YORK UNIVERSITÉ UNIVERSITY

ASRL
Adaptive Systems Research Lab

PID Controllers

Marin Litoiu
Department of Electrical Engineering and Computer Science
York University
mlitoiu@yorku.ca
http://www.ceraslabs.com

Engineering

- *"In science, if you know what you are doing, you should not be doing it. In engineering, if you do not know what you are doing, you should not be doing it. Of course, you seldom, if ever, see either pure state."*
- —Richard Hamming, *The Art of Doing Science and Engineering*

Three Performance Metrics for an Application

- **Performance Measurement**
  - Response time
  - Throughput
  - Utilization

Software Performance

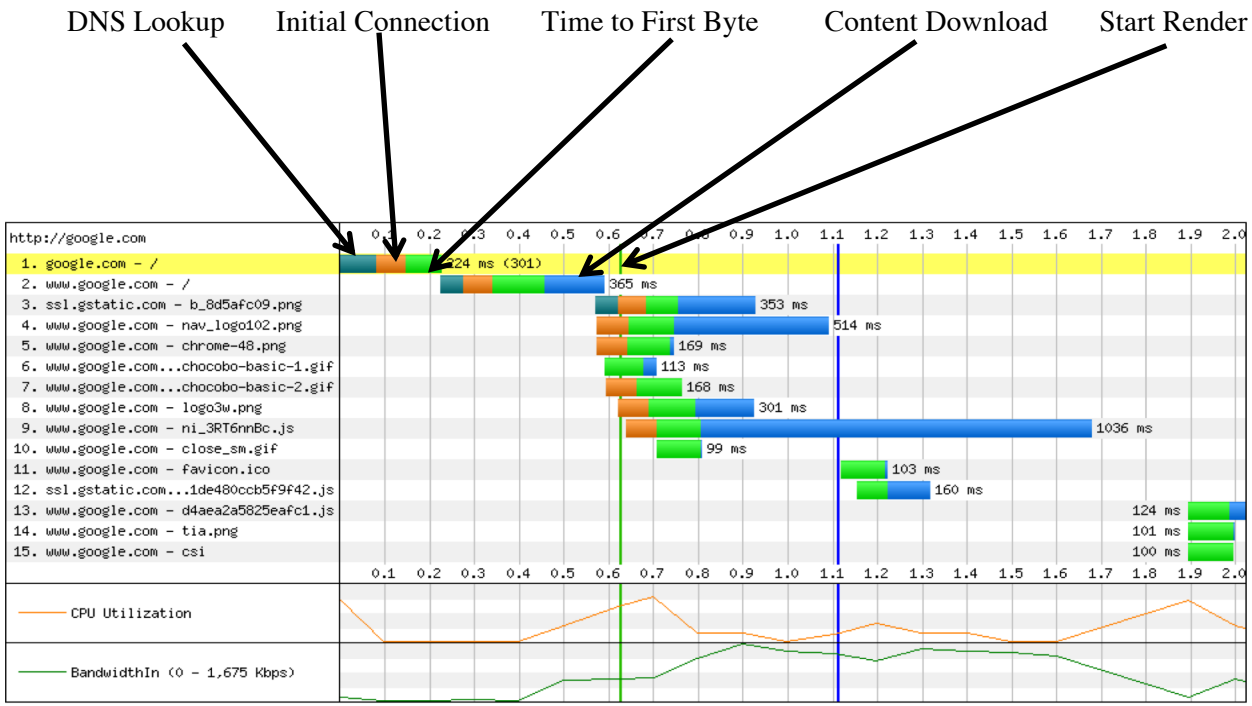
- **Performance = how fast the software performs its functions**
- **In common language, we say software is "fast," "slow"**
- **Among all software qualities (non-functional requirements), performance can be expressed quantitatively**
- **In on-line (web) and e-business applications**
  - From user perspective, performance means "response time"
  - From software owner perspective, performance means "throughput," how many transactions (requests) per second the software can process
    - Has implications on revenue(proportional with number of transactions- think Expedia, or number of users-think Facebook)

Response time

- **Response time: the difference between the time the user submits a request and the time the user gets a response**
- **It has many components**
  - Client time: rendering time in the browser
  - Network time: time to travel between client and server(s)
  - System time: time spent on server to process the request and generate the response
  - Each of the above can be detailed further
- **To see some of the components for any web page, use** www.webpagetest.org
  - Choose the location of the client
  - Choose the browser
  - Specify the URL of the page you want to test

Response time components (for google.com)

www.webpagetest.org



Utilization

- **Utilization: the percentage of time a device (CPU, disk, network) is busy**
- **Ex: If out of 60 sec, the CPU is processing data for 30 sec, the CPU Utilization =(30/60)*100%=50%**
- **Utilization takes values between 0% and 100% (sometimes is expressed as a fraction, with values between 0 and 1)**
  - U=50/60=0.5;
- **When a device has 100% utilization, then it is saturated**
  - A device saturated is also called a *bottleneck*
  - It is not good to have devices saturated
    - The response time will be slow, the system might crash, etc…
- **When utilization is high, response time is high**
- **One way to decrease utilization is to add more resources to the cluster**

