DEVS Peer-to-Peer Protocol for Distributed and Parallel Simulation of Hierarchical and Decomposable DEVS Models

Sunwoo Park
Department of Biopharmaceutical Sciences
University of California, San Francisco
San Francisco, CA 94143-0446, U.S.A.
parks@pharmacy.ucsf.edu

Sean HJ Kim
Joint Graduate Group in Bioengineering
University of California, Berkeley
Berkeley, CA 94720-1762, U.S.A
seanhjk@berkeley.edu

C. Anthony Hunt
Department of Biopharmaceutical Sciences
Joint Graduate Group in Bioengineering
University of California, San Francisco
San Francisco, CA 94143-0446, U.S.A.
a.hunt@ucsf.edu

Dongsun Park
Department of Info. and Comm. Eng.
Chonbuk National University
Jeonju, Chonbuk, 561-756, Korea
dspark@moak.chonbuk.ac.kr

Abstract

We propose a peer-to-peer (P2P) simulation protocol to support large-scale discrete event system specification (DEVS) simulation on distributed and parallel computing infrastructure. The proposed protocol requires less simulation processors and control messages than existing DEVS simulation protocols. It supports parallel output-to-input event translation and routing. The performance of the protocol is not directly bound to the hierarchical structure of a DEVS coupled model. Sequential event translation and structural dependence to the model are major performance issues in existing protocols. With hierarchical model partitioning, the protocol can be used to achieve optimal resource distribution and to obtain performance gain from it. The proposed protocol has been implemented on various DEVS modeling and simulation (M&S) frameworks. In this paper, we introduce DEVS P2P protocol and compare it to existing protocols.

1. Introduction

Simulation has been an alternative approach to traditional mathematical modeling in a wide range of scientific and engineering domains. Simulation is classified into three categories – *continuous*, *discrete time*, and *discrete event* [11]. Recently, discrete event simulation has been receiving more attention than other approaches [7][5]. However, most existing discrete event simulation methods focus on low level simulation time management [4]. To successfully and efficiently describe complex discrete event simulation models, it is desirable to use formal modeling methods that describe temporal aspect of the models and their collaboration with others as concise but descriptive formal specifications. DEVS is one of such methods.

DEVS is a M&S framework designed to model and simulate complex discrete event systems and their temporal activities [11]. There exist two main model specifications: $coupled$ and $atomic$. A coupled model specification describes a multi-scale network of systems and their hierarchical compositions. An atomic model specification describes proactive and reactive temporal activities of a system. Because of its heterogeneous and generic nature such as closure under coupling and model decomposition, DEVS can also handle continuous, discrete time, and hybrid simulation models with its core or extended specifications.

DEVS simulation protocols have been developed to execute DEVS models on various computing infrastructures [1][9][2][10]. A simulator-coordinator protocol requires a simulator and a coordinator for each atomic and coupled components of an associated coupled model [3][8]. It also requires a root coordinator to control simulation. The number of simulation processors increases linearly as the number of components increases. The number of control messages between these processors increases substantially. Simulation performance is bound to the hierarchical structure of the model. A client-server based protocol reduces the computational overhead of the simulator-coordinator protocol by reallocating components to a set of

0-7695-3045-1/07 $25.00 © 2007 IEEE
DOI 10.1109/ISITC.2007.47

computers loosely or tightly connected on distributed and parallel computing infrastructure [2]. However, the protocol does not solve the fundamental problem – dependence to the model structure. In parallel, DEVS bus has been proposed to support interoperability between different DEVS simulation processors [6]. It cannot solve the problem either.

We present a DEVS P2P protocol that improves simulation performance and resource management capability by excluding dependence to the model structure and by minimizing the number of simulation processors and control messages, and network communication. It is expected that the protocol will run DEVS models faster and more efficiently than existing protocols in many situations.

2. DEVS model specifications

A DEVS atomic model represents a discrete event system that reacts to incoming events, performs operations based on its temporal state transitions, and generates outgoing events. A DEVS coupled model represents a network (or aggregation) of DEVS components. A component is either an atomic or a coupled model. The coupled model can contain other coupled models as its components. It allows the coupled model to have a decomposable and hierarchical model structure (Figure 1).

