14

Analyzing performance bottlenecks in protocols based on finite state specifications

Sijian Zhang
P.O.Box 30004, 8602 Granville St., Vancouver, BC, Canada V6P 5A0
(email: <sijian@unixg.ubc.ca>)

Samuel T. Chanson
Dept. of Computer Science, HongKong Univ. of Science and Technology
Clear Water, Hong Kong (email: <chanson@cs.ust.hk>)

Abstract

This paper studies the problem of identifying performance bottlenecks in communication protocols. The model used is a Finite State Machine extended with time and transition probabilities known as PEFSM. A definition of PEFSM is given and the bottleneck identification methods proposed are based on this performance model. Informally, a bottleneck with respect to a performance metric is defined as the transition among all the transitions in a PEFSM which would produce the largest marginal improvement of the performance metric if the time of the transitions were reduced by the same small amount. We present two methods to locate the bottleneck transitions with respect to two of the most important performance metrics, i.e., *throughput* and *queue wait time*. These methods are partially validated by simulation.

Keywords

Finite state, specification, performance, bottleneck

Testing of Communicating Systems, M. Kim, S. Kang & K. Hong (Eds)
Published by Chapman & Hall

1 INTRODUCTION

Performance bottlenecks exist in almost all computer systems in various forms. System designers, managers, analysts and users have worked on identifying the performance bottlenecks in a computer system for a long time. A bottleneck can be a service center of a system [Leung88, Allen90] or, at a more abstract level, a system parameter. For instance, in [ZiEt92], the *sensitivities* of the parameters in the throughput expression are used to determine the throughput bottleneck.

There exist many definitions for performance bottlenecks and most of them are defined with respect to only *throughput* or *utilization* [Ferr78, Lock84, Leung88, Yang89, Allen90, ZiEt92]. Nevertheless, all definitions of the performance bottlenecks have a common characteristic: a bottleneck identifies the component[1] in the system which has the most significant impact on system performance. A small improvement to the bottleneck component can greatly improve system response time, throughput or utilization.

This paper is concerned with finding performance bottlenecks in communication protocols. We note that it is common to specify communication protocols as interacting Finite State Machines (FSMs) or Extended FSMs (EFSMs) which are FSMs extended with variables. Many standardized protocols are directly given as FSMs or EFSMs. Examples can be found in [Tane88, ISO2576, ISO7776]. At least two internationally standardized formal description techniques exist (Estelle [ISO8807] and SDL [SDL]) which provide a way of specifying protocols and distributed systems based on FSMs or EFSMs. Therefore, it is reasonable to define a performance model based on FSM or EFSM for use in performance analyses as well as bottleneck identification. In the following section, we shall call such a performance model as *performance extended FSM* (PEFSM).

The PEFSM is essentially an FSM enhanced with time and transition probabilities. In the FSM of a PEFSM, *state* and *transition* are the two main constructs. *States* are conceptual while *transitions* have direct correspondence in the implementation of the protocol specified by the FSM. The execution time of a transition directly affects performance. Therefore, it is natural to transform the bottleneck detection problem to the one of identifying the *bottleneck transition* in a PEFSM. This is useful because once the bottleneck transition is identified, we know where we should focus our efforts in improving system performance.

The execution of a transition takes non-zero time. In our model, each transition is associated with a class of incoming messages (i.e., message type). Futhermore, because of causal relationship, the transition service time affects the subsequent messages. Reducing the transition time in a PEFSM will im-

[1] A "component" can be a hardware device, a software module or a system parameter as mentioned earlier.

prove the overall performance of the PEFSM. For example, reducing a transition time will increase the throughput of each class of outgoing messages since the recurrence times of each state are decreased. However, the degree of improvement to a performance metric depends on the selected transitions. The one which results in the most improvement with respect to a performance metric is called the *bottleneck* of this performance metric.

