Towards a Passivity Framework for Power Control and Response Time Management in Cloud Computing

M.D. Lemmon
Dept. of Electrical Engineering, University of Notre Dame
lemmon@nd.edu

ABSTRACT

There has been great interest in using classical control theory to manage computing systems. Classical control, however, focuses on regulating a system's state in a neighborhood of an equilibrium point and it is unclear if such equilibrium based methods are well-suited for systems providing performance guarantees in the face of large and rapid input fluctuations. This may be the case for cloud computing applications where consumer workloads vary in a rapid and unpredictable manner. As an alternative to classical methods, this paper discusses a passivity framework for power control and response-time management in cloud computing applications. This paper suggests that passivity concepts provide a decentralized method for certifying whether a collection of interconnected cloud computing systems can coordinate their actions in a stable manner.

Categories and Subject Descriptors

I.2.8 [**Control Theory**]:

Keywords

Control Theory, Passivity

1. INTRODUCTION

Control theoretic methods have been used to manage a wide range of computational systems that include real-time embedded systems [LWK05, WJLK07], computer storage systems [KKZ05], web servers [DGH$^+$02], and virtualized data centers [XZSW06, RRT$^+$08].Much of this prior work has made extensive use of *classical control methods*. Classical control is well suited for designing robust controllers of linear systems. For nonlinear systems, classical methods only provide local guarantees on system stability and performance. It is unclear if such local guarantees are appropriate for computational systems that must respond to large and rapid variations in their input streams.

Prior surveys have already identified some of the issues encountered in using classical methods for computational resource management [ADH$^+$08, ZUW$^+$09]. Some of these issues may be recast into the following list

- **Global Performance:** Distributed computing applications must provide service guarantees in the face of "large" variations in user demand. Since classical methods presume regulation within a local neighborhood of the operating point, they can only provide limited guarantees of computational system performance.
- **Nonlinear Dynamics:** Classical methods presume the use of linearized dynamics in which the system state takes values over the whole real line. Computational system states, however, are usually *positive* and *quantized*. This restriction introduces nonlinear dynamics that may be difficult to linearize.
- **Model Uncertainty:** Classical methods require a formal model of the plant dynamics with bounds on the modeling uncertainty. Such models of computational systems may not be known beforehand.
- **Evolution versus Engineering:** Classical control was developed for *engineered* systems; i.e. systems whose development proceeds in a top-down manner. Computational systems, however, are often developed in a bottom-up manner as software upgrades and new components are incrementally added into an existing system. For such *ad hoc* systems, the use of classical methods may lead to conservative controllers that trade away performance for safety.

Rather than using classical methods, this paper examines the use of nonlinear passivity concepts [vdS00] that are well suited in controlling distributed systems.

Passivity is an alternative to more commonly used *stability concepts* such as asymptotic stability or bounded-input bounded-output (BIBO) stability. Informally, one says an input-output system is passive if the energy injected into the system is greater than or equal to the energy stored within the system. When this inequality is strict, then passivity is sufficient for the asymptotic stability of the undriven system. Passivity has its origins in network (circuit) synthesis, but can be generalized to other types continuous-time systems.

Passivity became important in control due to its connections with classical stability theory and the fact that passive systems are easily stabilized through high-gain feedback. One of the most attractive features of passive systems is that passivity is preserved under arbitrary system interconnections. This is useful in the control of large-scale interconnected systems [MH78] and as a result passivity frameworks have appeared for networked robotic systems [AS89], network congestion [WA04], and networked cyber-physical systems[SKK$^+$12].

This paper examines the use of passivity concepts in managing cloud computing systems [AFG$^+$10]. We adopt a

Feedback Computing '12 San Jose, California USA
.

market-orient viewpoint [BYV08] in which the cloud consists of a set of interconnected *brokers* and *servers* where brokers route consumer workloads to servers. The resulting network of brokers and servers interact in a manner that is reminiscent of network congestion problems. A passivity framework for such network congestion control problems was introduced in [WA04] and using that framework, this paper suggests a passivity framework for controlling cloud systems. One of the main findings of this paper is that one can use passivity concepts as a *certificate* whose satisfaction is sufficient for the safe operation of the system.

