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

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.**

I. Introduction

Control theoretic methods have been used to manage a wide range of computational systems that include real-time embedded systems [LWK05], [WJLK07], storage systems [KKZ05], web servers [DGH$^+$02], and virtualized data centers [XZSW06], [RRT$^+$08], [WCLK12], [UKIN10], [WW11]. With the exception of [UKIN10], 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 Guarantees:** 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, it is unclear how well-suited "classical" methods will be for real-life computing applications.
- **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 constrained to be *positive*. This restriction introduces nonlinear dynamics that cannot be easily "linearized" away.
- **Unknown Models:** Classical methods presume the designer has a formal model of the plant dynamics. To assure robust performance, the designer must also know bounds on system uncertainties. Dynamic models for computational systems are rarely known beforehand and even if such models were available, the need to keep certain information "private" prevents such modeling information from being widely used.
- **Evolution versus Engineering:** Classical control was developed for *engineered* systems; i.e. systems whose development have a well-defined termination point. Computational systems, however, *evolve* into being as software upgrades and new components are incrementally added into an existing system. For such *ad hoc* systems, the use of classical methods often leads to conservative controllers that trade away performance for safety.

Classical control is ill-equipped to deal with the above issues. Classical methods, however, are over 50 years old [Bod56], and many other methods have been developed over that time period. One such technique that seems well-suited for managing distributed systems is *passivity-based control*.

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. Injected power is usually measured by the product of inputs and outputs while stored energy is measured using a *storage function*. 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 [Gui57], but can be generalized to other types continous-time systems [Wil72].

Passivity became important in control due to its connections with classical stability theory [Zam66] and the fact that passive systems are easily stabilized through high-gain feedback. In some cases, it is possible to use feedback to passivate a system [BIW91], thereby making passivity an important tool in nonlinear control. One of the most attractive features of passive systems is that passivity is preserved under arbitrary system interconnections [HM76] which is useful in the control of large-scale interconnected systems [MH78]. As a result passivity-frameworks for control of interconnected systems have appeared for robotic systems [AS89], network congestion [WA04], and networked cyber-physical systems[SKK+12].

Cloud computing systems [AFG+10] may be viewed as an interconnected set of *brokers* and *servers* in which brokers route consumer workloads to servers. The resulting network of brokers and servers interact in a market-orient manner [BYV08] that is reminiscent of network congestion problems [KMT98]. A passivity framework for such network congestion control problems was introduced in [WA04] and building upon that framework, this paper suggests a passivity framework for cloud computing systems to coordinate response-time management and power control across the entire cloud enterprise. One of the main findings of this paper is that one can use passivity concepts as a *certificate* whose satisfaction assures the safe operation of the system in the presence of feedback delays and uncertain dynamics.

The remainder of this paper is organized as follows. Section II reviews definitions and notational conventions used throughout the paper. Section III introduces a formal model for a cloud computing system. Section IV provides an example of *power conflicts* between brokers and servers; reminiscent of issues identified in [RRT+08]. Section V introduces a passivity framework for the cloud computing system and shows how the dissipative inequality characterizing a system's passivity may be used as a *certificate* assuring that servers and brokers can be interconnected in an ad hoc manner. Conclusions are presented in section VI.

II. Mathematical Background

This section reviews formal notational conventions used throughout the paper. Let $\mathbb{Z}$ and $\mathbb{R}$ denote the set of integers and real numbers, respectively. Given a finite set, Ω, then $|\Omega| \in \mathbb{Z}^+$ represents the number of elements in Ω. The linear space of all real-valued n-dimensional vectors ($n \in \mathbb{Z}^+$) is denoted as $\mathbb{R}^n$. $|x|$ denotes the Euclidean norm of a vector in $\mathbb{R}^n$. Given a matrix $A \in \mathbb{R}^{m\times n}$, we let $a_{ji} \in \mathbb{R}$ denote the element on the jth row and ith column of the matrix. When defining a matrix in terms of its components, we will often write this as $A = \{a_{ji}\}$. A function $F(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is *positive* (semi) *definite* if $F(\xi) > 0$ ($F(\xi) \geq 0$) for all $\xi \neq 0$ and $F(0) = 0$. A matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if the function $F(\xi) = \xi^T A\xi$ is positive definite.