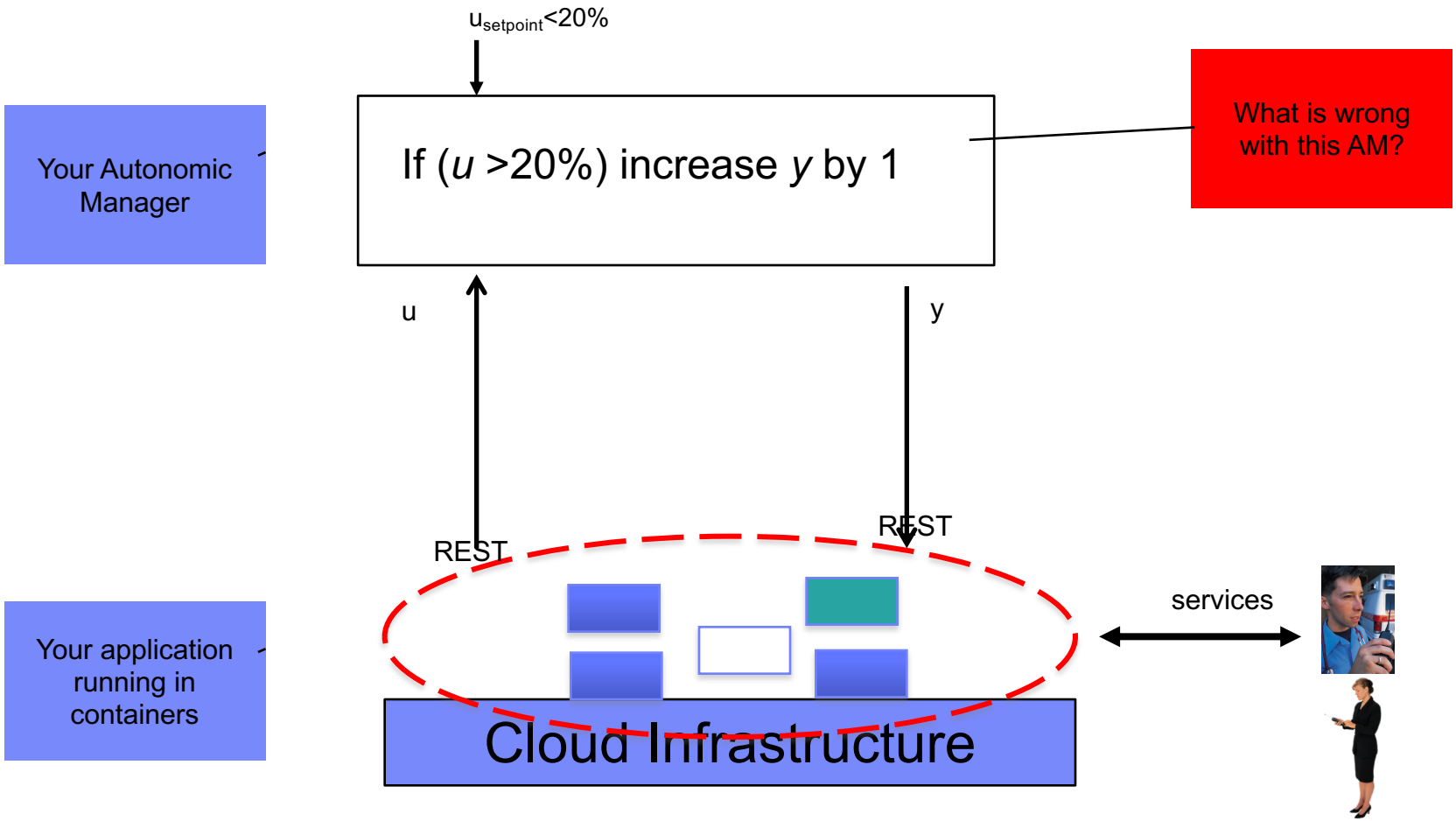
Throughput

- **Measures how many transactions (requests) per time unit the software can process**
  - Requests per minute
  - Requests per hour, etc..
- **Is different than arrival rates (how many requests arrive)**
- **Throughput <= Arrival rate**
- **In a well designed and managed system arrival rate=throughput**
- **What happens with requests that arrive and are not processed**
  - Are queued or rejected

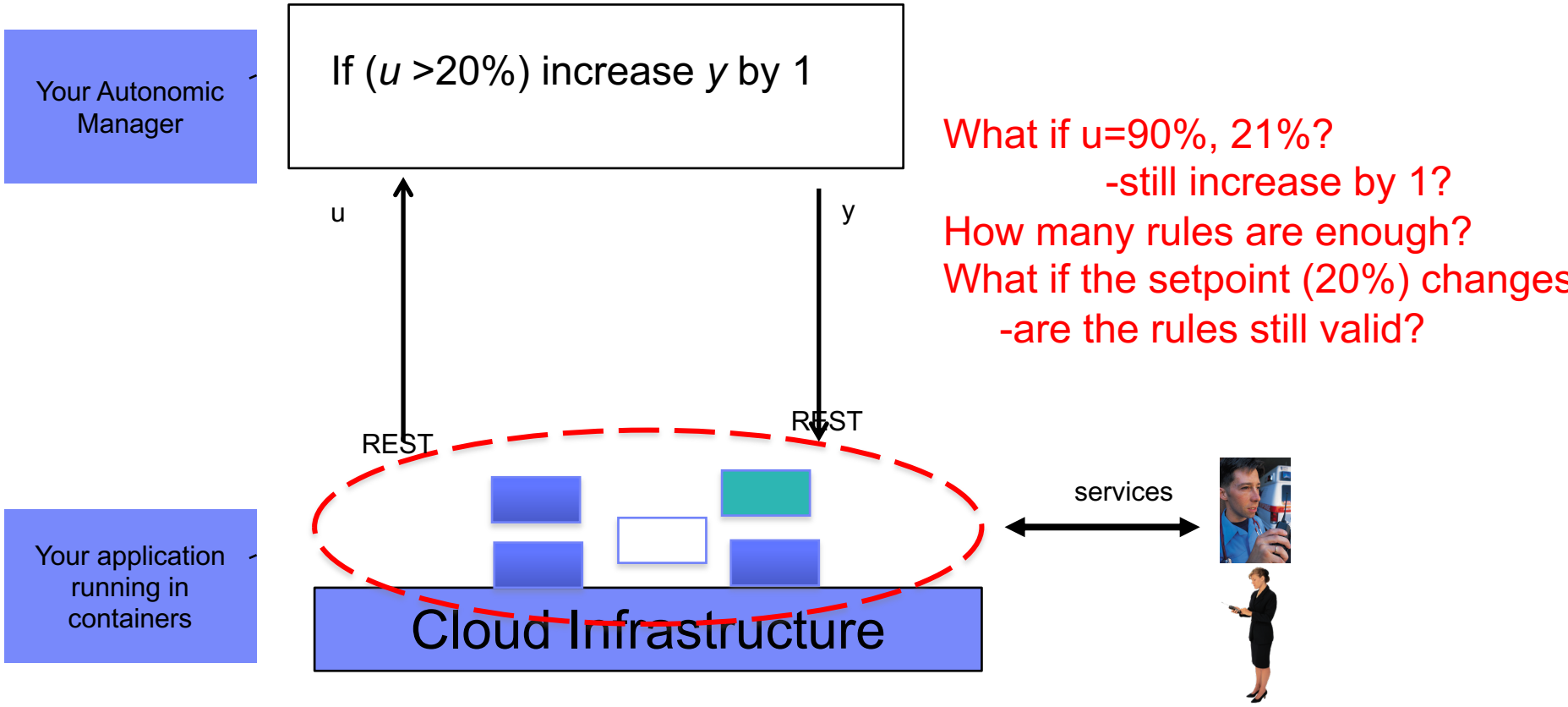
Your turn (10 min)

- **Compare the response time of google.com with that of yorku.ca**
  - Where is the difference coming from?
  - Does the browser type make a difference?
  - How would you improve the response time of yorku.ca
- **Work in teams of two**

Adaptive Applications on Clouds



Adaptive Applications on Clouds



What if u=90%, 21%?
-still increase by 1?
How many rules are enough?
What if the setpoint (20%) changes
-are the rules still valid?