The remainder of this paper is organized as follows. Section 2 introduces a formal model for the cloud computing system. Section 3 discusses the problem of selecting an appropriate feedback signal and section 4 uses passivity to address this issue. Concluding remarks are in section 5.

2. CLOUD COMPUTING APPLICATION

This paper adopts a market-orient view [BYV08] in which the "cloud" consists of a set $\mathcal{B}$ of *brokers* and a set $\mathcal{S}$ of *servers*. Brokers and servers are seen as independent economic agents in a large-scale business enterprise. For notational convenience we assume that $|\mathcal{B}| = N$ and $|\mathcal{S}| = M$. Figure 1 shows a block diagram for this "cloud" system. Brokers act as admission control agents, determining how much of the consumer's workload, $w(k)$, should be routed to each server in $\mathcal{S}$ in the kth time interval. The servers process a portion of the received workload. The workload that is not completed by the server at the end of the kth time interval is buffered as the backlogged workload. The server sends back to the broker a *throttling signal* that is used by the broker to control how much of the consumer's workload will be routed to the servers.

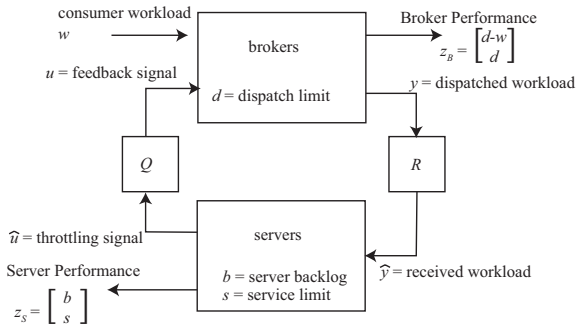

Figure 1: Cloud Computing Application

We now give a more formal description of the system in Figure 1. At time instant $k \in \mathbb{Z}^+$, the ith broker ($i \in \mathcal{B}$) receives the *consumer's workload* $w_i(k)$. The vector of consumer workloads received by all brokers is denoted by the vector $w(k) = [w_1(k), \cdots, w_N(k)]^T$. The maximum workload dispatched by broker i at time k is denoted as $d_i(k)$. The actual workload dispatched by broker i is

$$y_i(k) = \min\{d_i(k), w_i(k)\} \tag{1}$$

The maximum dispatch level, d_i, is a "control" for the broker that satisfies the difference equation

$$\begin{aligned} d_i(k+1) &= [d_i(k) + \beta_{i1}(w_i(k) - d_i(k)) \\ &\quad -\beta_{i2}u_i(k)]^+ \end{aligned} \tag{2}$$

where $\beta_{i1}, \beta_{i2} > 0$ are real-valued *control gains* and $[x]^+ = \max\{0, x\}$. The signal $u_i(\cdot) : \mathbb{Z}^+ \to \mathbb{R}$ is a *throttling signal* sent back to server i from the brokers. Together equations (2) and (1) represent the systems in the upper block of Figure 1.

The workload, $y_i(k)$, dispatched by broker i at time k is routed to the servers in $\mathcal{S}$. The amount of workload routed from broker i to server j at time k is denoted as $r_{ji}y_i(k)$ where $\sum_{j=1}^{M} r_{ji} = 1$ and $0 \le r_{ji} \le 1$ for all i and j. The coefficient r_{ji} may be viewed as a routing decision that broker i makes. This routing decision may be time varying. The total workload received by server j at time k is therefore

$$\hat{y}_j(k) = \sum_{i=1}^{N} r_{ji}y_i(k)$$

Let $y(k) = [y_1(k), \cdots, y_N(k)]^T$ and $\hat{y}(k) = [\hat{y}_1, \cdots, y_M(k)]^T$ with $R = \{r_{ji}\}$ being the *routing matrix*, the forward route from the brokers to servers may be written in matrix-vector form as $\hat{y}(k) = Ry(k)$.

