# A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption

Chiyoung Seo[1] George Edwards[1] Sam Malek[2] Nenad Medvidovic[1]

[1]*Computer Science Department*
*University of Southern California*
*Los Angeles, CA 90089-0781 U.S.A.*
*{cseo, gedwards, neno}@usc.edu*

[2]*Department of Computer Science*
*George Mason University*
*Fairfax, VA 22030-4444 U.S.A.*
*smalek@gmu.edu*

## Abstract

*The selection of an architectural style for a given software system is an important factor in satisfying its quality requirements. In battery-powered environments, such as mobile and pervasive systems, efficiency with respect to energy consumption has increasingly been recognized as an important quality attribute. In this paper, we present a framework that (1) facilitates early estimation of the energy consumption induced by an architectural style in a distributed software system, and (2) consequently enables an engineer to use energy consumption estimates along with other quality attributes in determining the most appropriate style for a given distributed application. We have applied the framework on five distributed systems styles to date, and have evaluated it for precision and accuracy using a particular middleware platform that supports the implementation of those styles. In a large number of application scenarios, our framework exhibited excellent precision, in that it was consistently able to correctly rank the five styles and estimate the relative differences in their energy consumptions. Moreover, the framework has also proven to be accurate: its estimates were within 7% of the different style implementations' actually measured energy consumptions.*

## 1. Introduction

We are witnessing an unrelenting pattern of growth in the size and complexity of software systems. This pattern is especially evident in the emerging class of distributed, mobile, embedded, and pervasive systems. A promising approach to addressing the challenges of developing such systems is to employ the principles of software architectures [6]. Software architectures provide abstractions for representing the structure, behavior, and key properties of a software system [12]. They are described in terms of *software components* (computational elements), *connectors* (interaction elements), and their *configurations* (specific compositions of components and connectors) [8]. Software *architectural styles* (e.g., publish-subscribe, peer-to-peer, client-server) are key design idioms which further refine the vocabulary of components and connectors and propose a set of constraints on how they may be combined in a system.

Architectural decisions made early in the design process are a critical factor in the successful development of a distributed system. In particular, the selection of an appropriate architectural style has a significant impact on various system quality attributes (e.g., latency, scalability, reliability, etc.) of the target system. Different styles are recognized as inducing different quality attributes in software systems. For example, publish-subscribe-based systems are scalable but may not be able to provide performance guarantees; on the other hand, client-server-based systems may be optimized for performance, but can suffer from scalability problems.

Efficient energy usage is increasingly being defined as an important quality attribute for mobile and pervasive applications. However, there are currently no available techniques for analyzing the impact of an architectural style on a system's energy consumption. In fact, unlike other quality attributes, such as those discussed above, a style's energy consumption characteristics are not understood even in such an informal and intuitive manner.

In this paper, we try to address this shortcoming. We present a framework whose objective is to estimate the impact of a distributed software system's architectural style on the system's energy consumption. The framework is intended to be used during architectural design, which would enable an engineer to use energy consumption estimates, along with other quality attributes, in determining the most appropriate style for an application. The energy cost models in our framework are implementation-independent, but we demonstrate how they can be refined and applied to specific middleware platforms that support a chosen set of architectural styles.

We have applied the framework on five distributed systems styles [3,9] to date: client-server, publish-subscribe, peer-to-peer, pipe-and-filter, and C2. We have evaluated the framework for precision and accuracy using a particular middleware platform that supports the implementation of these styles. In a number of distributed application scenarios, our framework exhibited excellent precision, in that it was consistently able to correctly rank the five styles and estimate the relative differences in their energy consumptions. Additionally, the framework has proven to be accurate in the context of our chosen middleware platform: it consistently produced energy consumption estimates that were within 7% of each style implementation's actually measured energy consumption.