We use the concept of *weight* to indicate the relation between the reduction of transition time and the improvement of a performance metric. A weight with respect to a performance metric is computed for each transition in a PEFSM. The higher the weight of a transition with respect to a performance metric, the more the performance metric can be improved by reducing the service (or execution) time of this transition. As such, the transition with the *greatest weight* with respect to a performance metric is the *bottleneck transition* (with respect to the performance metric). For instance, if the weight of transition ¡*i*,*j*¿ (from state *i* to state *j*) in a PEFSM with respect to the *mean queue wait time* of a message class is greater than that of any other transition, then transition ¡*i*,*j*¿ is the bottleneck transition with respect to the mean queue wait time. In other words, if each transition time is independently reduced by the same amount, the mean queue wait time will decrease the most in the case of a reduction in transition ¡*i*,*j*¿.

In this paper, we focus on two of the most important performance metrics of a PEFSM. They are the *throughput rate* of a class of outgoing messages and the *mean queue wait time* of a class of incoming messages. The methods to compute the *weights* of transitions with respect to each performance metric are also discussed.

The first method is to use *partial derivatives*. In general, if a performance metric $\mathcal{K}$ can be expressed as a function of a set of parameters, $t_1, t_2, \cdots, t_n$,

$$\mathcal{K} = \mathcal{F}(t_1, t_2, \cdots, t_n),$$

and the derivatives of $\mathcal{K}$ with respect to $t_1, t_2, \cdots, t_n$ exist, then the partial derivatives

$$\frac{\partial \mathcal{K}}{\partial t_1}, \frac{\partial \mathcal{K}}{\partial t_2}, \cdots, \frac{\partial \mathcal{K}}{\partial t_n},$$

indicate the relative impact of the change of each parameter on the performance metric. Therefore, the partial derivatives can be used as the weights of the parameters. In our studies, we compute the partial derivatives of the throughput of outgoing messages of a specific class with respect to each transition time. These derivatives are taken as the weights of the transitions with respect to throughput.

The second method is to use an approximation technique to compute the weights of transitions with respect to the mean queue wait time of a specific incoming message class. This method is useful in the case where the computation of partial derivatives is difficult.

The rest of the paper is organized as follows. Section 2 gives the definition of the performance model PEFSM. Section 3 presents a method to locate the bottleneck transition of the throughput rate of a class of outgoing messages. Section 4 presents a different method to compute the weights for each transition with respect to the mean queue wait time of a class of incoming messages. The method is partially validated by simulation. Section 5 discusses related work on defining and locating performance bottlenecks, and Section 6 summarizes this paper.

2 PERFORMANCE MODEL

The detailed definition of PEFSM can be found in [Zhang95]. Due to space limitation, only a brief description of PEFSM is given in this paper. Since a PEFSM contains an embedded FSM, we start with the definition of the FSM.

2.1 FSM

A finite state machine (FSM) which describes a protocol entity is formally defined as a six-tuple

$$M = (Q, I, O, \delta, \xi, q_0) \tag{1}$$

where Q – a finite set denoting *states*;
I – a finite set denoting *incoming message classes*;
O – a finite set denoting *outgoing message classes*;
δ – a function denoting *transitions*, i.e., $\delta : Q \times I \rightarrow Q$;
ξ – a function denoting *transition outputs*, i.e., $\xi : Q \times I \rightarrow O$;
q_0 – an initial state.

Note that an FSM of a communication protocol does not necessarily have any final state. This is because a protocol (such as that in the telephone system) can execute forever without termination.

2.2 PEFSM

During execution, the FSM of a protocol changes from state to state. The state changing process is a stochastic process. Our performance model is a model that describes this stochastic process.

We enhance the FSM with *time* and *probability* to define the performance model which is called the *performance extended FSM* (PEFSM). Each transition in the PEFSM is associated with a *transition time* and a *single-step transition probablity*. The transition time from state i to j is denoted as $\psi_{\tau_{ij}}$ which is the time period from the start to the completion of the transition. The single-step probability of the transition from state i to j is denoted as p_{ij} which is the probability that transition ¡i,j¿ will be executed when the PEFSM is in state i.

Formally, a **PEFSM**, denoted as Ψ, is defined as a pair

$$\Psi = (M, \mathcal{P}) \tag{2}$$

where $M = (Q, I, O, \delta, \xi, q_0)$ is the kernel which is an FSM whose formal definition is given in (1), and $\mathcal{P} = (\mathbf{P}, {}^{\psi}\mathbf{H})$ is the running environment expressed in terms of time and probability:

$\mathbf{P} = [p_{ij}]$ – a matrix of the single-step transition probabilities;
${}^{\psi}\mathbf{H} = [{}^{\psi}h_{ij}(t)]$ – a matrix of the probability density functions (p.d.f.s) of the transition times.