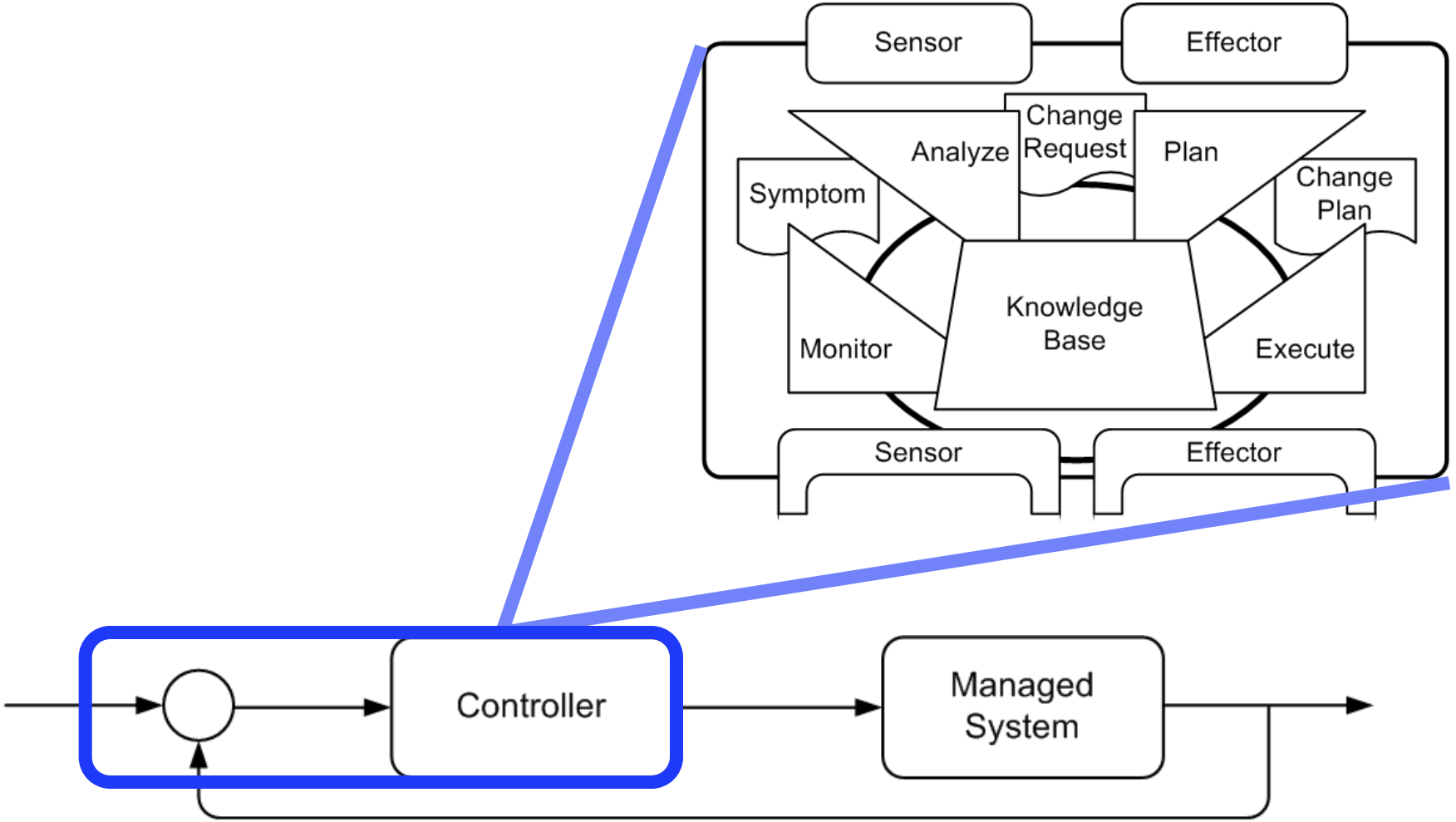
Control Theory

- **A theory that deals with influencing the behavior of dynamical systems**
- **An interdisciplinary subfield of science, which originated in engineering and mathematics**

Control System Goals

- **Regulation**
  - Thermostat, target service levels
- **Tracking**
  - Robot movement
  - Adjust TCP window to network bandwidth
- **Optimization**
  - Best mix of chemicals
  - Minimize response times
  - Optimize the cost and perfomance