0-7695-3092-3/08 $25.00 © 2008 IEEE
DOI 10.1109/WICSA.2008.28

In the remainder of this paper we first describe our estimation framework and detail the energy consumption models for one of the architectural styles considered in our work (Section 2). We then discuss related research (Section 3), and round out the paper with a discussion of some current and planned applications of this research (Section 4).

## 2. Energy Estimation Framework

Fielding [3] and Mehta [9] identified more than twenty common architectural styles for distributed software systems. Among these styles, our work has focused on the client-server, publish-subscribe, C2, peer-to-peer, and pipe-and-filter styles because each of them embodies a diverse set of recurring distributed systems concepts, such as distribution, concurrency, event-based interaction, implicit invocation, layering, remote procedure calls, etc.

In this section we detail the characteristics of the client-server style, and discuss how we model the energy consumption induced by the client-server style on a distributed application. Refer to [11] for the energy consumption models of the remaining four styles that are omitted due to space constraints. Our modeling approach is not limited to the above styles, but can be used for modeling the energy cost induced by an arbitrary style.

Two assumptions underlying our work are that (1) components interact with other components via connectors and (2) connectors can communicate with other connectors in addition to components. These assumptions are common in architectural literature [12], and they do not limit the types of distributed applications to which our framework can be applied. Based on these assumptions, the energy cost of a component $Comp_i$ can be expressed as follows:

$$EC(Comp_i) = E_{logic,i} + E_{commWithConn,i} \qquad \textbf{Eq. 1}$$

In this equation, $E_{logic,i}$ is the computational energy cost of the component $Comp_i$ due to executing its application logic, while $E_{commWithConn,i}$ represents the energy cost of exchanging data with connectors attached to the component.

The energy consumption of a connector $Conn_j$ can be expressed as the following equation:

$$EC(Conn_j) = E_{comm,j} + E_{logic,j} \qquad \textbf{Eq. 2}$$

$E_{comm,j}$ represents the energy consumption of communication, which in a distributed style amounts to the cost of exchanging data locally or remotely. *Communication* is one of the four types of services that a connector may provide [10]. $E_{logic,j}$ represents the energy cost of the other three types of service:

- *Coordination* – A connector may support transfer of execution control among components. Method invocations within a single process and inter-process communication (IPC) are examples of the coordination service.
- *Conversion* – A connector converts the interaction required by one component to that provided by another. Marshalling and unmarshalling data for exchange over the network is an example of the conversion service.
- *Facilitation* – A connector mediates and streamlines component interaction. Connection establishment and a routing facility for delivering a message to its destination are examples of the facilitation service.

Based on this classification, we can calculate $E_{comm,j}$ and $E_{logic,j}$ as follows:

$$E_{comm,j} = E_{commWithComp,j} + E_{remoteComm,j} + E_{localComm,j} \qquad \textbf{Eq. 3}$$

$$E_{logic,j} = E_{coordin,j} + E_{conver,j} + E_{facili,j} \qquad \textbf{Eq. 4}$$

$E_{commWithComp,j}$ represents the energy consumption of exchanging data with the components attached to the connector, while $E_{remoteComm,j}$ and $E_{localComm,j}$ are the energy costs of the connector caused by exchanging data with remote and local connectors, respectively. We should note that if a component and its attached connector run in separate processes, their interactions would be supported by an IPC mechanism, which incurs the energy overhead in both the component and its attached connector.

Once the energy costs of the components and connectors induced by a candidate style for a target distributed system have been calculated, the overall energy consumption resulting from the style can be estimated as follows:

$$overallEC = \left(\sum_{i=1}^{n} EC(Comp_i)\right) + \left(\sum_{j=1}^{m} EC(Conn_j)\right) \qquad \textbf{Eq. 5}$$

where $n$ and $m$ are, respectively, the numbers of the system's constituent components and connectors. In the following section, we discuss how we can model the above energy cost parameters for the client-server style.