A *discrete-time signal*, x, is a function $x(\cdot) : \mathbb{Z}^+ \to \Omega$ where $\Omega \subset \mathbb{R}^n$. We let $x(k) \in \Omega$ denote the value that signal x takes at time instant $k \in \mathbb{Z}^+$. Given a discrete-time signal, x, we let $[x]^+ = \max(0, x)$ denote the *positive projection* of x.

A discrete-time *positive* system, $\mathbf{G}$, with input signal $u(\cdot) : \mathbb{Z}^+ \to \mathbb{R}^m$ and output signal $y(\cdot) : \mathbb{Z}^+ \to \mathbb{R}^m$ has an *internal state* $x(\cdot) : \mathbb{Z}^+ \to \mathbb{R}^n$ that satisfies the difference equation

$$\mathbf{G} \quad \left\{ \begin{array}{lcl} x(k+1) & = & [x(k) + f(x(k), u(k))]^+ \\ y(k) & = & h(x(k), u(k) \end{array} \right.$$

for all $k \in \mathbb{Z}^+$ with $x(0) = x_0$ specified as the system's *initial condition* and $f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ are piecewise continuous functions.

The vector $\overline{x} \in \mathbb{R}^n$ is called an *equilibrium point* if $f(\overline{x}, 0) = 0$. Given a discrete-time positive system $\mathbf{G}$ with equilibrium point at 0 and a positive definite function $V(\cdot) : \mathbb{R}^n \to \mathbb{R}^+$, we define the *first difference* of V as the function $\Delta V(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ that takes values

$$\Delta V(x, u) = V([x + f(x, u)]^+) - V(x)$$

III. Cloud Computing Application

This section presents a formal model for a cloud computing system. The "cloud" consists of a set $\mathcal{B}$ of *brokers* and a set $\mathcal{S}$ of *servers*. 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 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. A more formal description of the broker/server workflows is given below.

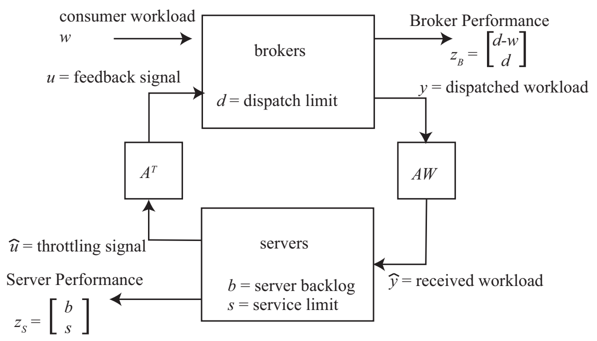


Fig. 1. Cloud Computing Application

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 state variable for the broker and it satisfies the following 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*. 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 a discrete-time positive system.

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 $0 \leq r_{ji} \leq 1$ 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 equal to

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

Note that if we let $y(k) = [y_1(k), \cdots, y_N(k)]^T$, $\hat{y}(k) = [\hat{y}_1, \cdots, y_M(k)]^T$, and $R = \{r_{ji}\}$, then the forward route from the brokers to the 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 $0 \leq s_j(k) \leq \overline{s}_j$. 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 the following difference equations,

$$\begin{aligned} b_j(k+1) &= [e_j(k)]^+ && (3) \\ s_j(k+1) &= \min\left\{\overline{s}_j, [s_j(k) + \sigma_j e_j(k)]^+\right\} && (4) \end{aligned}$$

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 discrete-time system shown in figure 1. The second state equation (4) determines the server's maximum service rate, $s_j(k)$. This determination is controlled by the gain $\sigma_j > 0$.

The throttling signal, $\hat{u}_j$, is routed back to the brokers through a *return routing matrix*, $Q = \{q_{ij}\}$. where $\sum_{j=1}^{M} q_{ij} = 1$ and $0 \leq q_{ij} \leq 1$. 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 $u = Q\hat{u}$.

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

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

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,

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

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.