At time k, we let $b_j(k)$ denote the workload that is waiting for processing on server j. Clearly, $\hat{y}_j(k)$, denotes the new workload arriving at server j at time instant k. The total workload that needs to be processed by server j is therefore $b_j(k) + \hat{y}_j(k)$. Server j processes at most $s_j(k)$ of this workload where $0 \le s_j(k) \le \overline{s}_j$. The constant $\overline{s}_j$ denotes the physical service limit for the server. Let $e_j(k) = b_j(k) + \hat{y}_j(k) - s_j(k)$ denote the *excess workload*. The service limit, $s_j(k)$, and the backlogged workload $b_j(k)$ are internal states of the server that satisfy

$$b_j(k+1) = [e_j(k)]^+ \tag{3}$$

$$s_j(k+1) = \min\left\{\overline{s}_j, [s_j(k) + \sigma_j e_j(k)]^+\right\} \tag{4}$$

We assume that the server generates a *throttling signal*

$$\hat{u}_j(k) = h(b_j(k), s_j(k), \hat{y}_j(k)) \tag{5}$$

where $h(\cdot,\cdot,\cdot)$ is a function that will be defined in the next section. Together equations (3-5) represent the lower block in Figure 1.

The throttling signal, $\hat{u}_j$, is routed back to the brokers through a *return matrix*, $Q = \{q_{ij}\}$. where $\sum_{j=1}^{M} q_{ij} = 1$ and $0 \le q_{ij} \le 1$. If we let $u(k) = [u_1(k), \cdots, u_N(k)]^T$ and $\hat{u}(k) = [\hat{u}_1(k), \cdots, \hat{u}_M(k)]^T$, then the relation between the throttling signal, $\hat{u}$, generated by the servers and that signal u received by the brokers can be written in matrix-vector form as $u(k) = Q\hat{u}(k)$.

The broker's system equation 2 forms a *control system* in which the gain β_{i1} is chosen to ensure tracking of the consumer's workload w_i and β_{i2} is selected to adjusted how strongly the server's feedback signal $u_i(k)$ throttles the broker. The first gain, particular, may be selected independently of the throttling signal to minimize the broker's "cost" function

$$J_i^B = \lim_{L\to\infty} \frac{1}{L} \sum_{k=0}^{L} \left((d_i(k) - w_i(k))^2 + \rho_i^B d_i^2(k) \right) \tag{6}$$

where $\rho_i^B > 0$ is a weighting coefficient. This objective may be seen as trying to minimize the mean square error between the consumer's workload and the broker's dispatch rate.

The server's system equation (4) is also a control system with control gain σ_j. This control gain may be selected to

minimize a local "cost" function of the form,

$$J_j^S = \lim_{L\to\infty} \frac{1}{L} \sum_{k=0}^{L} \Big(b_j^2(k) + \rho_j^S s_j^2(k) \Big) \tag{7}$$

where $\rho_j^S > 0$ is a weighting coefficient. This objective may be seen as trying to minimize the server's power consumption (i.e. keep s_j small) while simultaneously reducing the server's response time (i.e., keeping the backlog, b_j, small). In general, these two objectives conflict with each other and the coefficient ρ_j^S is chosen to balance the tradeoff between these two objectives.

3. PROBLEM STATEMENT

As noted in earlier papers, assuring the asymptotic stability of a number of systems in isolation, does not guarantee stability of the interconnection. One must select an appropriate feedback or throttling signal to ensure the server backlogs remain bounded under overload scenarios. To explore this problem, let us consider the impact that the selection of throttling signal, $u_j(k)$, has on the system's ability to handle overloads.

We first have the brokers and servers select gains that minimize the two cost functions in equations (6) and (7) for the weighting coefficients $\rho_i^B = 01$ and $\rho_j^S = 0.5$. With these gains the brokers place a high value on tracking consumer workload closely, whereas the server places a high value on minimizing power costs. For this simulation example, the optimal server gain is $\sigma_j^* = 0.2$ and the optimal broker gain is $\beta_{i1}^* = 0.95$. These gains assure the asymptotic stability of the broker and server equilibria when the subsystems are disconnected from each other.

We now consider the feedback interconnection of brokers and servers in which the feedback signal, u_j, used to throttle the broker's dispatch level is

$$\hat{u}_j(k) = \min\{s_j(k), b_j(k) + \hat{y}_j(k)\}$$

We call this the *serviced workload* and it is the total amount of work returned to the broker by the server at the end of the kth time interval. Since this is passed through the return routing matrix, Q, this signal is something that can be measured directly by the broker and serves as a crude measure of congestion.

