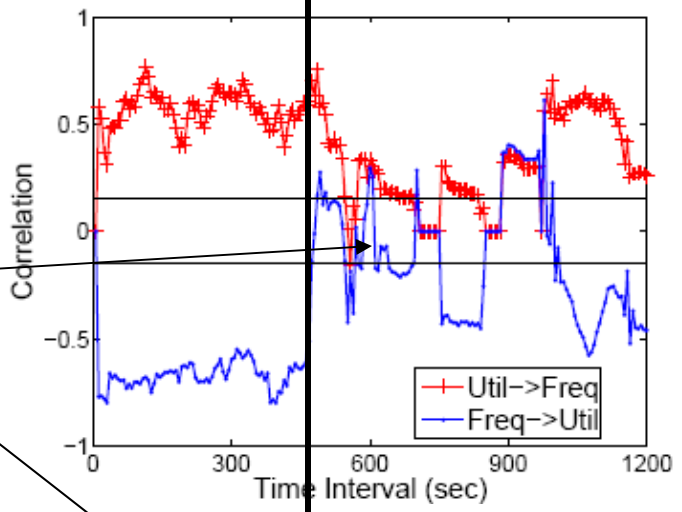
Admission control policy starts to run at 450 sec.

All correlation values become destabilized

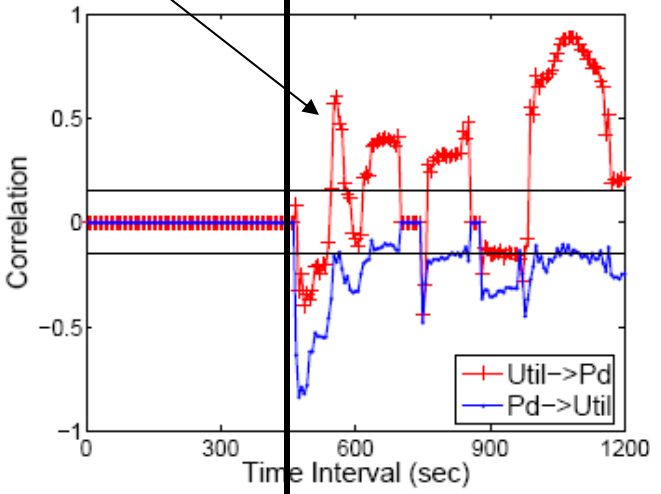


Adaptation graph for combined admission control and DVS policies

an emergent path across the boundaries
→ positive cycle!