Note that these control gains are chosen based on the broker's or server's local state information. Once a set of weighting coefficients (ρ_i^B and ρ_j^S) have been selected, a simple simulation can be used to search for the *optimal* gains that minimize the two cost functions in equations (6) and (7). An example is shown in Figure 2 with $\rho_i^B = 0.1$ and $\rho_j^S = 0.5$. With these choices, the broker places high value on tracking the consumer workload closely, whereas the server places a higher value on keeping power costs down. Figure 2 plots the cost functions J^B and J^S as a function of the control gains β_{i1} and σ_j, respectively. From these plots we see that the optimal sever gain is $\sigma_j^* = 0.2$ and the optimal broker gain, $\beta_{i1}^* = 0.95$. These gains guarantee the local asymptotic stability of the broker and server subsystem's equilibrium point when the systems are disconnected from each other. In the next section, we examine how well these locally stable systems using optimal gains work together when they are interconnected.

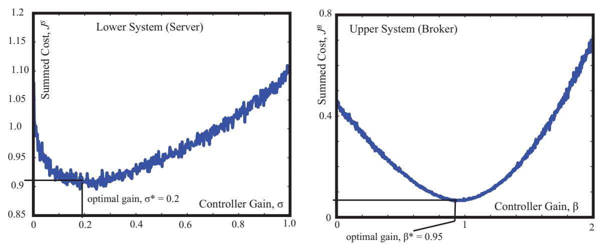


Fig. 2. Broker and Server Summed Costs versus Controller Gains

IV. Power Conflicts

As noted in earlier papers, assuring the asymptotic stability of a number of systems in isolation, does not guarantee stability of the interconnection. This is even true of simple cascade connections. In particular, let's consider our cloud system in which the throttling gain β_{2i} is set to zero in all brokers. With this gain set to zero, the feedback loop in Figure 1 is broken and we have a simple cascade connection from the broker to the server. Let's now consider a simulation run in which the initial consumer workload is an i.i.d random process that is normally distributed with mean 2 and variance 1. There are 3 brokers and 6 servers with a forward routing matrix that is randomly selected at the start of the simulation. The capacity limit on the servers, $\overline{s}_j$, is taken to be 3. At time 250, the mean workload jumps to 25 and then drops back to its nominal value of 2 at time instant 500. Over the time interval $[250, 500]$ the system is in an overload situation and the question is how well does this cascade interconnection of two locally stable and optimal systems handle the overload.

Simulation results are shown in Figure 3. 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. At time instant 250 a large consumer workload is injected into one of the brokers and we see a corresponding jump in that broker's dispatch level as it seeks to follow the increase. The impact that this disturbance has on the downstream servers is seen in the remaining plots. The backlog increases, the service rate increases until it hits its capacity level, $\overline{s}_j$. This behavior is to be expected because the arrival rate of the workload into the server is greater than the maximum service limit. Once the disturbance is removed from the system (time instant 500) the server eventually returns back to normal at about time instant 600.

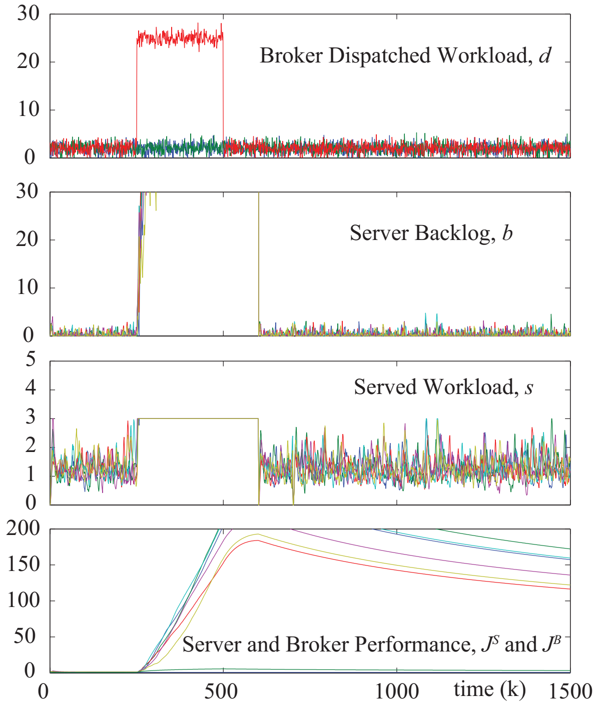


Fig. 3. Simulation of Cascaded System (throttling signal not used)

There are two things to 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 3 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. For this cascaded system, it should be clear that even though the individual controllers were chosen to be optimal, the resulting system is not very resilient.