A DEVS atomic model specification is a 8-tuple $\langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, \tau \rangle$. It is divided into three parts – I/O interface, delta and output functions, and time advance. X and Y are sets of input and output events of the model. Each event is described as a pair of a port and a message bag. $\delta_{ext} : Q \times X^b \rightarrow S$ is the external transition function that handles a bag of incoming events and triggers a temporal state transition. Q is a set of temporal states, $S \times T$. For example, $(s_1, e) \rightarrow s_2$ where $s_1, s_2 \in S$ and $e \in T$, and $T \in \mathbb{Z}^+$. $\delta_{int} : S \rightarrow S$ is the internal transition function that changes the current state to another. $\delta_{con} : Q \times X^b \rightarrow S$ is the confluent transition function that serializes the execution order of δ_{ext} and δ_{int} to resolve the race condition when δ_{ext} and δ_{int} occur simultaneously – $\delta_{ext}(\delta_{int}(s), e, x)$ or $\delta_{int}(\delta_{ext}(s, e, x))$. $\lambda : S \rightarrow Y$ generates a set of output events. $\tau : S \rightarrow T$ advances simulation time.

A DEVS coupled model specification is a 6-tuple $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\} \rangle$. It is divided into three parts – I/O interface, a set of components, and coupling relation. X and Y are sets of input and output events of the model. D is a set of component references. $\{M_d\}$ is a set of DEVS models referenced by $d \in D$. Coupling relation describes I/O event propagation and routing between components. For each $d \in D \cup \{self\}$, I_d is a set of references representing influencees of M_d. $self$ is a reserved reference of the coupled model itself. For each $i \in I_d$, $Z_{d,i}$ is a function, d-to-i output translation with (i) $Z_{d,i}$: $X \rightarrow X_i$, if d=$self$; (ii) $Z_{d,i}$: $Y_d \rightarrow Y$, if i=$self$; and (iii) $Z_{d,i}$: $Y_d \rightarrow X_i$, if $d \neq self \wedge i \neq self$.

3. Simulator-coordinator protocol

The *simulator-coordinator* protocol is described by temporal interactions between simulators, coordinators, and a root coordinator [3][8]. A simulator and a coordinator are required for an atomic component and a coupled component, respectively. Additionally, a root coordinator is reserved for the top-most coordinator (Figure 1). The structural organization between simulation processors is homogeneous to the hierarchical structure of an associated coupled model.

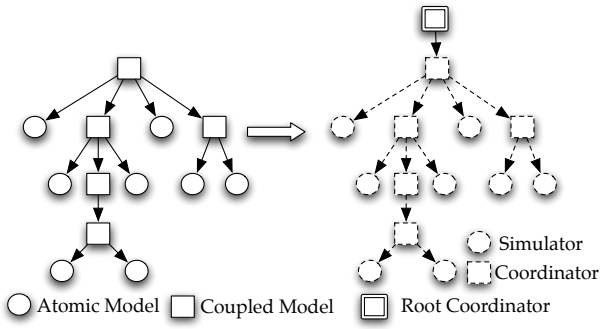

Figure 1. The hierarchical structure of a DEVS coupled model and its mapping to simulation processors in the simulator-coordinator protocol: a simulator or a coordinator is recursively created for each component of the referent coupled model. A root coordinator is also created for the top-most coordinator. The structural organization of simulation processors is identical to the hierarchical structure of the referent model.

For simulation time management, DEVS uses t_L, t, t_N, e, σ, Δ to denote the time the last event occurs, the current simulation time, the time the next event will occur, the elapsed time since t_L, the time left until t_N, and the time between two adjacent event occurences, respectively (Figure 2). $e = t - t_L = (t_N - t_L) - \sigma$, $\sigma = t_N - t = t_N - (t_L + e)$, $t = t_L + e = t_N - \sigma$, and $t_N = t_L + \tau(s) = t + \sigma = t_L + \Delta$ where $t_L \leq t \leq t_N$. Seven control messages – $\#$, $\%$, @, x, y, $*$, and $done$ – are used to control temporal activities of simulation processors and their collaboration.