We now simulate this system with a consumer workload that is modeled as an anomalous Brownian motion. Fluctuations in the consumer workload are generated by two processes; a normally distributed i.i.d random process with mean 2 and variance 1 and a Poisson jump process that injects large jumps in consumer demand at random points in time. In this simulation example there are 3 brokers and 6 servers with a forward routing matrix R that is randomly selected at the start of the simulation. The capacity limit on the servers, $\overline{s}_j$, is taken to be 3. The first Poisson jump occurs at time instant 250 and jumps the workload to 25. The second Poisson jump occurs at time 500 and returns to the nominal consumer workload of 2. The system is overloaded over the time interval $[250, 500]$ since the consumer workload exceeds the server capacity limit, $\overline{s}_j$ over this time interval.

Figure 2 shows the results when $\beta_{i2} = 0.1$. This figure plots the time histories for the server's maximum dispatch level, d_i, the backlogged workload, b_j, the service limit, s_j, and the cost functionals, J^B and J^S. There are two things to

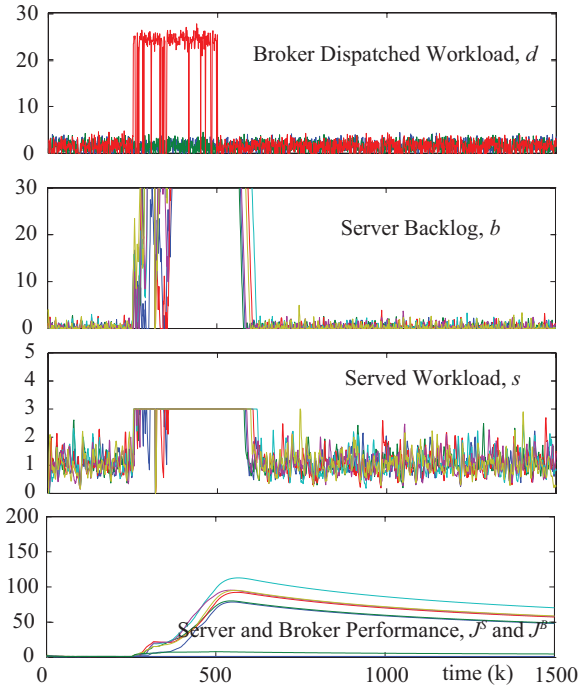

Figure 2: Throttle with Serviced Workload

note in these results. First, the system is "unstable" when the consumer input jumps to 25. Secondly, the system is not very "resilient", since it takes nearly 100 time-steps for the system to return to "normalcy" after the consumer's input, w_i, returns to its normal level. The unstable nature of the system is seen in the cost functional time history, which exhibits an exponential rate of growth when the consumer demand jumps up. This is to be expected, of course, because the increase in workload arrival is greater than the maximum service rate $\overline{s}_j$ supplied by the servers. After the consumer demand returns to normal, however, the plots in Figure 2 show that it takes a relatively long time (100 time steps) before the server states return to their normal levels. We take this time of return as a measure of the system's resilience to the overload situation.

We now consider a different throttling signal. In particular, let's assume that the server sends information about its backlogged workload, b_j, through the return routing matrix. In particular, we let

$$\hat{u}_j(k) = 2\sigma b_j(k) + 2\sigma s_j(k) \tag{8}$$

Again we set the broker's throttling gain, $\beta_{2i} = 0.1$, and an overloading input drives one of the brokers between $[250, 500]$. The results for this simulation are shown in figure 3. In this case, we see a more "controlled" increase in the cost functionals. In particular, the cost J^S no longer grows in an uncontrolled exponential manner. Instead, these costs grow and appear to converge to a constant value. This system, therefore, appears to be "stable" during the overload scenario. As a result, when the overloading input is removed, we see the server states quickly return to normal. Since the return time to normalcy is much shorter than the 100 time steps seen in Figure 2, we can conclude that the throttling signal used in equation (8) results in a system that is more resilient to overloads.