In a PEFSM, M, $\mathbf{P}$ and ${}^{\psi}\mathbf{H}$ are primitive data. They are assumed to be provided directly by the performance evaluator.

Let $\{X(t), t \geq 0\}$ be the state changing process of the PEFSM;
$t_0, t_1, ..., t_n, ...$ be the sequence of the epochs right before the processing of a transition is completed;
$X_0, X_1, ..., X_n, ...$ be the sequence of the states in the PEFSM corresponding to the time sequence $t_0, t_1, t_2, ..., t_n, ...$, respectively.

The components of a PEFSM and their relationships are formally defined in the following:

1. $X(t) \in Q$ for all $t \geq 0$.
2. $X(0) = X_0 = q_0$.
3. $X_n = X(t_n^-)$ [2] and $X_{n+1} = X(t_n)$.
4. $Pr\{X_{n+1} = j | X_n = i\} = p_{ij}$.
5. If $X_n = i$ and $X_{n+1} = j$, then $t_n - t_{n-1} = {}^{\psi}\tau_{ij}$.
6. ${}^{\psi}h_{ij}(t)$ is the p.d.f. of ${}^{\psi}\tau_{ij}$, i.e. $Pr\{{}^{\psi}\tau_{ij} = t | X_n = i,\ X_{n+1} = j\} = {}^{\psi}h_{ij}(t)$.

From the above definitions, it can be seen that the trajectory of the state variable X of a PEFSM is governed by M, $\mathbf{P}$ and ${}^{\psi}\mathbf{H}$. M determines the state space of X and the possible next value of X statically. The other parameters govern the dynamic control of X.

We assume that the transition probability matrix $\mathbf{P}$ is known. When the stochastic process of state changing is ergodic[3] the steady-state state probability vector, π, can be computed from $\mathbf{P}$ by solving the matrix equation (see [Zhang95] for details):

$$\pi\mathbf{P} = \pi.$$

---

[2] t_n^- denotes the epoch right before the time instant t_n. $t_n^- < t_n$ but t_n^- is infinitely close to t_n.

[3] A stochastic process is *ergodic* when it is recurrent non-null and irreducible [Allen90].

3 THROUGHPUT BOTTLENECK

In a PEFSM, an outgoing message is generated when a transition is processed. Therefore, the throughput rate of the outgoing messages of a class is equal to the recurrence rate of the transition associated with the class.

Let ${}^{\psi}\bar{e}_i$ and ${}^{\psi}\bar{e}_{ij}$ be the mean recurrence rates of state i and transition ¡i,j¿ of a PEFSM, respectively. Their relationship can be expressed as

$$ {}^{\psi}\bar{e}_{ij} = {}^{\psi}\bar{e}_i \cdot p_{ij} $$

where p_{ij} is the probability of transition ¡i,j¿.

When the PEFSM is in equilibrium, we have (see [Howa71](p.725))

$$ {}^{\psi}\bar{e}_i = \frac{\pi_i}{\psi\bar{\tau}}. $$

Therefore,

$$ {}^{\psi}\bar{e}_{ij} = \frac{\pi_i}{\psi\bar{\tau}} \cdot p_{ij} = \frac{\pi_i p_{ij}}{\sum_{u,v \in Q} \pi_v p_{uv} \, {}^{\psi}\bar{\tau}_{uv}}. $$

Let $\bar{\eta}_{ij}$ be the throughput rate of the class ij outgoing messages. Since class ij outgoing messages are associated with transition ¡i,j¿,

$$ \bar{\eta}_{ij} = {}^{\psi}\bar{e}_{ij} = \frac{\pi_i p_{ij}}{\sum_{u,v \in Q} \pi_u p_{uv} \, {}^{\psi}\bar{\tau}_{uv}}. $$

The above equation gives the relationship of the throughput rate $\bar{\eta}_{ij}$ and the transition times ${}^{\psi}\bar{\tau}_{uv}$ $(u, v \in Q)$. For a given PEFSM, $\pi_i p_{ij}$ $(i, j \in Q)$ is fixed, so the change of $\bar{\eta}_{ij}$ varies inversely with the change in the value of the denominator of the above equation.