The root coordinator initiates simulation by sending $\#$ to its children (Figure 3). When each coordinator receives the message from its parent coordinator, it sends $\%$ to its components and awaits until they respond with the same message containing σ. Upon receiving σ from all of its children, the coordinator computes its t_N and sends $done$

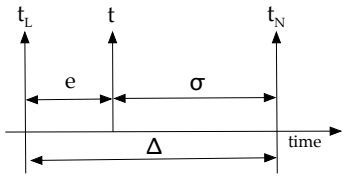

Figure 2. DEVS simulation time: t_L is time of the last event, t is current time, t_N is time of the next event, σ is remaining time to the next event, e is elapsed time since the last event, and Δ is inter-arrival time between two adjacent events.

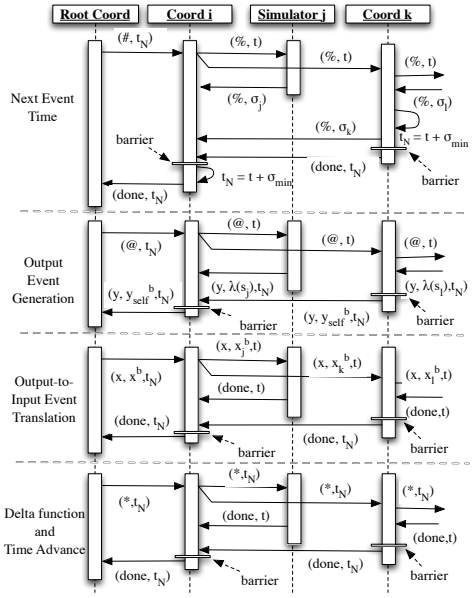

Figure 3. The sequential diagram of the simulator-coordinator protocol: coordinator *i* is comprised of simulator *j* and coordinator *k*. The coordinator *k* contains at least one component.

to the parent. t_N is adjusted to t + σ_{min} where σ_{min} is $min\{\sigma_d | d \in D\}$.

Using DEVS closure under coupling property, we present σ_d by $\tau(s_d) - e_d$ where σ_d, s_d, e_d, and $\tau(s_d)$ are σ, s, e, and $\tau(s)$ of $M_d \in M$. The $done$ is used for a synchronization barrier. When the root coordinator receives $done$ from the top coordinator, it sends @ to its children to trigger them produce output events. Only imminent components of each coordinator IMM can produce the events. IMM is a set of component references that has output events at t + σ_{min}, $\{d | \sigma_d = \tau(s_d) \wedge \lambda_d^b(s_d) \neq \phi\}$. $\lambda_d^b(s_d)$ is the bag of output events produced by M_d. The events are routed to other components based on $\{I_d\}$ and $\{Z_{i,d}\}$. Two control messages x and y are used to route events. To facilitate the process, the coordinator maintains an influencee set INF. INF is a set of component references that have input events, $\{i | i \in I_d, d \in IMM \wedge x_i^b \neq \phi\}$. x_i^b is the bag of input events of M_i that satisfies $\{Z_{d,i}(\lambda_d^b(s_d)) | d \in IMM \wedge I_d\}$. After event routing, the root coordinator sends $*$ to its children. Each coordinator propagates the message to its components. Upon receiving the message, each simulator runs δ function based on the current state, time, and new I/O events. The δ function is one of δ_{int}, δ_{int}, and δ_{con}. Finally, simulation time is advanced by $\tau(s)$. This cycle repeats until any simulation termination condition is met.

4. DEVS P2P simulation protocol

DEVS P2P protocol is described by temporal interactions between $peers$. Each peer advertises itself to others and identifies them with or without an additional peer resolving protocol. The peer is not bound to a particular network middleware. However, a preliminary version of the peer has been implemented on various network middlewares for peer advertisement and identification, communication channel setup, and message exchange [1][9].

A peer is a wrapper for a local simulator (Algorithm 1). It sends output events produced by the simulator to and receives input events from other peers. All simulation control messages are locally exchanged between the peer and the simulator. It eradicates network communication overhead caused by control message exchanges between remote simulation processors in other protocols. Only I/O events are exchanged between peers in DEVS P2P protocol.