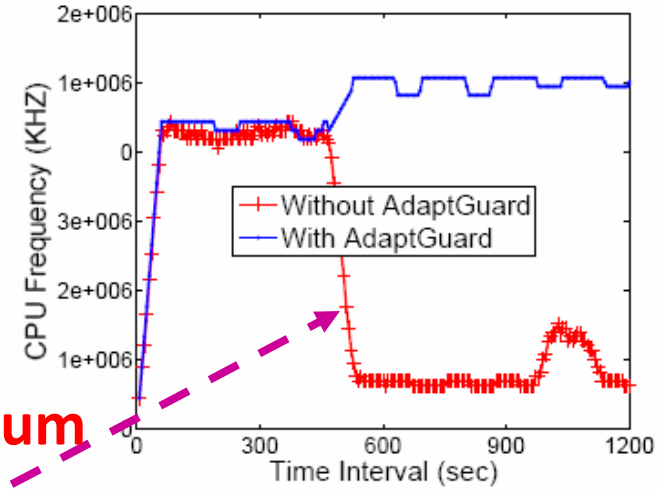
Correlations - DVS Policy



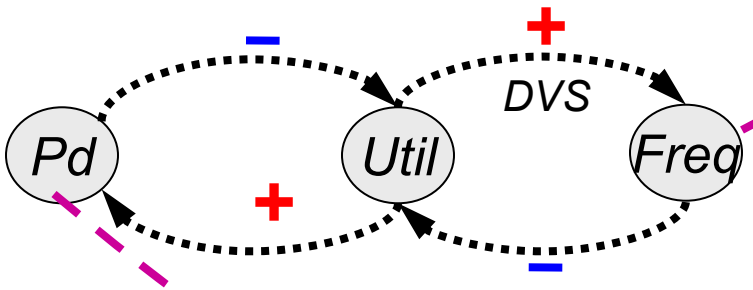
Correlations + admission control Policy

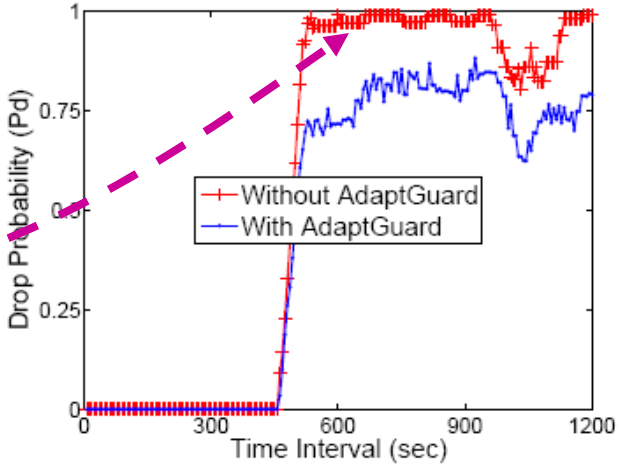
Fault Injections – Combined policies

positive cycle is unstable



Frequency becomes minimum



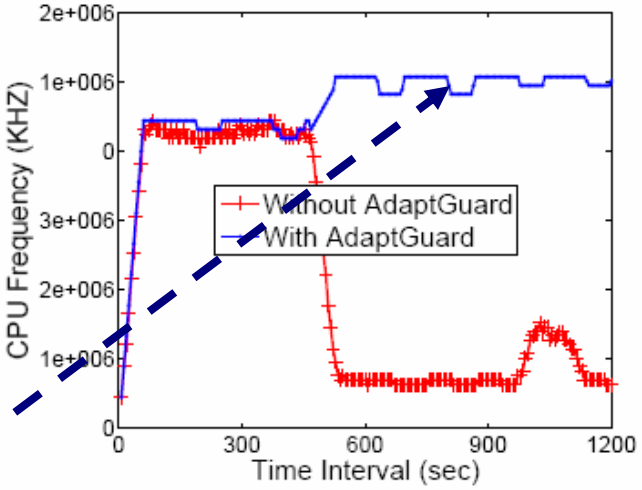


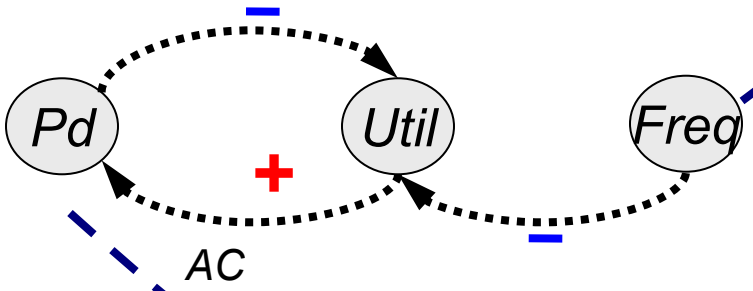
***Pd* becomes maximum →most requests are dropped even though it's not overloaded!**

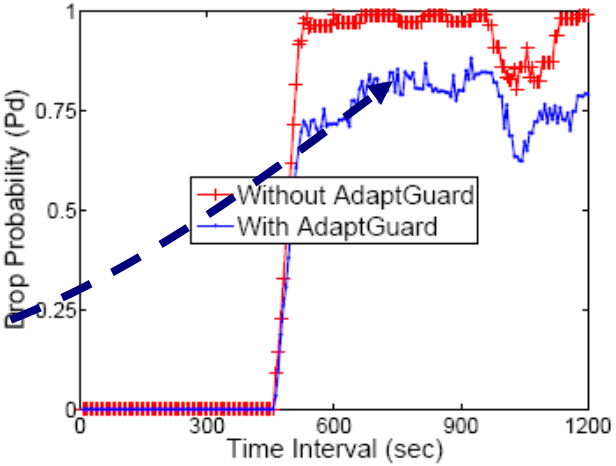
Fault Injections – Cont.