Using the derivatives, one can determine that the coefficient $\pi_v p_{uv}$ of ${}^{\psi}\bar{\tau}_{uv}$ in the denominator indicates the relative degree of the improvement on $\bar{\eta}_{ij}$ by transition uv $(u, v \in Q)$ compared to the other transitions. $\pi_v p_{uv}$ in fact is the steady-state probability of transition uv. Among the transitions, the one with the largest steady-state transition probability has the greatest impact on increasing the throughput rate $\bar{\eta}_{ij}$.

Therefore we define $\pi_v p_{uv}$ as the *weight* of the transition uv with respect to $\bar{\eta}_{ij}$. The bottleneck transition of $\bar{\eta}_{ij}$ is the transition which has the largest $\pi_v p_{uv}$ $(u, v \in Q)$.

Since $\pi_v p_{uv}$ $(u, v \in Q)$ is not related to $\bar{\eta}_{ij}$, we can further conclude that the bottleneck transition is the same for the throughput rates of all classes of outgoing messages.

4 QUEUE WAIT TIME BOTTLENECK

We say that an incoming message is a *firable* message of state i if it is associated with a transition starting from state i. If the PEFSM is not in state i

when a firable message of state i arrives, the message will be stored in a queue. We shall assume that there is a *first in first out* (FIFO) queue for each class of incoming messages. We are interested in identifying the bottleneck transition with respect to the mean queue wait time of a specific class of incoming messages.

It has been proved that [Zhang95] the mean queue wait times of different classes of firable messages of a state are the same, i.e.,

$$\overline{W}_{ij} = \overline{W}_i \quad (j \in Q)$$

if class ij (for all $j \in Q$) of messages arrive independently and in a Poisson pattern. So it is necessary only to compute the *overall* mean queue wait time of all classes of firable messages of a state in this case.

To compute the overall mean queue wait time $\overline{W}_i$, we construct the *virtual jobs* of state i and treat the queuing system of the PEFSM as an M/G/1 system. The virtual jobs of state i are the sequences of transitions where each sequence forms a *first passage* from state i to i in the PEFSM. The mean queue wait time can then be computed by applying the well-known solution technique for M/G/1:

$$\overline{W}_i = \frac{\lambda_i \overline{\zeta_i^2}}{2(1-\rho_i)} = \frac{\sum_{j \in Q} \lambda_{ij} \overline{\zeta_{ij}^2}}{2(1-\rho_i)}.$$

In the above equation, $\overline{\zeta_{ij}^2}$ is the second moment of the service time of class ij virtual jobs. A set of equations has to be solved in order to compute $\overline{\zeta_{ij}^2}$ $(i, j \in Q)$ (see [Zhang95] for details). The closed-form solution of $\overline{\zeta_{ij}^2}$ is difficult to obtain. So it is generally infeasible to compute the partial derivatives of $\overline{W}_i$ with respect to each transition time $\bar{\tau}_{uv}$ $(u, v \in Q)$ for use as the *weights* to identify the bottleneck transition of $\overline{W}_i$. Therefore, the following approximate solution is proposed instead.

4.1 An approximation approach

As mentioned earlier, the FSM of a PEFSM contains information on the service order of incoming messages. This ordering affects the performance of the PEFSM and should be taken into consideration to obtain more accurate results.

In general, we can assume the queuing system of a PEFSM to consist of a single server with a single queue. Figure 1 shows a queuing system which serves a PEFSM. The service order of incoming messages is controlled by the FSM of the PEFSM and the incoming messages.