In our work, we assume that a component's core business logic (e.g., logic for processing messages received from other components) remains the same across all the candidate styles for a target distributed application. We acknowledge that this logic may need to be refactored in some cases. For example, the logic required by the component for managing its interfaces will change, and may incur additional energy overhead. We distinguish this energy cost from the cost associated with the component's core business logic, and account for it in $E_{commWithConn,i}$ of Equation 1. Consequently, this indicates that the computational energy cost of a component (i.e., $E_{logic,i}$ in Equation 1) remains the same across all candidate styles. Therefore, our framework does not require the actual value of $E_{logic,i}$ while performing the energy consumption comparisons of multiple styles.

### 2.1. Client-Server Style

The client-server style is composed of service-providing (i.e., server) and service-invoking (i.e., client) components. A client is a triggering process; a server is a reactive process. Clients make requests that trigger reactions from servers. Thus, a client initiates activity at times of its choosing, and may block until its request has been serviced. On the

other hand, a server waits for requests to be made and then reacts to them. Figure 1 shows an example of a distributed software system designed according to the client-server style. Connectors in this scenario are implemented as middleware stubs and skeletons. A connector provides each client and server component with an interface for registering and finding remote objects, and sending requests/responses.

Based on this characterization, we can model the energy cost parameters introduced Equations 1-5. First, the energy cost $E_{commWithConn,i}$ on a client $Comp_i$ due to sending requests to and receiving responses from its attached connector can be calculated as follows:

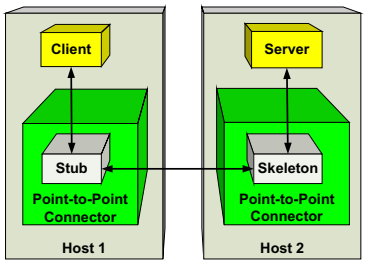


**Figure 1. A distributed Client-Server architecture.**

$$E_{commWithConn,i} = \sum_{k=1}^{p_i} \left( E_{toConn,k} + E_{fromConn,k} \right) \qquad \textbf{Eq. 6}$$

$p_i$ is the total number of requests made by the client. $E_{toConn,k}$ and $E_{fromConn,k}$ represent the energy costs due to sending the $k$th request to and receiving the $k$th response from the connector, respectively. $E_{commWithConn,i}$ on a server $Comp_i$ can be also calculated in the same manner. $E_{toConn}$ and $E_{fromConn}$ depend on the communication mechanism used between a component and its attached connector.

The energy cost $E_{commWithComp,j}$ on a connector $Conn_j$ caused by receiving requests from and sending responses to its attached client component can be calculated as follows:

$$E_{commWithComp,j} = \sum_{l=1}^{q_j} \left( E_{fromComp,l} + E_{toComp,l} \right) \qquad \textbf{Eq. 7}$$

$q_j$ is the total number of requests received from the client component, while $E_{fromComp,l}$ and $E_{toComp,l}$ represent the energy costs due to receiving the $l$th request from and sending the $l$th response to the client. We can also calculate $E_{commWithComp,j}$ on a connector $Conn_j$ attached to a server component by using the above equation.

To model the energy consumption due to transmission of data over a network, we assume that the total energy used is proportional to the size of the data exchanged. This has been shown to be an accurate characterization for commonly used network protocols, including TCP and UDP [2]. Based on this, $E_{remoteComm,j}$ of a connector $Conn_j$ attached to a client due to sending $q_j$ requests and receiving their responses over the network can be estimated as follows:

$$E_{remoteComm,j} = \sum_{l=1}^{q_j} \left( \left( tSize_l \times tEC + tS \right) + \left( rSize_l \times rEC + rS \right) \right) \qquad \textbf{Eq. 8}$$

$tSize_l$ and $rSize_l$ are the sizes (e.g., $KB$) of the $l$th transmitted request and its received response. $tEC$ and $rEC$ are the energy costs ($Joule/byte$) on the connector's host while it transmits and receives a unit of data, respectively. $tS$ and $rS$ represent constant energy overheads associated with channel acquisition [2]. Similarly, we can calculate $E_{remoteComm,j}$ on a connector $Conn_j$ attached to a server using the above equation, where $rSize_l$ and $tSize_l$ are the sizes of the $l$th received request and its transmitted responses.