Algorithm 1 Peer

```
1:  t_N := 0
2:  advertise itself and identify other peers
3:  do
4:      send(%, t_N) to simulator
5:      wait for(%, t_N) from simulator
6:      send(@, t_N) to simulator
7:      wait for(y, y^b, t_N) from simulator
8:      send y^b to and receive x^b from other peers
9:      send(x, x^b, t_N) to simulator
10:     wait for(done, t_N) from simulator
11:     send(*, t_N) to simulator
12:     wait for(done, t_N) from simulator
13: While t_N ≠ ∞
```

The local simulator is an atomic or coupled simulator. There exists neither a coordinator nor a root coordinator. Each simulator is assigned to a particular component of interest, not to every component (Figure 4). It allows the protocol to decrease the total number of simulation processors and simulation control messages, and at the same time increase resource manageability. The coupled simulator directly executes a coupled component (Algorithm 3). Computational overhead exhibited during output-to-input event translation and routing in existing protocols is significantly reduced in P2P protocol. Each simulator handles event

translation and routing related to itself in both distributed and parallel manners. It increases the concurrency and performance of the process.

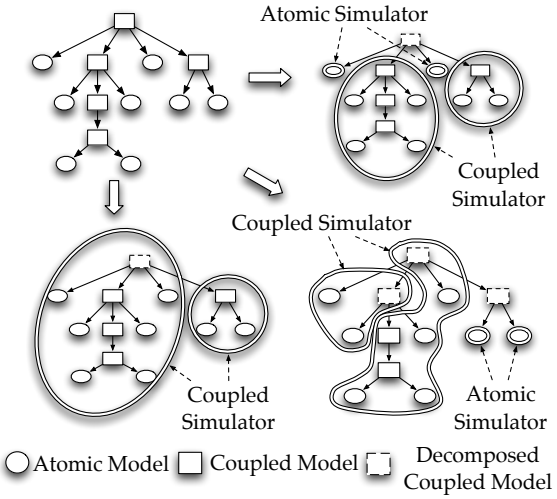

Figure 4. Flexibility of simulation processor mapping in DEVS P2P protocol: a simulator is mapped to a particular component of interest. With hierarchical model partitioning, it can achieve optimal resource distribution and obtain performance gain from it.

Algorithm 2 Atomic Simulator

$t_L := e := 0$; $t_N := \tau(s)$; $x^b := y^b := \phi$
when $(\%, t)$ injected by peer
  send(%,$\tau(s)$ - (t - t_L)) to peer
end when
when $(@, t)$ injected by peer
  send $(y, ((t = t_N)\ ?\ \lambda(s) : \phi), t)$ to peer
end when
when (x, x_d^b, t) injected by peer
  $x^b := x^b \oplus x_d^b$
  send $(done, t)$ to peer
end when
when $(*, t)$ injected by peer
  if $t < t_L \vee t > t_N$ **then** timing synchronization error
  else
    $e :=$ t - t_L
    if $t_L \leq t \leq t_N$ **then**
      $s := (x^b \neq \phi)\ ?\ \delta_{ext}(s, e, x^b) : s$
    else if t = t_N **then**
      $s := (x^b = \phi)\ ?\ \delta_{int}(s) : \delta_{conf}(s, e, x^b)$
    else raise error
    end if
    $t_L := t$; $t_N :=$ t_L + $\tau(s)$; $x^b := \phi$
    send $(done, t_N)$ to peer
  end if
end when

5. Analytical comparison results

Two comparison measures – the number of simulation processors needed for simulation (η_p) and the number of control messages produced during simulation (η_m) – were used to compare DEVS P2P protocol to simulator-coordinator protocol. Hierarchical coupled models were characterized by $T(d, k, n)$. d is the depth of the coupled model, k is the number of children per coupled component, and n is the total number of atomic components of the coupled model being generated. $T(d, k, k^d)$ is used for all analysis and experiments.