The reason for the cascaded system's instability is fairly obvious. The broker is not lowering its dispatch level to stay within the capacity of the servers. One obvious way of handling this is to use the feedback signal, u_j, provided by the servers to throttle or reduce the broker's dispatch level. The question is what should we choose for this throttling signal? One obvious choice is to simply let

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

This 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, the actual signal received by the broker is simply a fraction of the workload serviced. This is something that can be measured directly by the broker and may be taken as a crude measure of how congested the server is. We can then select a gain β_{i2} which forces the broker to reduce (throttle) its dispatch rate when the server's dispatch level, s_j, is large. Figure 4 shows the results for such a simulation run when $\beta_{i2} = 0.1$. Figure 4 shows that this feedback system is better able to control the unstable growth in the server's states under overloads. However, it still takes nearly 100 time steps to return to normal after the overloading inputs are removed. This system still fails to exhibit the resilience to overloads that one would hope for.

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 5. 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 4,

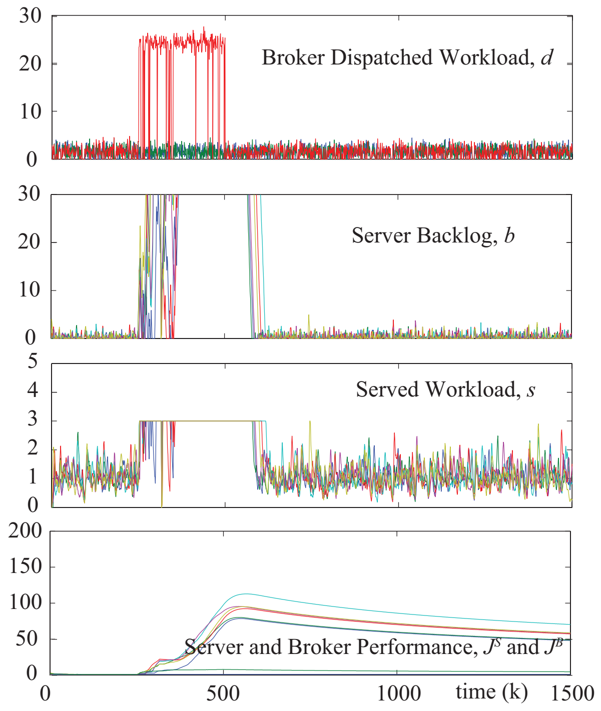


Fig. 4. Simulation Results using Serviced Workload as Feedback Throttle

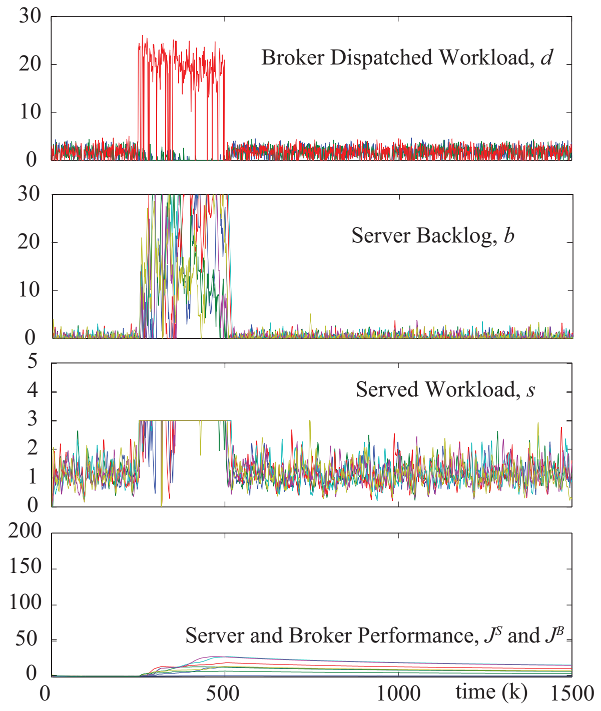


Fig. 5. Simulation Results using Backlogged Workload as Feedback Throttle

we can conclude that the throttling signal used in equation (8) results in a system that is more resilient to overloads. Why should the backlogged workload be a better throttling signal than the served workload? The next section shows that this is because using backlog as the feedback signal renders the server subsystems *passive*.