Open the loop and apply backup action

Set the frequency to a fixed value, maximum

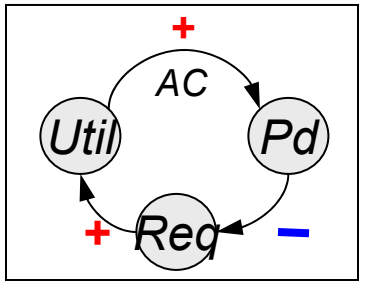




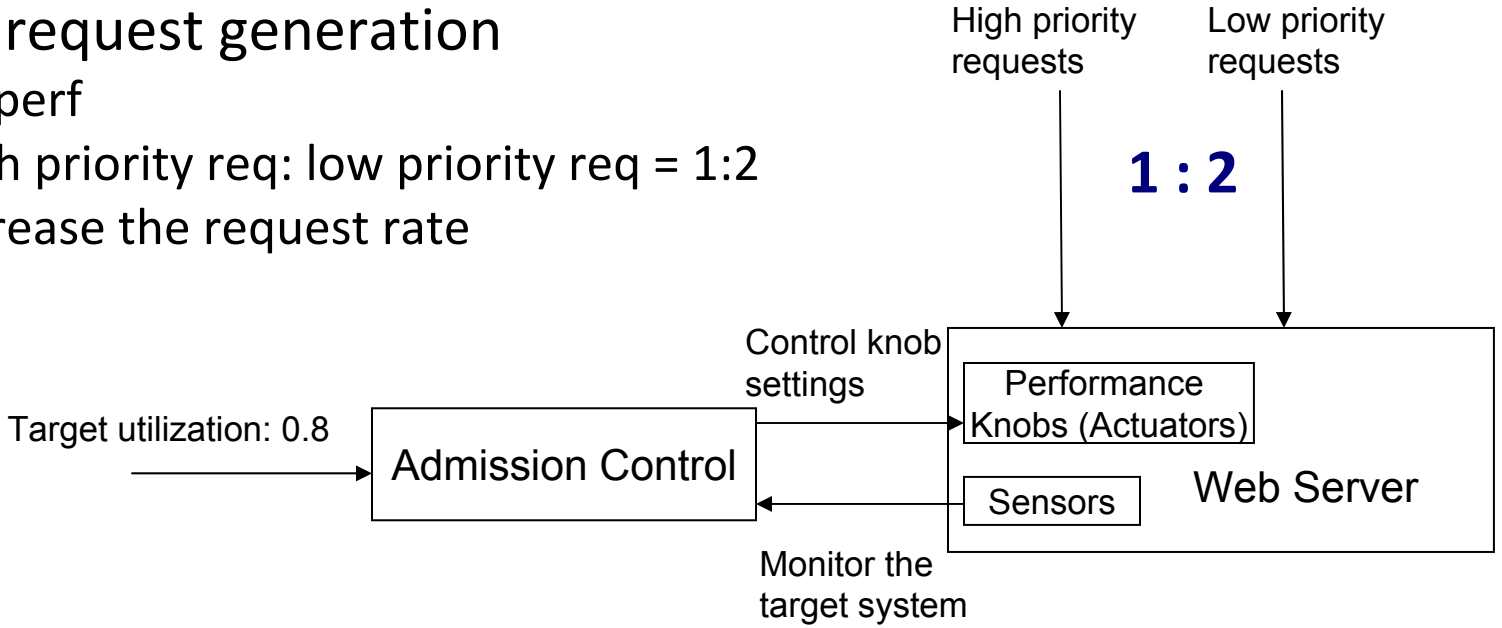


***Pd* becomes smaller → admitting more requests, performance improved**

Case Study



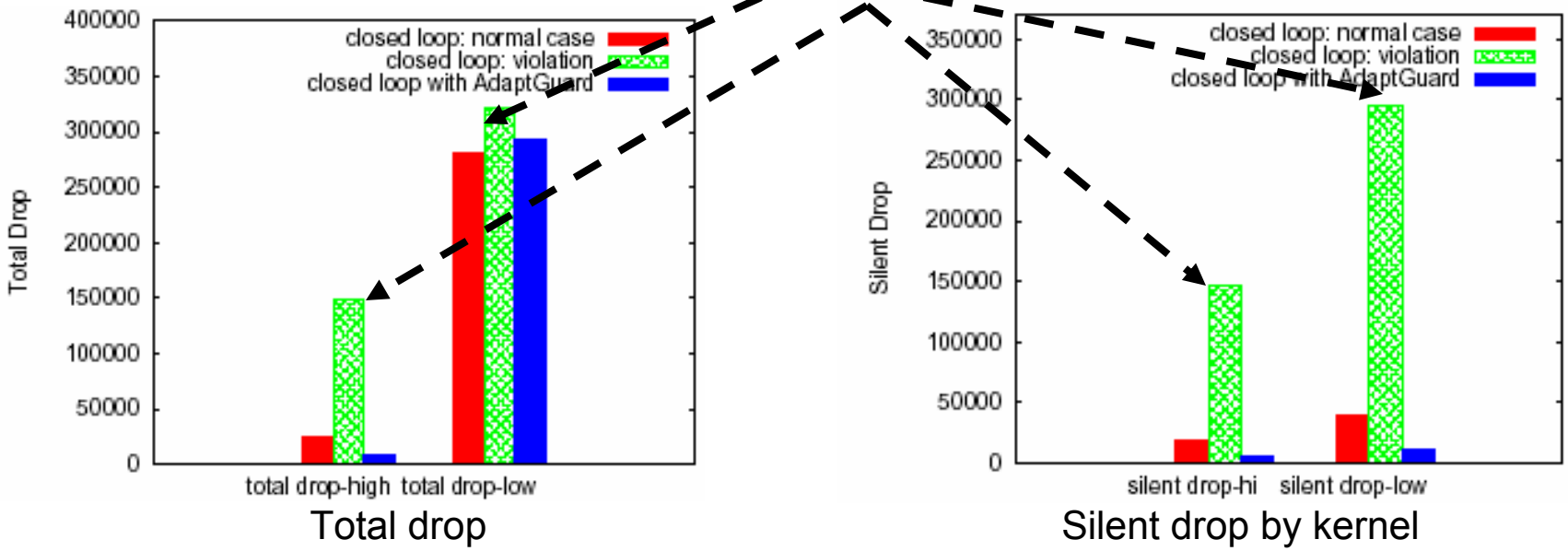
- QoS-adaptive Web server - admission control policy
  - Target CPU utilization = 0.8
  - Multiple priority classes: drops lower priority requests first
- Client request generation
  - Httperf
  - High priority req: low priority req = 1:2
  - Increase the request rate