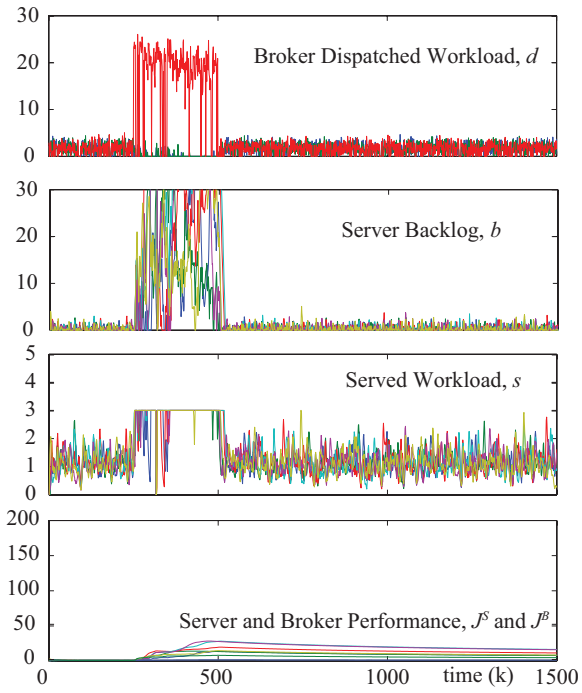

Figure 3: Throttle with Server Backlog

The preceding simulations in Figures 2 and 3 demonstrate that the choice of feedback signal, $u_j(k)$ can have a profound impact on the behavior of the networked system during overload scenarios. What is the fundamental difference between the use of "serviced workload" and "backlogged workload" that makes one a better feedback than the other? The next section shows that the selection of an appropriate feedback signal can be made using a *passivity* framework.

4. PASSIVITY FRAMEWORK

Passivity refers to the basic idea that the power flowing into a system should not exceed the energy stored within it. In particular, consider a discrete-time system whose state-space representation may be written as

$$\begin{aligned} x(k+1) &= x(k) + f(x(k), u(k)) \\ y(k) &= h(x(k), u(k)) \end{aligned}$$

where u is the input signal and y is the output signal. We say that this system is *passive* if there exists a positive definite function $V(\cdot) : \mathbb{R}^n \to \mathbb{R}$ such that

$$\Delta V(x(k)) \leq u^T(k) y(k) \tag{9}$$

The function V is usually called a *storage function*. It represents the energy stored within the system. The inner product, $u^T y$, represents the instantaneous power flowing into the system. If there is a positive definite function $W(\cdot) : \mathbb{R}^n \to \mathbb{R}$ such that

$$\Delta V(x(k)) \leq u^T(k) y(k) - W(x(k)) \tag{10}$$

then this system is *strictly passive*. If one replaces $W(x)$ with $W(y)$, then the system is said to be *strictly output passive*. We sometimes refer to equations (9) and (10) as *dissipative inequalities*. Finally, if no solution of the difference equation $x(k+1) = f(x(k), 0)$ other than the trivial solution $x(k) = 0$ can satisfy $0 = h(x(k), 0)$, then we say the system is *zero-state observable*.

Passivity is useful in control theory due to its connections with asymptotic stability as summarized in the following proposition,

PROPOSITION 1. *[vdS00] Consider an input-output system, G, such that $f(0, 0) = 0$. If this system is strictly output passive and zero-state observable, then the origin of $x(k+1) = x(k) + f(x(k), 0)$ is asymptotically stable.*

Another important property of passivity is that feedback interconnections of passive systems are again passive. Feedback connections such as those shown in Figure 1 are strictly output passive when their constituent subsystems possess some degree of passivity. This well known result is summarized in the following proposition.

PROPOSITION 2. *[vdS00] Consider the feedback interconnection of two systems, G_1 and G_2 in which one system is strictly passive and the other is passive and zero-state observable, then the feedback system is strictly output passive and zero-state observable.*

Since the "cloud" system in Figure 1 is a feedback system, if we can show that the broker is strictly passive and the servers are passive and zero-state observable, then one can conclude that the equilibrium for the cloud computing system will be asymptotically stable. The passivity of the server system is proven below.