Algorithm 3 Coupled Simulator

$t_L := e := 0$; $t_N := \sigma_{min} := min\{\sigma_d | d \in D\}$
$IMM := INF := \phi$
$x_d^b := y_d^b := \phi\ \forall d \in D \oplus self$
when $(\%, t)$ injected by peer ▷ find σ_{min}
  send(%, $min\{\sigma_d | d \in D\}$) to peer
end when
when $(@, t)$ injected by peer ▷ generate output events
  $y_{self}^b := \phi$; $x_d^b := \phi, \forall d \in D$
  for each $d \in D$ **do**
    $IMM := IMM \oplus d$
    $y_d^b := (t = t_N)\ ?\ \lambda(s_d) : \phi$
    for each $i \in I_d \backslash self$ **do** $x_i^b := x_i^b \oplus z_{d,i}(y_d^b)$ **end for**
    if $self \in I_d$ **then** $y_{self}^b := y_{self}^b \oplus z_{d,self}(y_d^b)$**end if**
  end for
  send(y,y_{self}^b, t) to peer
end when
when (x, x^b, t) injected by peer ▷ prepare input events
  for each $i \in I_{self} \wedge x^b \neq \phi$ **do**
    $INF := INF \oplus d$
    $x_i^b := x_i^b \oplus z_{self,i}(x^b)$
  end for
  send $(done, t_N)$ to peer
end when
when $(*, t)$ injected by peer ▷ run δ function
  $e :=$ t - t_L
  if $t_L \leq t \leq t_N$ **then**
    for each $d \in IMM \oplus INF$ **do**
      if $d \in IMM \wedge INF$ **then** $s_d := \delta_{conf}(s_d, e, x_d^b)$
      else if $d \in IMM$ **then** $s_d := \delta_{int}(s_d)$
      else if $d \in INF$ **then** $s_d := \delta_{ext}(s_d, e, x_d^b)$
      end if
    end for
    $IMM :=INF := \phi$
  else raise an error
  end if
  $t_L := t$; $t_N :=$ t_L + $\tau(s_d)$; $x_d^b := \phi$
  send $(done, t_N)$ to peer
end when

η_p corresponds to $\sum_{i=0}^{d} k^i$ which is a polynomial of degree d in simulator-coordinator protocol. There exist one root coordinator, $\sum_{i=1}^{d-1} k^i$ coordinators, and k^d simulators. η_p is bound to $\Theta(k^d)$ because $\sum_{i=0}^{d} k^i = \frac{k^{d+1}-1}{k-1} \approx k^d$. Meanwhile, η_p is k^{d-1}, k^d, and $k^d - k + 1$ when (i) only coupled simulators, (ii) only atomic simulators, and (iii) both coupled and atomic simulators, respectively, are used in the worst case analysis of DEVS P2P protocol. η_p is bound to $O(k^{d-1})$, $O(k^d)$, and $O(k^d)$, respectively. In the average analysis, η_p is $\sum_{i=0}^{d-1} p(k^i)k^i, d \geq 1$ where $p(k^i)$ is the probability that k^i coupled simulators exist in the case of (i). η_p is $\sum_{i=1}^{d-1} \sum_{j=1}^{k^i-1} p(\eta_p(i,j))\eta_p(i,j)$ where $\eta_p(i,j)$ is $(k^i - j) + k^{d-i}(j-1)$ in the case of (iii). $p(\eta_p(i,j))$ is the probability that $k^i - j$ coupled simulators and $k^{d-i}(j-1)$ atomic simulators exist when i is the tree depth and j is the

number of coupled simulators at the depth. Both analytical and empirical studies (unpublished data) showed that η_p in DEVS P2P protocol is significantly less than that of simulator-coordinator protocol.

η_m is $\frac{1}{k-1}\left(8k^{d+1}+k^d-9\right)$ per simulation cycle in simulator-coordinator protocol. Specifically, one $\#$ message, $\sum_{i=0}^{d} 2k^i - 1$ $\%$ messages, and $\sum_{i=0}^{d-1} k^i$ $done$ messages were generated to find σ_{min}; $\sum_{i=0}^{d} k^i$ @ messages and $\sum_{i=0}^{d} k^i$ y messages to produce output event messages; $\sum_{i=0}^{d} k^i$ x messages and $\sum_{i=0}^{d} k^i$ $done$ messages to perform d-to-i event translation and routing; and $\sum_{i=0}^{d} k^i$ $*$ messages and $\sum_{i=0}^{d} k^i$ $done$ messages to perform δ function and advance time. Meanwhile, η_m is $7 \times k^{d-1}$, $7 \times k^d$, and $7 \times (k^d - k + 1)$, respectively, in the case of (i), (ii), and (iii) in the worst case analysis of DEVS P2P protocol. Each simulator generates 5 messages per simulation cycle: one $\%$ message to identify σ_{min}; one @ message and one y message to produce output events; one x message to d-to-i event translation and routing; and one $*$ message to perform δ function and advance time. Each peer requires two extra messages to exchange its output with others. In the average case analysis, η_m is $7 \times \sum_{i=0}^{d-1} p(k^i)k^i, d \geq 1$ in the case of (i). η_m is $\sum_{i=1}^{d-1}\sum_{j=1}^{k^i-1} p(\eta_p(i,j))\eta_p(i,j)$ in the case of (iii). These analytical results indicate that DEVS P2P protocol requires substantially less number of control messages than in simulator-coordinator protocol in both the worst case and the average case analyses.