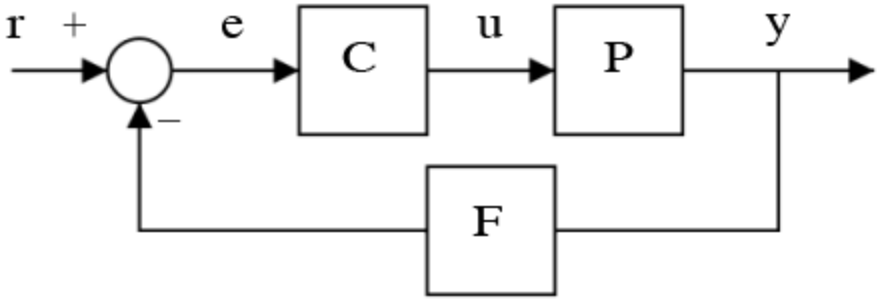
Controller as an Autonomic Element



Closed Loop Controller or Feedback Controller

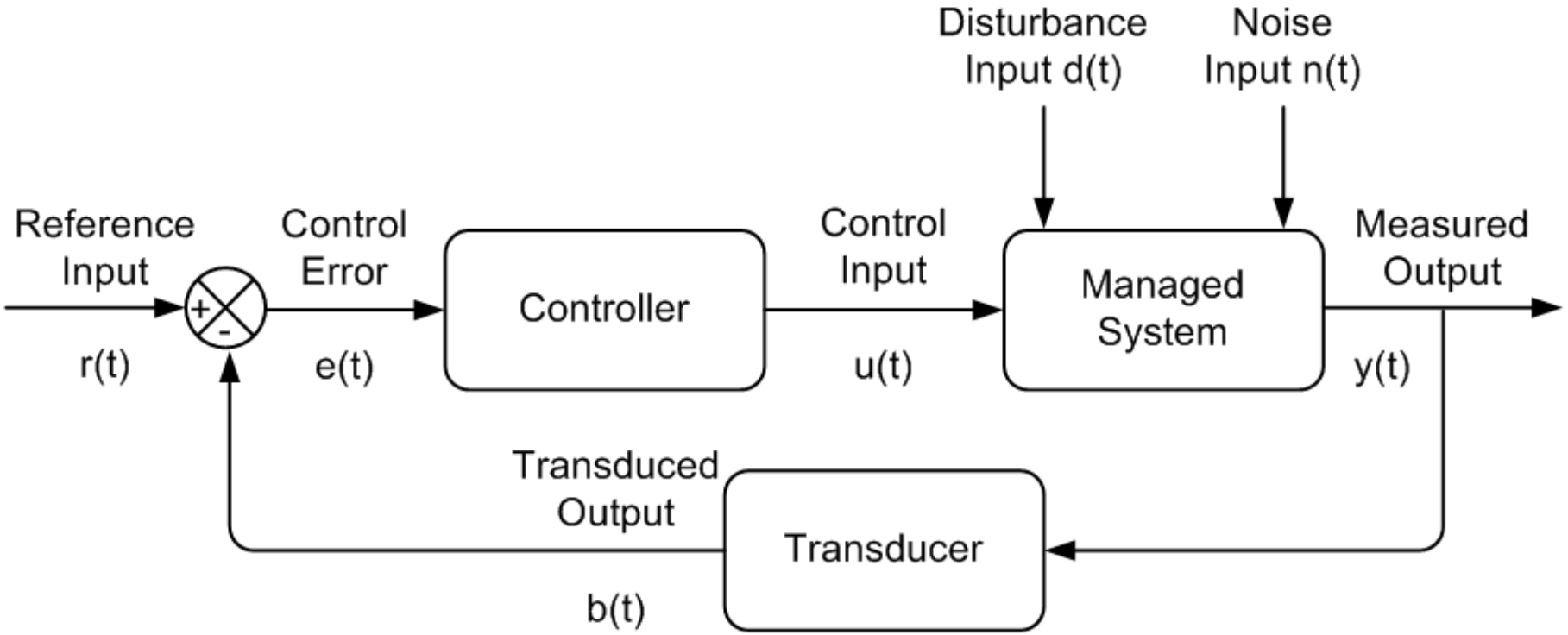
- **The output *y(t)* of the feedback system is fed back through a sensor measurement *F* to the reference value *r(t)*.**
- **The controller *C* then takes the error *e* (difference) between the reference and the output to change the inputs *u* to the control process *P*.**

- **SISO**
  - Single-input-single-output (*SISO*) control system
  - Variables are simple scalar values (i.e., r(t), e(t), u(t), y(t)
- **MIMO**
  - Multi-Input-Multi-Output systems, with more than one input/output, are common
  - Variables are vectors



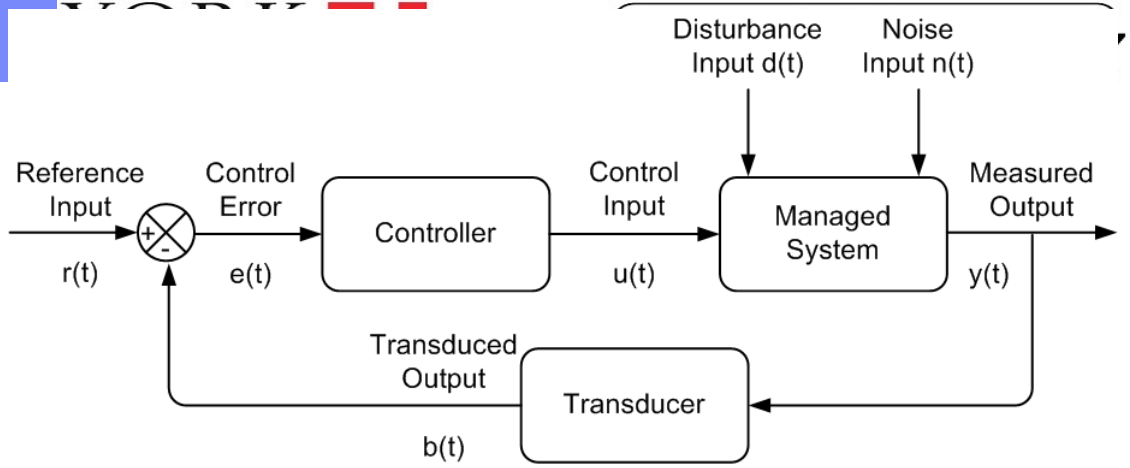
Realization of a Dynamic Architecture

- **Feedback control system with disturbance**



Hellerstein, Diao, Parekh, Tilbury: Feedback Control of Computing Systems. John Wiley & Sons (2004)

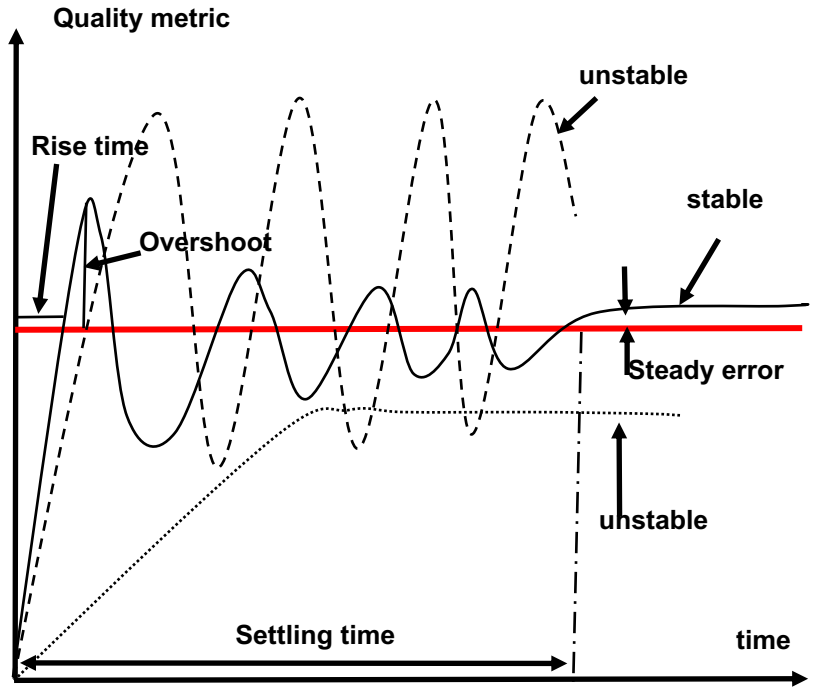
Realization of a Dynamic Architecture



- **Reference input**
  - Goal, objectives, specified desired output
- **Control Error**
  - Reference input minus transduced output
- **Control Input**
  - Parameters which affect behavior of the system—number of threads, CPU, memory
- **Disturbance input**
  - Affects control input—arrival rate