PROPOSITION 3. *The server system defined in equations (3-5) is passive and zero-state observable.*

Proof: Equations (3-5) may be rewritten as

$$\begin{aligned} \xi(k+1) &= \min\left\{\overline{\xi}, [A\xi(k) + B\hat{y}(k)]^+\right\} \\ \hat{u}(k) &= C\xi(k) \end{aligned}$$

where $\xi = [b, s]^T$, $\overline{\xi} = [\infty, \overline{s}]^T$, $A = \begin{bmatrix} 1 & -1 \\ \sigma & 1-\sigma \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ \sigma \end{bmatrix}$, and $C = \begin{bmatrix} 2\sigma & 2\sigma \end{bmatrix}$. Note that for notational convenience, we've dropped the subscript on the server states. From these equations it can be readily seen that the system is zero-state observable. Let's consider a storage function of the form

$$V(\xi) = \xi^T P \xi$$

where $P = \begin{bmatrix} \sigma & -\sigma/2 \\ -\sigma/2 & 1 \end{bmatrix}$. With this choice for V, we can now go ahead and compute the first difference

$$\Delta V(\xi) = V(\xi(k+1)) - V(\xi(k))$$

Due to the non-smooth nature of the right-hand side of the difference equations, there are a number of conditions we need to consider. We classify these conditions on the basis of the backlog projection in equation (3) being active/inactive. It can be readily seen that the projection operator in equation (4) can never be active.

Backlog Projection is inactive: In this case, the first difference can be written as

$$\Delta V(\xi) = \xi^T(k+1) P \xi(k+1) - \xi^T(k) P \xi(k)$$

This bound holds whether $s(k+1)$ hits its upper limit or not. This difference may be rewritten as

$$\Delta V(\xi) \leq \xi^T \left(A^T P A - P \right) \xi + 2\xi^T A^T P B \hat{y}$$

For the given choice of P, we have $A^T P A - P = 0$ and we have $2\xi A^T P B \hat{y} \leq \xi^T C \hat{y}$ which implies $\Delta V \leq \hat{u}\hat{y}$ and so the dissipative inequality is satisfied.

Back Projection active: Let's consider

$$\begin{aligned} \Delta V(\xi) &= ((1-\sigma)s + \sigma(b+\hat{y}))^2 \\ &\quad -\sigma b^2 + \sigma b s - s^2 \end{aligned}$$

Since the backlog project is active, we know that $b + \hat{y} < s$ and in particular since b and $\hat{y}$ are non-negative this means that $\sigma b s \leq \sigma s^2$ and we rewrite the above inequality as

$$\begin{aligned} \Delta V(\xi) &\leq ((1-\sigma)s + \sigma(b+\hat{y}))^2 \\ &\quad -\sigma b^2 + (\sigma - 1)s^2 \end{aligned}$$

We now expand out the first square term and collect terms in the states b and s to obtain

$$\begin{aligned} \Delta V(\xi) &\leq (-1 + \sigma + (1-\sigma)^2)s^2 \\ &\quad +2\sigma(1-\sigma)bs + (\sigma^2 - \sigma)b^2 \\ &\quad +2\sigma^2 b\hat{y} + \sigma^2 \hat{y}^2 + 2\sigma(1-\sigma)s\hat{y} \end{aligned}$$

It can be shown that the largest value the first two lines can take is zero, so the above inequality reduces to

$$\Delta V(\xi) \leq 2\sigma^2 b\hat{y} + \sigma^2 \hat{y}^2 + 2\sigma(1-\sigma)s\hat{y}$$

Again $\hat{y} < s$ since the backlog projection is active. Using this fact in the second term we obtain

$$\begin{aligned} \Delta V(\xi) &\leq 2\sigma^2 b\hat{y} + (2\sigma - \sigma^2)s\hat{y} \\ &\leq 2\sigma^2 b\hat{y} + 2\sigma s\hat{y} \\ &\leq 2\sigma b\hat{y} + 2\sigma s\hat{y} \\ &= \hat{u}\hat{y} \end{aligned}$$

where we used the fact that $0 < \sigma < 1$. So the dissipative inequality is again satisfied. Since the dissipative inequality is satisfied for all conditions, this system is passive. $\diamondsuit$