In the client-server style, local communication has an energy cost that is different from the remote case. The energy cost on a connector $Conn_j$ attached to a client caused by locally sending $q_j$ requests and receiving their responses can be calculated as follows:

$$E_{localComm,j} = \sum_{l=1}^{q_j} \left( E_{localTrans,l} + E_{localReceiv,l} \right) \qquad \textbf{Eq. 9}$$

where $E_{localTrans,l}$ and $E_{localReceiv,l}$ represent the energy costs of sending the $l$th request and receiving its response, respectively. We can also calculate $E_{localComm,j}$ on a connector $Conn_j$ attached to a server using the above equation. $E_{localTrans}$ and $E_{localReceiv}$ depend on the communication mechanism (e.g., IPC, socket, shared memory, queue, etc.) used for implementing the connector.

The coordination cost $E_{coordin,j}$ of a connector $Conn_j$ is captured by $E_{commWithComp,j}$ in the client-server style because when a connector passes a request or response to its attached component, it effectively transfers execution control from client to server (or vice versa).

The conversion cost $E_{conver,j}$ of a connector is incurred by marshalling and unmarshalling requests and responses. Therefore, we can quantify $E_{conver,j}$ as follows:

$$E_{conver,j} = \sum_{l=1}^{q_j} \left( E_{mar,l} + E_{unmar,l} \right) \qquad \textbf{Eq. 10}$$

$E_{mar,l}$ and $E_{unmar,l}$ are the energy costs of marshalling the $l$th request and unmarshalling its response, respectively. $E_{conver,j}$ of a server connector $Conn_j$ can be calculated in an analogous manner.

The facilitation energy cost $E_{facili,j}$ of a connector $Conn_j$ is incurred by establishing connections with local and remote connectors, and can be calculated as follows:

$$E_{facili,j} = \left( E_{remoteConn} \times Num_{remoteConns,j} \right) + \left( E_{localConn} \times Num_{localConns,j} \right) \qquad \textbf{Eq. 11}$$

$E_{remoteConn}$ and $E_{localConn}$ are the constant energy costs due to establishing a single remote or local connection, respectively.

Note that after a server processes a request, it does not always send the response for that request to its client. In this case, the energy costs due to exchanging a response between components and connectors in Equations 6 to 10 will not be incurred. Also note that the energy cost parameters (e.g., $E_{toConn}$, $E_{fromConn}$, $E_{toComp}$, $E_{fromComp}$, $tEC$, $rEC$, etc.) are platform-specific, i.e., their values depend on the hardware, OS, and middleware on which a distributed application is deployed. All of these parameters can be readily obtained

on a target platform [11].

### 2.2. Evaluations

For evaluating our framework, we selected a lightweight, component-based middleware platform called Prism-MW [7] as our implementation platform. Prism-MW provides implementation-level support for realizing architectural elements and various architectural styles, giving us a common platform to evaluate the framework's accuracy in identifying the energy trade-offs between different styles. We chose a version of Prism-MW that runs on top of the JamVM 1.4.5 [5], which is a lightweight JVM. As our target hardware platform, we used a Compaq iPAQ 3800 device running embedded Linux. On the above chosen platform, we measured platform-specific energy cost parameters introduced in Section 2.1 for each of the five styles considered in our work. Interested readers should refer to [11] for details of the measurement steps taken for platform-specific parameters.

We have evaluated our estimation framework for a large number of distributed application scenarios (e.g., sensor, rescue, mobile employee applications, etc.). Our framework was consistently able to correctly rank the five styles and estimate the relative differences in their energy costs. Moreover, the energy cost estimates from our framework were always within 7% of the actual energy costs for all five styles considered in our work. Refer to [11] for more detailed explanations of our evaluation results.