- Controller
  - Change control input to achieve reference input—design is based on a model of the managed system
- Managed system
  - Dynamical system, process, plant—often characterized by differential equations
- Measured output
  - Measurable feature of the system—response time
- Noise input
  - Affects measured output
- Transducer
  - Transforms measured output to compare with reference

Quality of Control Metrics



- short rise time
- short settling time
- small overshoot
- no oscillations
- stability
- tolerance to perturbations
- Red: the reference(setpoint) input, e.g throughput, can be an interval
- Black: the measured output (e.g. throughput)

PID Controller

- **The PID algorithm is the most popular feedback controller algorithm used**
- **It is a robust easily understood algorithm that can provide excellent control performance despite the varied dynamic characteristics of processes**
- **PID algorithm consists of three basic modes:**
  - **P**roportional mode
  - **I**ntegral mode
  - **D**erivative mode

P, PI, or PID Controller

- **When utilizing the PID algorithm, it is necessary to decide which modes are to be used (P, I or D) and then specify the parameters (or settings) for each mode used.**
- **Generally, only three basic algorithms are used: P, PI or PID**

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

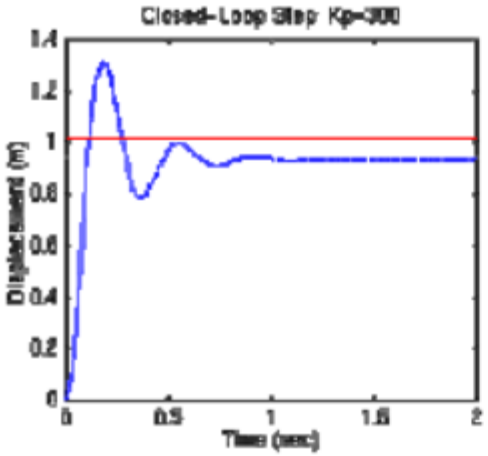
proportional gain integral gain derivative gain

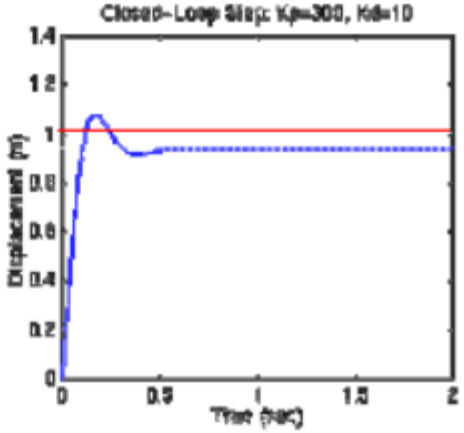
Controller Effects

- **A proportional controller (P) reduces error responses to disturbances, but still allows a steady-state error.**
- **When the controller includes a term proportional to the integral of the error (I), then the steady state error to a constant input is eliminated, although typically at the cost of deterioration in the dynamic response.**
- **A derivative control typically makes the system better damped and more stable**