V. Passivity Framework

Passivity refers to the basic idea [Wil72] 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 $V(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 $W(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$ 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. Consider the feedback interconnection shown in Figure 6 and assume that the upper system G_1 is strictly passive while the lower system G_2 is passive and zero-state observable. It is then possible to show that the entire feedback system is strictly output passive and zero-state observable as stated in the following proposition. Variations on this result will be found in [vdS00]. In this result, G_1 and G_2 are both passive with storage function V_1 and V_2, respectively. The result is proven by selecting $V = V_1 + V_2$ as the storage function for the feedback system and then showing that the dissipative inequality in equation (10) is satisfied.

Proposition 2: Consider the feedback interconnection shown in Figure 6 in which G_1 is strictly passive and G_2 is passive and zero-state observable, then the feedback system is strictly output passive and zero-state observable.

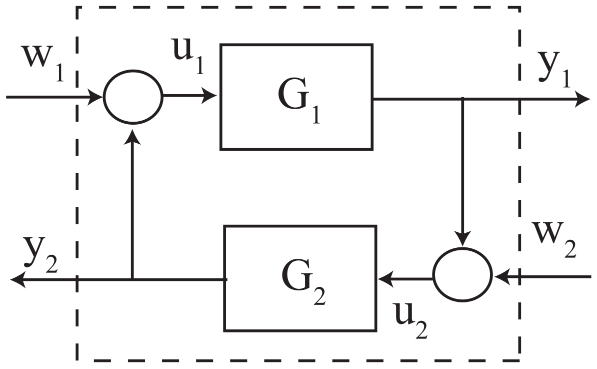


Fig. 6. Feedback Interconnection

The implication of Propositions 1 and 2 is that if we can show the broker system in Figure 1 is strictly passive and the server system is passive and zero-state observable, then we can conclude the equilibrium for the cloud computing system in Figure 1 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

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

  For the given choice of P, we have $A^T PA - P = 0$ and we have $2\xi A^T PB\hat{y} \le \xi^T C\hat{y}$ which implies $\Delta V \le \hat{u}\hat{y}$ and so the dissipative inequality is satisfied.
- *Back Projection active:* Let's consider

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

  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 $\sigma bs \le \sigma s^2$ and we rewrite the above inequality as

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

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

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

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 is asymptotically stable as was seen in the earlier simulations of section IV. We can experimentally verify the passivity of both broker and server subsystems by introducing a *passivity certificate*

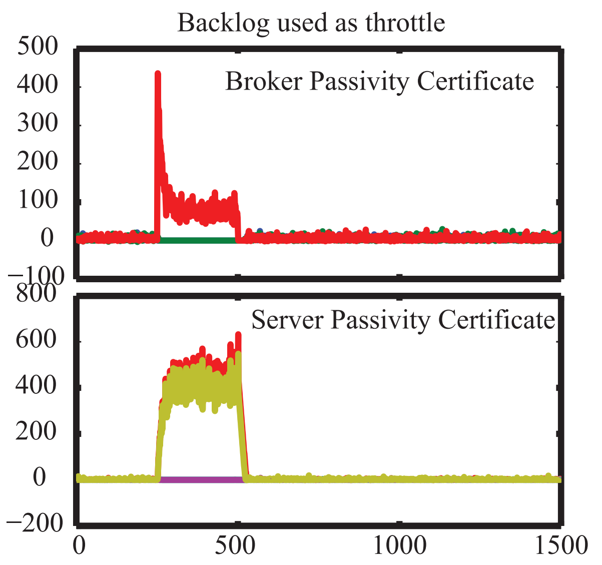
$$C(x, u, y) = u^T y - \Delta V(x) \tag{11}$$

which can be computed on-line. If the system is passive then $C(x(k), u(k), y(k)) \geq 0$ for all k. Figure 7 shows the passivity certificate for two simulation runs. The top two plots in the figure show the passivity certificates when the server transmits the throttle signal $\hat{u} = \sigma(b+s)$. As predicted in propositions 3 and 4, we see that the certificates are positive, which is consistent with the fact that the broker and server systems are passive.