An asynchronous incoming message to a PEFSM may arrive before the PEFSM is ready to process this message. When this happens, the message will have to wait in a queue. The *queue wait time* of this message is the

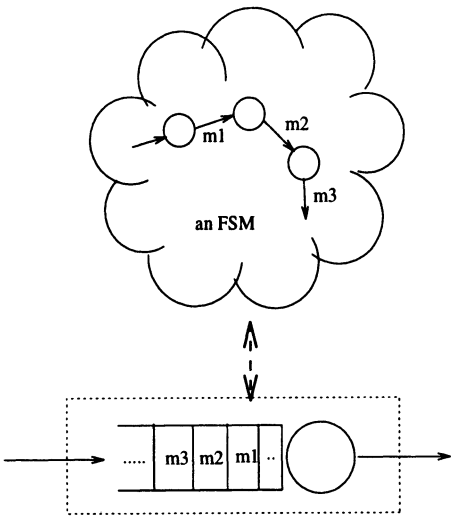


Figure 1: Service order implied by the FSM of a PEFSM.

elapsed time between the moment it arrives and the moment it is processed. From Figure 1, it is not difficult to see that the waiting period of this message includes not only the processing times of all the messages of the same class which arrived earlier that are still in the system, but also the processing times of the transitions associated with the messages of the other classes. These transitions bring the PEFSM to the right state so that the target message can be processed. For example, in Figure 1, message $m3$ has to wait for service until the transitions associated with $m1$ and $m2$ are processed.

Before we show how the incoming messages of a class in a PEFSM wait statistically when they arrive early, the definitions of *transition path* and *transition subpath* are first given.

Definition 4.1 (transition path) *A* transition path *of a PEFSM is a sequence of consecutive transitions in the PEFSM.*

Definition 4.2 (transition subpath ij) *A transition subpath ij of a PEFSM is a finite number of consecutive transitions in the PEFSM starting from state i and ending in state j. The first transition of the subpath is called the head of the subpath; the last transition is called the tail of the subpath.*

Figure 2 shows a state-transition tree of a PEFSM. Each state in the tree is a state in the FSM of the PEFSM, and each transition is a transition in the FSM. This tree includes all the possible transition subpaths to transition $¡^{i,j}¿$.

Suppose γ is an incoming message of class ij of the PEFSM and W_{ij} ($W_{ij} > 0$) is the queue wait time of γ. Furthermore, suppose transition subpath 1 in Figure 2 includes the transitions which must be processed before message γ.

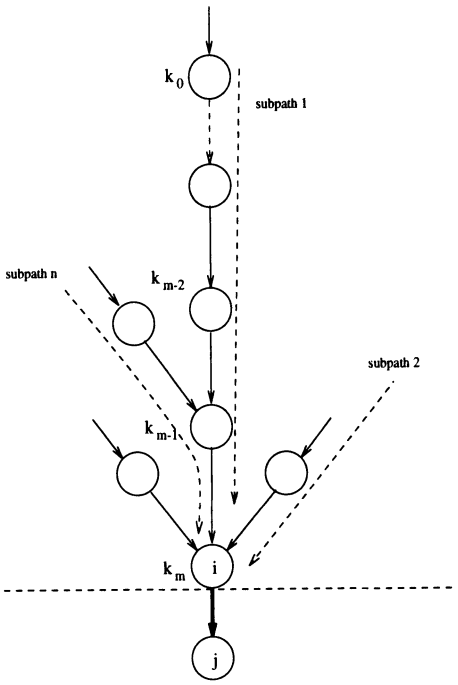


Figure 2: The subpaths to transition ¡i,j¿ in a PEFSM.

Assume the PEFSM is in state k_0 when message γ arrives. Then, the transition subpath $k_0 i$ includes the transitions which are seen by γ and will be processed before γ. Let these transitions be transitions k_0k_1, k_1k_2, $\cdots$, $k_{m-1}k_m$ ($k_m = i$), and D_1, D_2, D_3, $\cdots$, D_m be their transition times, respectively.

By definition, we have

$$\sum_{n=2}^{m} D_n < W_{ij} \leq \sum_{n=1}^{m} D_n. \tag{3}$$

Transition k_0k_1 may already be in progress at the time γ arrives, in this case $W_{ij} \leq \sum_{n=1}^{m} D_n$.

γ will have to wait until the processing of all of these transitions is finished whether or not the incoming messages associated with them have already arrived. The decrease of the transition time of any of these transitions will reduce the queue wait time of γ, W_{ij}. Those transitions that appear in the transition subpath $k_0 i$ more than once will have a higher impact in reducing W_{ij}.