PID Controller

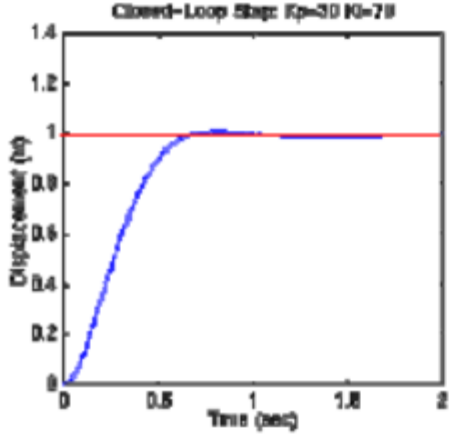
P

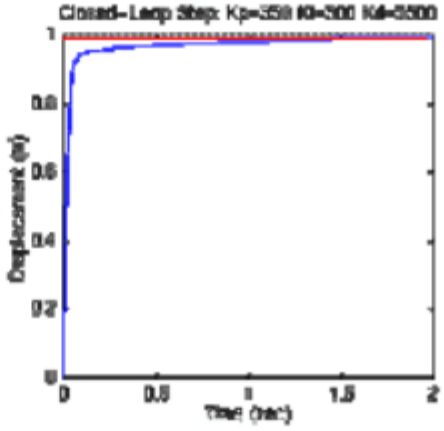




PD

PI





PID

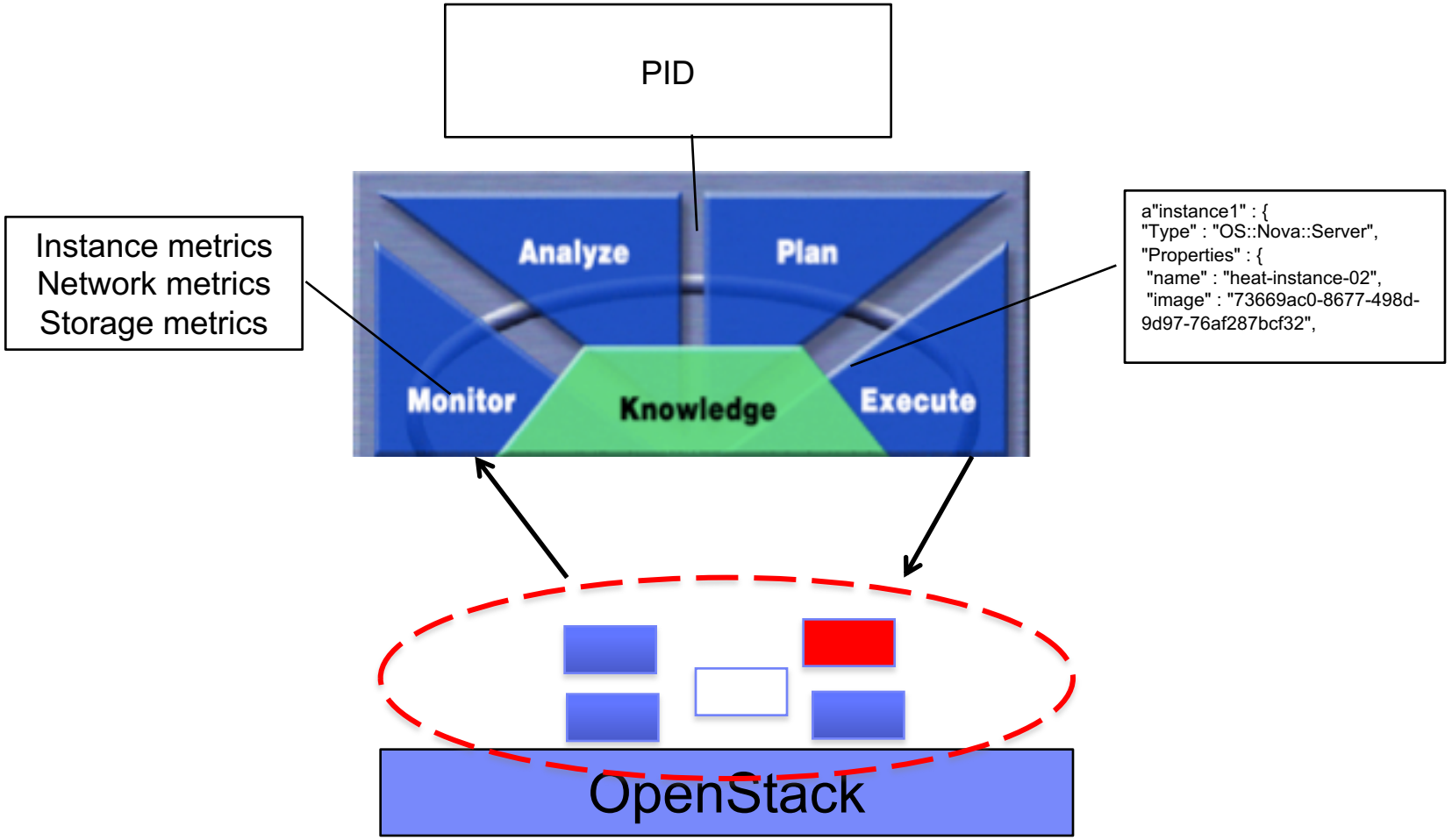
Closed-Loop Response

	Rise time	Max overshoot	Settling time	Steady-state error
P	Decrease	Increase	Small change	Decrease
I	Decrease	Increase	Increase	Eliminate
D	Small change	Decrease	Decrease	Small change

PID Controller

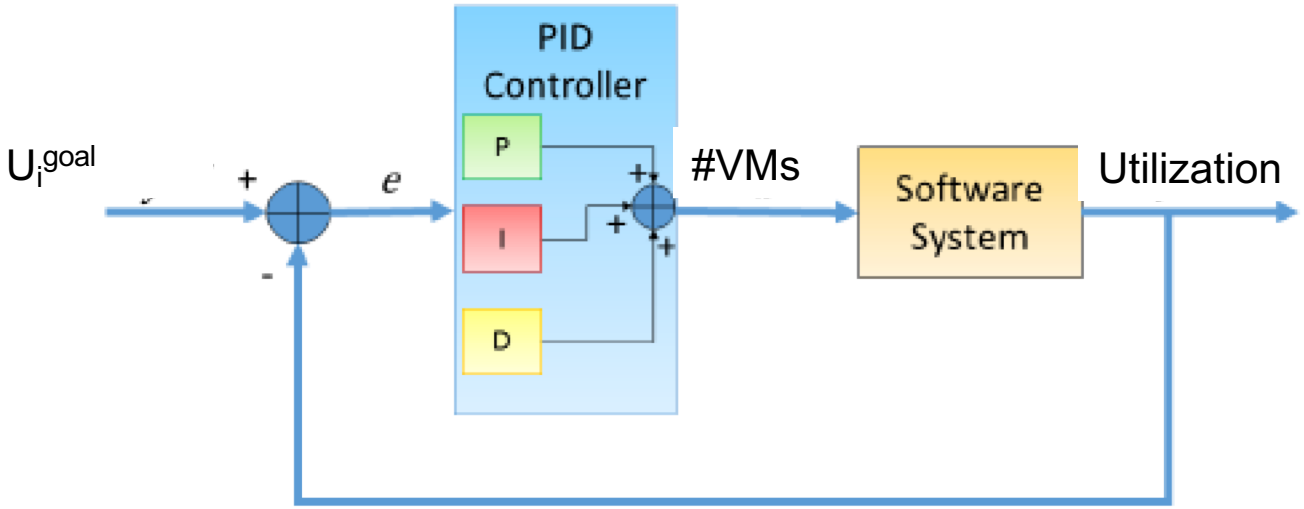
- **Output feedback**
  - From Proportional action
  - Compare output with set-point
- **Eliminate steady-state offset or error**
  - From Integral action
  - Apply constant control even when error is zero
  - Eliminates transients; sum of all previous errors
- **Anticipation**
  - From Derivative action
  - React to rapid rate of change before errors grows too big
  - Anticipate the trends; rate of change of the error

Case Study*

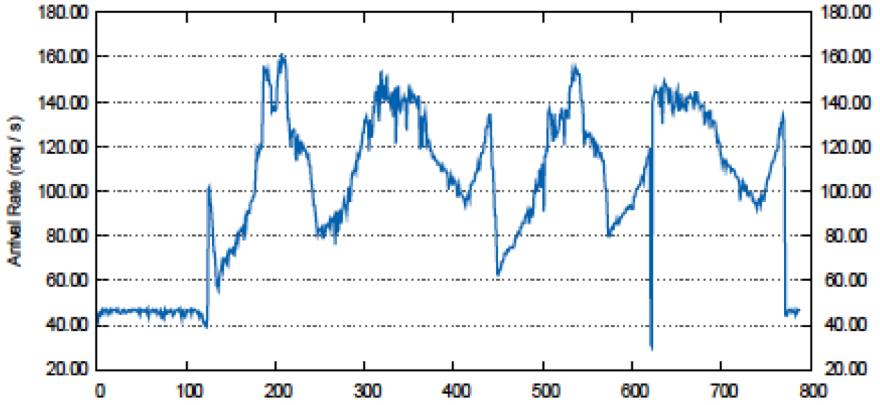


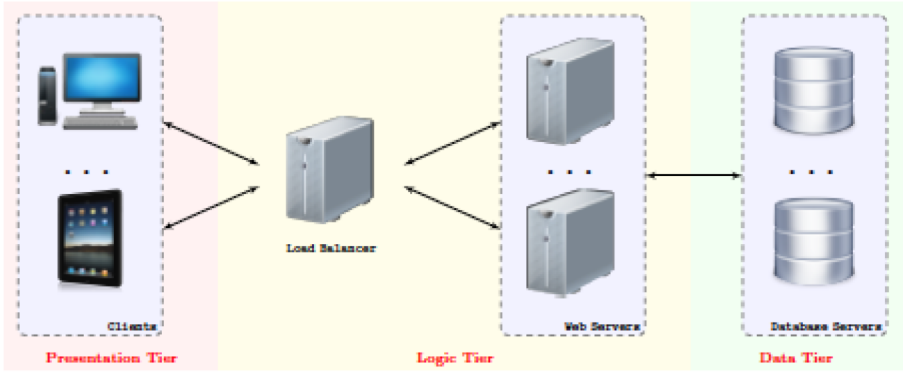
C. Barna, M. Fokaefs, M. Litoiu, M. Shtern and J. Wigglesworth, "Cloud Adaptation with Control Theory in Industrial Clouds," *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, Berlin, 2016, pp. 231-238.