We now establish that the broker subsystem is strictly passive

PROPOSITION 4. *The broker system defined in equations (1-2) is strictly passive.*

Proof: We assume that the consumer input $w_i = 0$ and only consider the input-output system from u to d. In this case, the state equations may be rewritten as

$$d(k+1) = [(1-\beta_1)d(k) - \beta_2 u(k)]^+$$

where the gains $\beta_1, \beta_2 > 0$. In this case $u(k)$ is the input and the output $y(k) = d(k)$. We select the storage function $V(d) = d^2$ and compute the first difference. Again we need to consider the case when the projection operator is active and inactive.

Projection Inactive: This means that the first difference is

$$\begin{aligned} \Delta V(d) &= ((1-\beta_1)d - \beta_2 u)^2 - d^2 \\ &= (-1 + (1-\beta_1)^2)d^2 \\ &\quad -2\beta_2(1-\beta_1)ud + \beta_2^2 u^2 \end{aligned}$$

Since the projection is inactive, we know that $(1-\beta_1)d > \beta_2 u$. Inserting this into the last term yields,

$$\begin{aligned} \Delta V(d) &\leq (-1 + (1-\beta_1)^2)d^2 \\ &\quad -\beta_2(1-\beta_1)ud \end{aligned}$$

which implies this block is strictly passive since the output $y = d$.

Projection Active. In this case the first difference is

$$\Delta V(d) \leq -d^2 \leq 0$$

which implies this system is also strictly passive. $\diamondsuit$.

Propositions 3 and 4 establish that the broker is strictly passive and that the server is passive and zero-state observable. Combining this result with proposition 1 and 2, allows us to conclude that the overall cloud system's equilibrium point is asymptotically stable as was seen in the earlier simulations of section 3.

It is important to note that other methods could have been used to establish asymptotic stability when server backlog is used as a throttling signal. The passivity framework, however, possesses certain attractive features. One feature relaxes the requirement for precise system models; particularly with regard to system gains. For example as long as the local gains, σ and β_1 are chosen to assure the isolated subsystem's stability, then the closed loop system is stable under any positive throttling gain and under any routing matrix (Q and R). There may, of course, be a wide variation in the performance attainable under such a wide range of parameters, but the important structural property of system stability is preserved.

Perhaps the most important feature of the passivity framework rests with its modularity. The fact that passivity is preserved under any arbitrary interconnection of subsystems means that passivity of the overall system can be effectively assured by simply checking the passivity of each individual subsystem. This is something that can be checked locally, in a way that keeps private a subsystem's internal state information. For a cloud built on free market principles, such a local method for subsystem *certification* provides a scalable way of upgrading the overall system. In particular, one can introduce a *passivity certificate*

$$C(x,u,y) = u^T y - \Delta V(x) \tag{11}$$

for each broker and server in the system. The certificate in equation (11) is simply the dissipative inequality characterizing the local system's passivity and by verifying that $C(x(k), u(k), y(k)) \geq 0$ for all k, then we know this subsystem is passive and hence can be "safely" interconnected into the larger system.

Figure 4 shows the passivity certificate for two simulation runs presented above. The top plot shows the servers passivity certificates when the server transmits the backlogged workload as the throttle. As predicted in propositions 3 and 4, these certificates are positive, which is consistent with the fact that the server systems are passive. The bottom plot shows the same certificates for the case using the served workload as a throttle. the throttling Figure 4 we see that server's certificate is negative when the system is overloaded, which is consistent with the loss of system stability.

Passivity certificates, of course, only represent sufficient conditions for a passive system, but these are certificates that the individual subsystem "publishes" to the entire group as a certificate of the subsystem's ability to work safely (i.e.