Results

High and low priority requests dropped indiscriminately

Most of drops come from silent drop



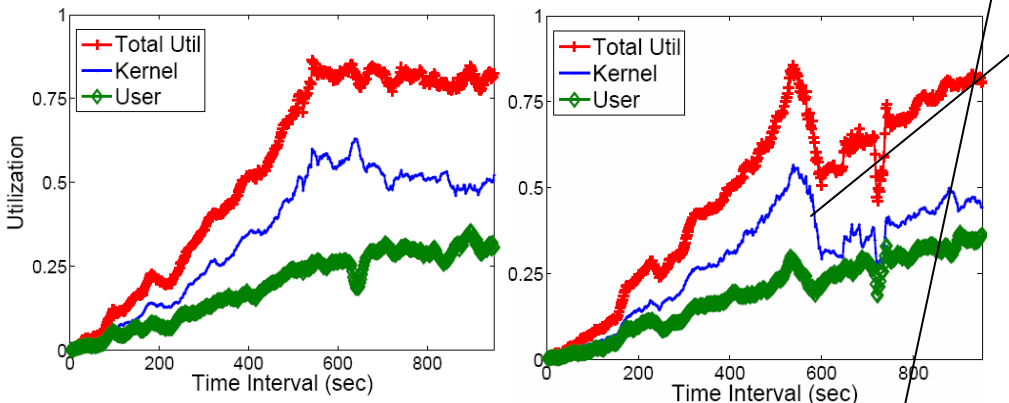
- Compare the number of dropped requests: total drops, drops by kernel
  - Closed loop works well, there is a violation, there is a violation but AdaptGuard activated
- When there is a violation – system destabilized
  - Anti-livelock mechanism – protect system from network overload

Results Cont.

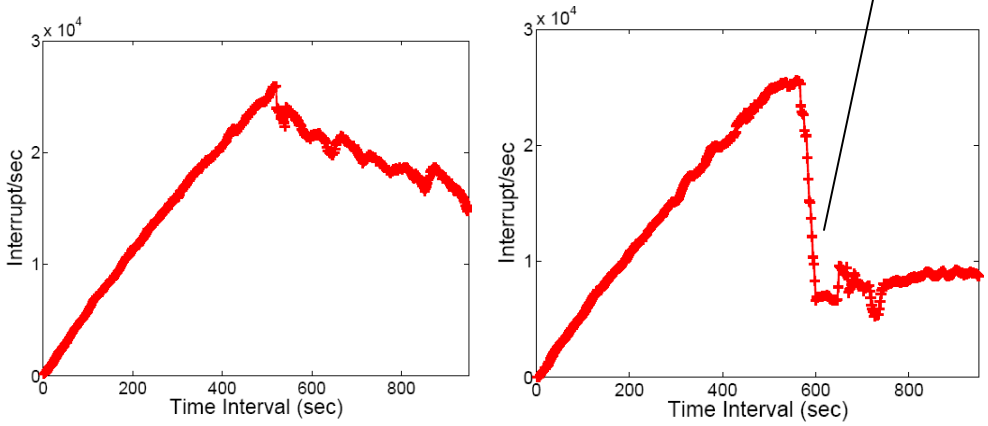
1. Kernel determines network processing is overloaded: switching from interrupt handling to polling