PID Controller for Utilization



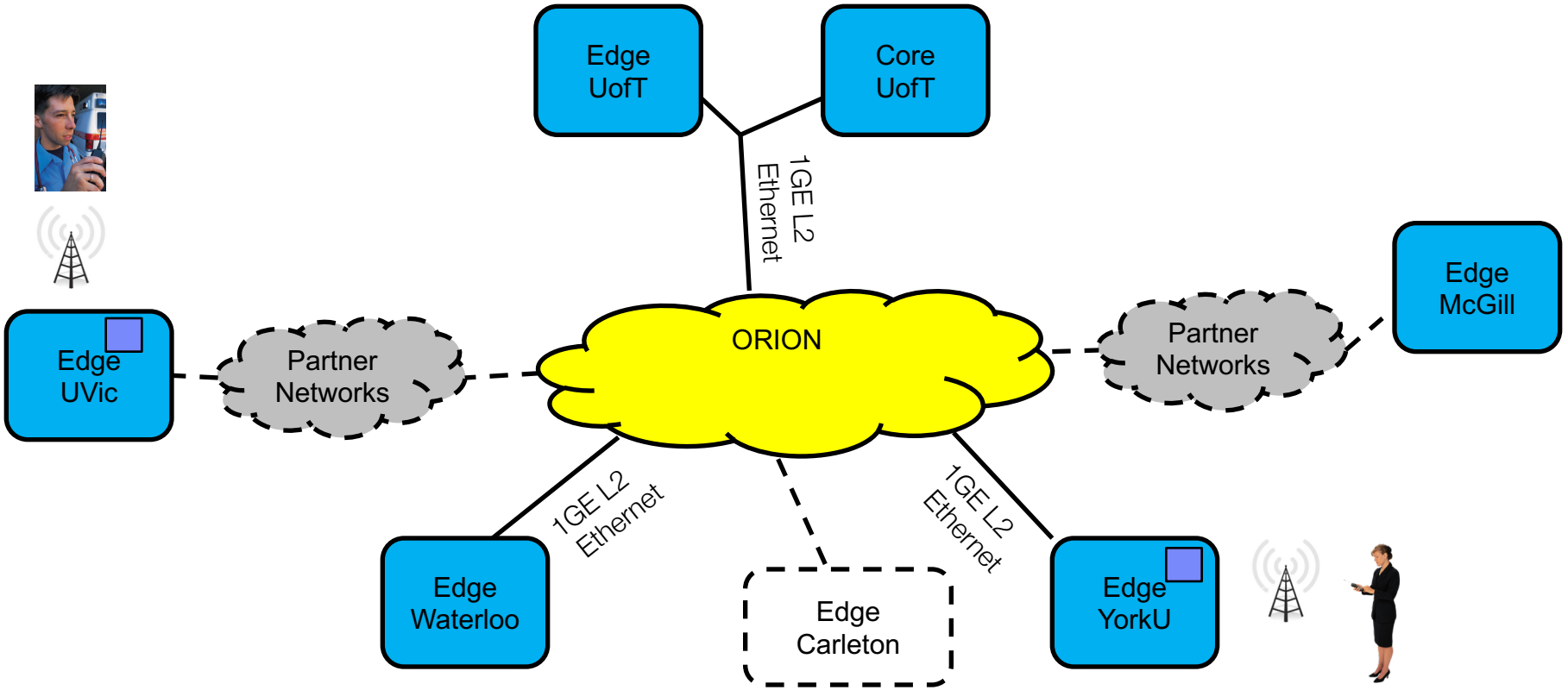
Experimental Settings





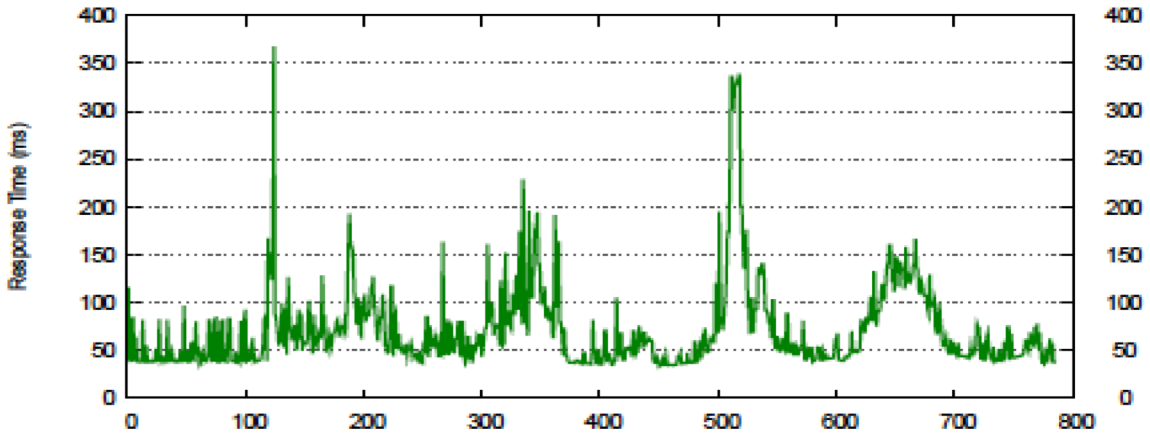
- Are PID controllers tunable with no performance model at hand?
- Are PID controllers efficient when the control actions are limited?
- Are the PID and tuning portable across clouds?