However, different messages of class ij may see different transition subpaths when they arrive. Furthermore, a transition may appear in more than one transition subpath. So the relative frequency of each transition subpath

seen by γ should be taken into account in computing the *weights* for all the transitions with respect to W_{ij}. The relative frequency of a subpath can be computed using the *transition subpath probability* defined below.

Definition 4.3 (transition subpath probability) *The* probability *of a transition subpath $k_0 k_m$ is defined as:*

$$Pr\{subpath\ k_0 k_m\} = \prod_{n=1}^{m} p'_{k_{n-1}k_n}$$

where transition $k_{n-1}k_n$ $(n = 1, 2, ..., m)$ are the transitions in the subpath and $p'_{k_{n-1}k_n} = \pi_{k_{n-1}} p_{k_{n-1}k_n}$ is the steady-state probability of transition $k_{n-1}k_n$, and $p_{k_{n-1}k_n}$ is the single-step probability of transition $k_{n-1}k_n$.

Given the probabilities of transition subpaths, we can compute the relative frequency of a transition seen by a specific class of incoming messages. The frequency is simply the sum of the probabilities that this transition appears in all the possible transition subpaths which satisfy Inequality (3). It is useful to compute the relative frequency of a transition because decreasing the time of the transition with the highest frequency by the same amount will reduce the mean queue wait time of the specific class the most. Therefore, in this case, the frequency can be used as the *weight* in identifying the bottleneck transition of the mean queue wait time of the incoming messages of the specific class.

Let w_{uv} be the weight of transition uv. From the discussion above, we can define

$$w_{uv} = \sum_{l \in subpaths} (Pr\{\text{subpath } l\} \cdot Pr\{\text{transition } uv \text{ appears in the subpath } l\}).$$

Next, we present an algorithm to compute the weights given the mean queue wait time of a class of incoming messages.

4.2 Computation of weights

Assume that the single-step transition probabilities in **P** of a PEFSM are given, and the steady-state state probabilities π as well as the transition times of each transition have been computed.

Suppose $\overline{W}_{ij}$ is known either by computation or measurement. An algorithm to compute the weights with respect to $\overline{W}_{ij}$ is given in Figure 3.

Procedure 1 of the algorithm initializes all the weights to zero before calling Procedure 2. Procedure 2 computes the weights of all the transitions in the PEFSM. Using a recursive procedure, it traverses all the transition subpaths which end in state i and satisfy Inequality (3). The subpath starts backwards from state i. A transition is added to the current head of the subpath in each iteration. This transition becomes the new *head transition* of the subpath. The transition subpath grows until the sum of the mean transition times of the transitions along the subpath is larger than the given mean queue wait time,

Procedure 1 : *compute weights given mean queue wait time*
Inputs : $\overline{W}_{ij}$ – the mean queue wait time of class ij incoming messages;
Outputs : the weights of all the transitions in the PEFSM, w_{uv} $(u, v \in Q)$;
Steps :
1. initialize the weights of all the transitions to zero, i.e., $w_{ij} = 0$ for $i, j \in Q$;
2. call *Procedure 2* with arguments (1, ij, $\overline{W}_{ij}$).

Procedure 2 : *recursively backtrack to add the subpath probabilities to the transitions which are the heads of the subpaths*
Inputs : 1) the current subpath probability p;
2) the reference of the current transition uv;
3) the remaining waiting time R_w;
Outputs : weights of the transitions;
Steps :
1. if $R_w \leq 0$, return;
2. for (each transition which is immediately before the current transition uv in the FSM of the PEFSM, say transition ku) do :
   1) let $p = p * p'_{ku}$ where p'_{ku} is the steady-state transition probability of transition ku;
   2) let $w_{ku} = w_{ku} + p$;
   3) call *Procedure 2* with arguments (p, ku, $(R_w - {}^{\psi}\bar{\tau}_{uv})$) ;
   endfor.

Figure 3: Algorithm for weight computation.

$\overline{W}_{ij}$. At each step, the current *subpath probability* is added to the weight of the head transition.

When Procedure 2 terminates, the weights of all the transitions in the given PEFSM with respect to the mean queue wait time, $\overline{W}_{ij}$, are computed. These weights reflect the relative frequency of the transitions seen by class ij incoming messages. If the transition time of each transition is reduced by the same amount one at a time, the one which has the largest weight will cause the largest improvement in the mean queue wait time of class ij messages. Therefore, the transition with the largest weight is the *bottleneck transition* with respect to the mean queue wait time.