## 3. Related Work

There has been a lot of research on analyzing various quality attributes of software systems at the architectural level. We discuss a couple of representative examples. Wang et al. [13] have evaluated the performance and availability of two software architectural styles. They modeled three real applications (*Unix sort*, *scientific*, and *statistics* programs) into pipe-filter and batch-sequential styles, and compared the two styles with respect to the above two quality attributes. Grahn et al. [4] characterized the performance of three architectural styles by using an event-driven simulation approach. They compared pipe-filter, layered, and blackboard styles with respect to various performance metrics (e.g., throughput, response time, queue time for events, etc.). Neither of the above approaches considered a software system's energy consumption as one of its key quality attributes.

We previously developed the eXtensible Toolchain for Evaluation of Architectural Models (XTEAM), a modeling and analysis framework targeted at distributed, embedded and pervasive software systems [1]. XTEAM leverages the model-driven engineering (MDE) paradigm to provide a reusable infrastructure for facilitating domain-specific architectural analyses and weighing trade-offs among multiple design goals, such as performance, reliability, and resource consumption in terms of memory and energy. However, XTEAM does not support energy cost analysis of a distributed software system with respect to its candidate styles.

## 4. Conclusion

In this paper, we proposed and provided early evaluation of a framework that facilitates the early estimation of the energy consumption induced by an architectural style on a distributed software system. This capability enables an engineer to employ energy cost predictions along with other quality attributes in determining the most appropriate architectural style for a given distributed application. We considered five architectural styles that are commonly used in the design of distributed applications, and evaluated the framework with respect to precision and accuracy for a large number of distributed application scenarios.

We envision several ways to extend, apply, and supplement the work described in this paper. First, we intend to model and evaluate the energy consumption of push-based, pull-based, and hybrid data distribution strategies. Such an evaluation would provide an interesting extension to our style-based energy consumption estimation framework, because several styles (such as peer-to-peer) may be realized using either push- or pull-based strategies. Another important extension to this work is to consider how an intelligent deployment of publish-subscribe connectors, when used alongside sophisticated event filtering mechanisms, can reduce the energy consumption of a distributed system. We believe that the work reported in this paper presents a good starting point for further work in this area.

## 5. References


[1] G. Edwards, et al. Scenario-Driven Dynamic Analysis of Distributed Architectures. In *Proceedings of FASE*, 2007.

[2] L. M. Feeney, et al. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *Proceedings of IEEE INFOCOM*, 2001.

[3] R. Fielding. Architectural Styles and the Design of Network-Based Software Architecture. *Ph.D Thesis*, UCI, June 2000.

[4] H. Grahn, et al. Some Initial Performance Characteristics of Three Architectural Styles. In *Proceedings of WOSP*, 1998.

[5] JamVM 1.4.5. http://jamvm.sourceforge.net/, March, 2007.

[6] E. A. Lee. Embedded Software. *Advances in Computers (Marvin V. Zelkowitz, ed.), Academic Press*, London, 2002.

[7] S. Malek, et al. A Style-Aware Architectural Middleware for Resource Constrained, Distributed Systems. *IEEE Transactions on Software Engineering*, Vol. 31, No. 3, 2005.

[8] N. Medvidovic, et al. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, January 2000.

[9] N. Mehta. Composing Style-Based Software Architectures From Architectural Primitives. *Ph.D Thesis*, USC, 2004.

[10] N. Mehta, et al. Towards a Taxonomy of Software Connectors. In *Proceedings of ICSE*, 2000.

[11] C. Seo, et al. A Framework for Estimating the Impact of a Distributed Software System's Architectural Style on its Energy Consumption. *Tech. Report, USC-CSE-2007-723*, 2007.

[12] M. Shaw and D. Garlan. Software Architecture: Perspectives on an Emerging Discipline. *Prentice Hall*, 1996.

[13] TW. Wang, et al. Software Architectural Analysis - A Case Study. In *Proceedings of COMPSAC*, 1999.