The bottom two plots show the same certificates for the simulation in which the throttling signal was only $\hat{u} = s$. As we saw in section IV, this choice of throttling signal was not stable when the system was in an overload condition. This fact is also seen in figure 7 where the certificate for the server is negative when the system is overloaded. The passivity certificate, therefore, can be used to detect the loss of passivity (with respect to the given certificate) that may be an indicator of unstable operation.

VI. Summary

This paper describes preliminary efforts to develop a passivity framework for the cloud computing systems in which brokers and servers act in a market-oriented manner. The passivity framework may address some of the limitations observed in classical controllers. Passivity tools can be applied to nonlinear systems and do



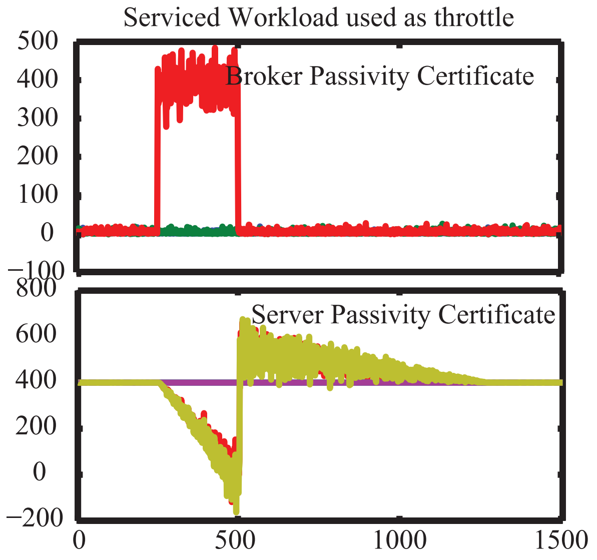


Fig. 7. Passivity Certificates for both Examples Simulations

not require regulation of the system in a small neighborhood of an operating point. Passivity is a compositional system property that provides stability guarantees on ad hoc interconnections of systems. It does not need to have extremely precise system models that are known to some central designer. 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. The passivity certificate in equation (11) provides a tool for detecting when a system may be faulty (i.e. no longer passive). This tool can be used as an on-line monitor to disconnect misbehaving servers or brokers from the network in an automated manner. Finally, preliminary simulation results (not included in this paper) suggest that interconnections of passive brokers and servers will be very tolerant of time-varying changes in the routing topology as well as delays in the throttling feedback signal. This is a potential feature of passivity-based control which should be studied more closely in the future.

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.

[BIW91] C.I. Byrnes, A. Isidori, and J.C. Willems. Passivity, feedback equivalence, and the global stabilization of minimum phase nonlinear systems. *Automatic Control, IEEE Transactions on*, 36(11):1228–1240, 1991.

[Bod56] H.W. Bode. *Network analysis and feedback amplifier design*. Van Nostrand Reinhold, 1956.

[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.

[Gui57] E.A. Guillemin. *Synthesis of passive networks: theory and methods appropriate to the realization and approximation problems*. Wiley, 1957.

[HM76] D. Hill and P. Moylan. The stability of nonlinear dissipative systems. *Automatic Control, IEEE Transactions on*, 21(5):708–711, 1976.

[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.

[KMT98] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[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.

[UKIN10] R. Urgaonkar, U.C. Kozat, K. Igarashi, and M.J. Neely. Dynamic resource allocation and power management in virtualized data centers. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 479–486. IEEE, 2010.

[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.

[WCLK12] X. Wang, M. Chen, C. Lefurgy, and T. Keller. Ship: A scalable hierarchical power control architecture for large-scale data centers. *Parallel and Distributed Systems, IEEE Transactions on*, 23(1):168–176, 2012.

[Wil72] J.C. Willems. Dissipative dynamical systems part i: General theory. *Archive for Rational Mechanics and Analysis*, 45(5):321–351, 1972.

[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.

[WW11] X. Wang and Y. Wang. Coordinating power control and performance management for virtualized server clusters. *Parallel and Distributed Systems, IEEE Transactions on*, 22(2):245–259, 2011.

[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.

[Zam66] G. Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *Automatic Control, IEEE Transactions on*, 11(2):228–238, 1966.

[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.