DEVS P2P protocol eradicates unnecessary network communication due to control message exchanges between remote simulation processors. Each peer requires only two network messages to exchange output events with others. All control messages are locally exchanged between the peer and its simulator. However, coordinators and simulators in simulator-coordinator protocol exchanges all or most of control messages, and I/O events with others using network communication channels.

6. Summary and conclusion

We described DEVS P2P simulation protocol and compared it to existing DEVS simulation protocols. In large-scale simulation, the proposed protocol considerably reduces the number of simulation processors needed for simulation and the total number of control messages generated during simulation. By wrapping a simulator into a peer and allowing only that peer to handle network communication, the protocol eradicates unnecessary network communication and achieves better resource manageability in distributed and parallel computing infrastructure. Parallel event translation improves I/O event routing between peers. We also presented analytical results that compare DEVS P2P protocol with simulator-coordinator protocol. The comparison results showed that the proposed protocol provides higher degrees of scalability, efficiency, and resource manageability than existing protocols in large-scale distributed and parallel simulation.

7. Acknowledgments

This research was funded in part by CDH Research Foundation, Postdoctoral Fellowship provided to SP by CDH R.F., and Graduate Fellowship to SHJK from IFER. We thank other members of the BioSystems Group for helpful discussion and commentary.

References

[1] S. Cheon, C. Seo, S. Park, and B. P. Zeigler. Design and implementation of distributed DEVS simulation in a peer-to-peer network system. In *2004 Advanced Simulation Technologies Conference*, Arlington, VA, USA, 2004.

[2] Y. K. Cho, X. Hu, and B. P. Zeigler. The RTDEVS/CORBA environment for simulation-based design of distributed real-time systems. *Simulation*, 79(4):197–210, 2003.

[3] A. C. H. Chow and B. P. Zeigler. Parallel DEVS : A parallel, hierarchical, modular modeling formalism and its distibuted simulator. *Transaction of the Society for Computer Simulation International*, 13(2):55–67, 1996.

[4] R. M. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53, 1990.

[5] F. J. Kaudel. A literature survey on distributed discrete event simulation. *ACM SIGSIM Simulation Digest*, 18(2):11–21, 1987.

[6] Y. J. Kim, J. H. Kim, and T. G. Kim. Heterogeneous simulation framework using DEVS BUS. *Simulation*, 79(1):3–18, 2003.

[7] J. Misra. Distributed discrete-event simulation. *ACM Computing Survey*, 18(1):39–65, 1986.

[8] S. Park, S. H. Kim, and C. A. Hunt. Simulation processors and their collaboration for DEVS models integrated with universal coupling specification. In *International Conference on Computer Applications in Industry and Engineering*, San Francisco, CA, 2007.

[9] C. Seo, S. Park, B. Kim, S. Cheon, and B. P. Zeigler. Implementation of distributed high-performance DEVS simulation framework in the grid computing environment. In *2004 Advanced Simulation Technologies Conference*, Arlington, VA, USA, 2004.

[10] B. P. Zeigler, S. B. Hall, and H. S. Sarjoughian. Exploiting HLA and DEVS to promote interoperability and reuse in lockheed's corporate environment. *Simulation*, 73(5):288–295, 1999.

[11] B. P. Zeigler, T. G. Kim, and H. Praehofer. *Theory of Modeling and Simulation*. 2nd edition. Academic Press, 2000.