SAVI Cloud (as 2016)
http://savinetwork.ca

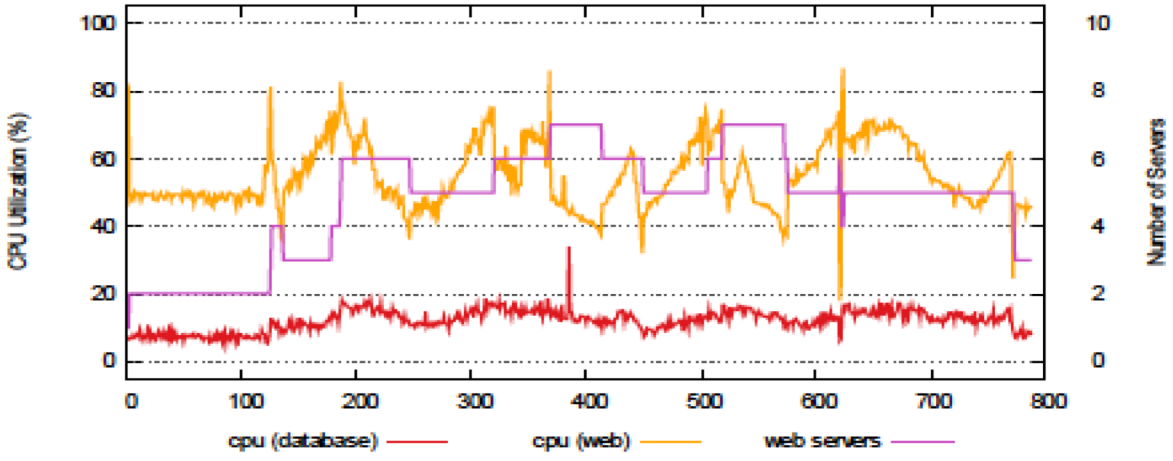


In SAVI, the cloud is at your fingertip…

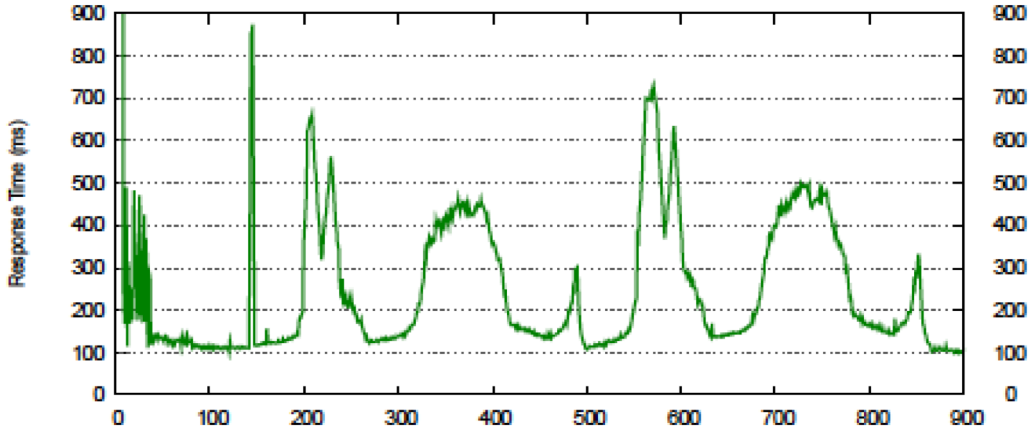
PID on Savi Cloud



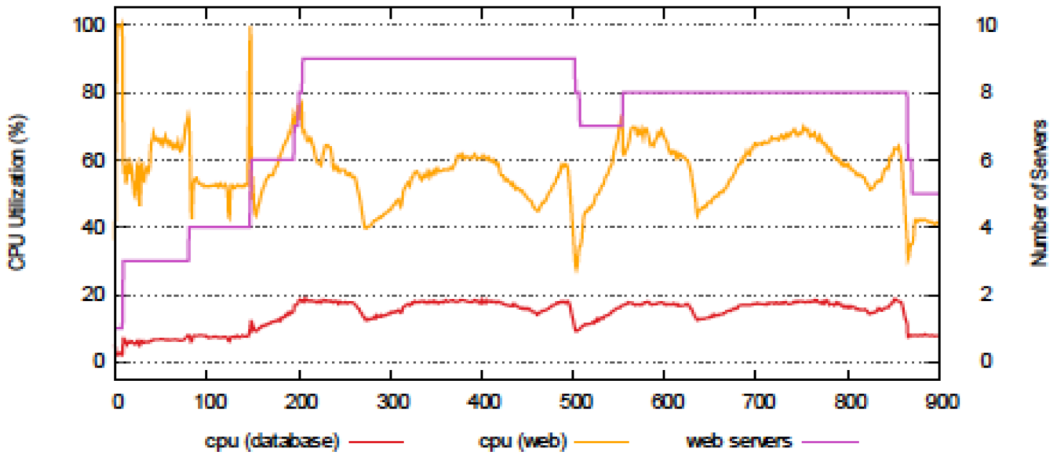
(b) Response time



PID on Amazon EC2



(b) Response time



(c) CPU utilization and number of web servers.

Issues: sampling and discrete time

- **Need to work in discrete time:**
- **Need to decide on the sampling interval, T**
  - Use mean values for response time, throughput, utilization
    - For example, if 1 min is the sampling interval, collect enough metrics to be able to calculate the mean values
- **u(t)→u(k); k is the discrete time: k+1=k+T**
- **Derivative: (e(k)-e(k-1))/T**
- **Integral->sum**

Issues: Tuning Kp, Ki, Kd

Ziegler–Nichols method

Control Type	K_p	K_i	K_d
P	$0.50K_u$	-	-
PI	$0.45K_u$	$0.54K_u/T_u$	-
PID	$0.60K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

- **Ki=Kd=0;**
- **Increase Kp until the system start to oscilate: Ku is the max overshoot, Tu is the period of oscillations;**
- **Calculate Ki and Kd with the formulas above**

PID Manual Tuning

- **Ki=Kd=0;**
- **Increase Kp until the system starts to oscillate**
  - Kp=Kp/2
- **Increase Ki until you remove the steady error**
- **Increase Kd until the loop is fast enough**

Issues

- **Output is calculated as a real number value**
- **In general, in computing, we use integer control values (no of VMs, no of threads, timeouts, etc..)**
- **Need to round the values: up, down?**

Conclusions

- **PID controllers are efficient and practical**
  - Simple and easy implementation
  - Easy tuning and portable tuning
  - D component is not that effective, better disable Kd=0;
    - Prediction on short term not accurate
- **Limitations and further work**
  - Not easy to apply to MIMO cases
  - Further work: cascaded control, adaptive, MIMO