2. Utilization drops due to decrease in the number of interrupts, while the request rate still increases.

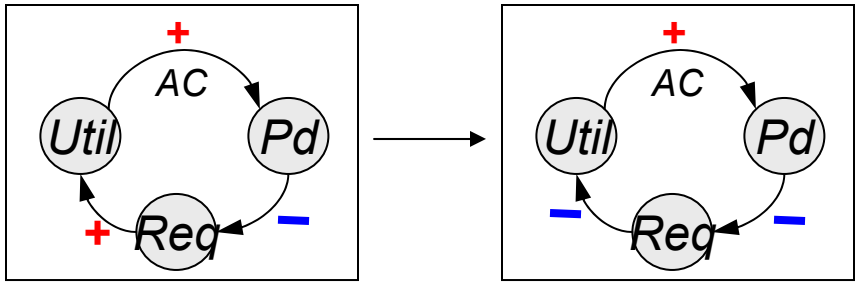
3. Admission control policy tries to accept more requests, while packets is being dropped by the kernel



CPU utilization



# of network interrupts



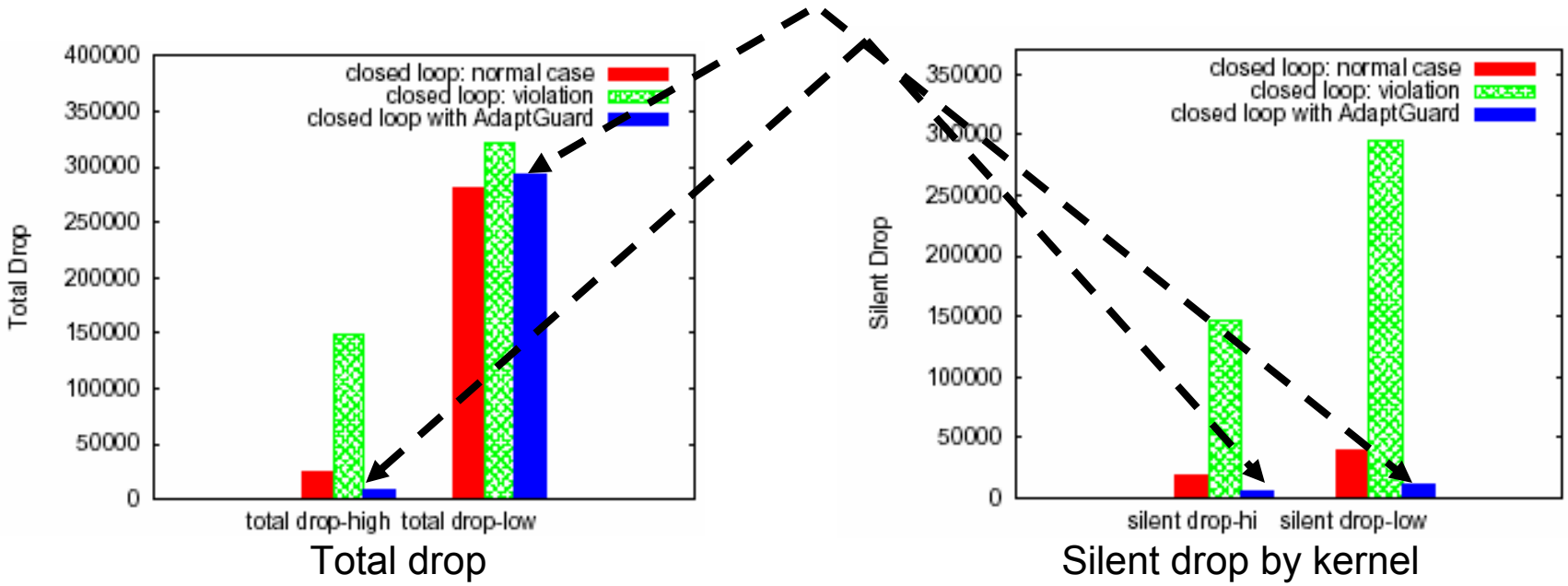
1. Closed loop

2. Closed loop - violation

AdaptGuard detects positive feedback loop and tries to recover from it!

Results

With AdaptGuard: almost no drops at high priority requests and no silent drops at the kernel at both priority requests



- Conflicts between the kernel mechanisms and the admission control policy
- Recovery
  - Stops the admission control policy
  - Runs the back-up controller
    - drops 60% of incoming requests

Conclusion

- Presented AdaptGuard, a software service that guards the target adaptive system from instability
- Simple online detection mechanism specialized in closed loop behavior, independent of applications
- Recovery mechanism
- Evaluated through artificial fault injections and a case-study in a QoS-adaptive Web server

Questions