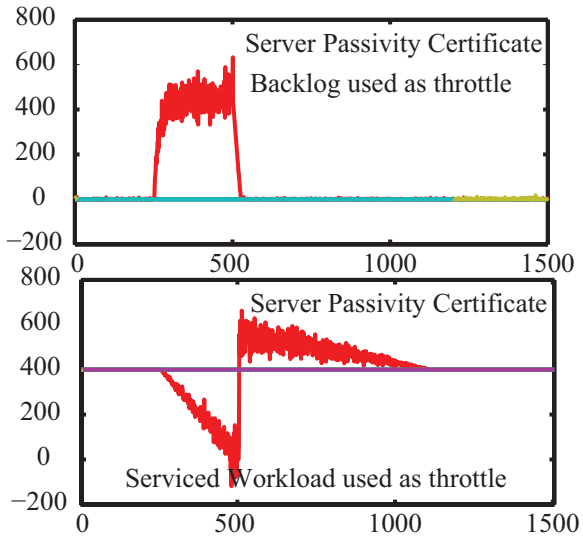

Figure 4: Passivity Certificates for Examples

stably) with its neighbors. Violation of the certificate can be detected locally in time and may therefore be used as a method for detecting potential faults. This "certification" procedure is highly decentralized since no central authority is needed to monitor and observe the internal states of the subsystems. The certificate can be computed locally by an agent based on its published inputs and outputs and another published function of state $\Delta V(x)$. In no case is complete knowledge of the internal and potentially private operations of the agent exposed to the outside world. The use of such passivity certificates may therefore provide a secure mechanism for detecting faults within the cloud system.

5. SUMMARY

This paper describes a passivity framework for the cloud computing systems in which brokers and servers act in a market-oriented manner. The passivity framework provides modular stability guarantees for ad hoc interconnections of subsystems. This means system designs can be done on a piece-by-piece basis and as long as these systems satisfy a published passivity certificate, the interconnection of the system is guaranteed to assure stable operation.

Acknowledgements: The authors acknowledge the partial financial support of the National Science Foundation (CNS-0931195).

6. REFERENCES

[ADH+08] T. Abdelzaher, Y. Diao, J.L. Hellerstein, C. Lu, and X. Zhu. Introduction to control theory and its application to computing systems. *Performance Modeling and Engineering*, pages 185–215, 2008.

[AFG+10] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[AS89] R.J. Anderson and M.W. Spong. Bilateral control of teleoperators with time delay. *Automatic Control, IEEE Transactions on*, 34(5):494–501, 1989.

[BYV08] R. Buyya, C.S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. Ieee, 2008.

[DGH+02] Y. Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury. Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 219–234. IEEE, 2002.

[KKZ05] M. Karlsson, C. Karamanolis, and X. Zhu. Triage: Performance differentiation for storage systems using adaptive control. *ACM Transactions on Storage (TOS)*, 1(4):457–480, 2005.

[LWK05] C. Lu, X. Wang, and X. Koutsoukos. Feedback utilization control in distributed real-time systems with end-to-end tasks. *Parallel and Distributed Systems, IEEE Transactions on*, 16(6):550–561, 2005.

[MH78] P. Moylan and D. Hill. Stability criteria for large-scale systems. *Automatic Control, IEEE Transactions on*, 23(2):143–149, 1978.

[RRT+08] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No power struggles: Coordinated multi-level power management for the data center. *ACM SIGOPS Operating Systems Review*, 42(2):48–59, 2008.

[SKK+12] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and Shige Wang. Toward a science of cyber physical system integration. *Proceedings of the IEEE*, 100(1):29 –44, jan. 2012.

[vdS00] A.J. van der Schaft. *L2-gain and passivity techniques in nonlinear control*. Springer Verlag, 2000.

[WA04] J.T. Wen and M. Arcak. A unifying passivity framework for network flow control. *Automatic Control, IEEE Transactions on*, 49(2):162–174, 2004.

[WJLK07] X. Wang, D. Jia, C. Lu, and X. Koutsoukos. Deucon: Decentralized end-to-end utilization control for distributed real-time systems. *Parallel and Distributed Systems, IEEE Transactions on*, 18(7):996–1009, 2007.

[XZSW06] W. Xu, X. Zhu, S. Singhal, and Z. Wang. Predictive control for dynamic resource allocation in enterprise data centers. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 115–126. IEEE, 2006.

[ZUW+09] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin. What does control theory bring to systems research? *ACM SIGOPS Operating Systems Review*, 43(1):62–69, 2009.