4.3 Simulation results

Simulations have been conducted to validate the accuracy of the bottleneck identification method. The architecture of the simulation experiment is shown

in Figure 4.

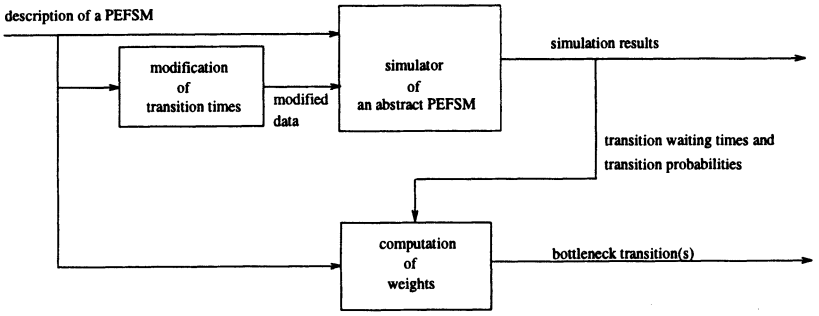


Figure 4: A simulation architecture.

The *simulator* module accepts a model description of the PEFSM and simulates the execution of transitions in the FSM of the PEFSM. The module contains an *incoming message generator* which generates the incoming messages to the PEFSM based on the given arrival model of the PEFSM.

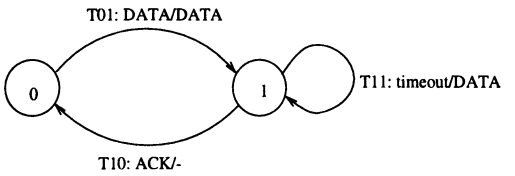
The simulation results are fed to the *weight computations* module. This computation module also stores the description data of the PEFSM. The algorithms in Figure 3 are used to compute the weights of all the transitions with respect to the mean queue wait time of a specific transition. The transition with the largest weight is the *bottleneck transition.*

The *modification* module reduces the service time of each transition of the PEFSM by the same small amount one at a time. This module resends the modified data of the PEFSM to the simulation module and the simulation is rerun. All the mean queue wait times of class ij incoming messages in each run are recorded so as to verify if the bottleneck transition in fact causes the largest reduction in the mean queue wait time.

Several protocols were used in our experiments which showed that the proposed technique for bottleneck transition identification works in practice. We report the result of the *alternating bit protocol* in the following.

The FSM of the alternating bit protocol is given in Figure 5. The input data of the PEFSM are given in Columns 2, 3 and 4 of Table 1. The incoming data packets to be transmitted arrives in a Poisson pattern with a mean rate of 200.0 packets/second.

Columns 4, 5 and 6 are the simulation results. The steady-state transition probabilities were recorded in Column 4. These results agree with the results from computation of $p'_{ij} = \pi_i p_{ij}$ where π_i is the steady-state probability of state i, and p_{ij} is the single-step probability of transition ¡i,j¿. The weights were computed with respect to the mean queue wait time of a class of incoming messages and recorded in Column 5. Then, in each of the subsequent runs, we selected one of the transitions and reduced its service time by a small amount (0.002 second). The simulation was re-run with the modified data.



STATES :
state 0 : idle
state 1 : waiting for ACK (acknowledgement)

TRANSITIONS :
T01 : sending a DATA packet
T10 : receiving an ACK
T11 : timeout retransmission

Figure 5: An FSM of the alternating bit protocol.

Table 1: A simulation result of bottleneck identification.

transition identifier	single-step tran. prob.	mean transition time[a]	transition probability[b]	transition weight[c]	queue wait time reduction[d]
T01	1.0	0.001	0.4738	0.3102	0.0069
T10	0.9	0.002	0.4738	**0.6207**	**0.0090**
T11	0.1	0.002	0.0525	0.0343	0.0010

[a] All times are in seconds.

[b] Steady-state transition probability.

[c] The weight is computed with respect to the mean queue wait time of the data packets.

[d] The new average queue wait time is measured by decreasing the mean service time of the corresponding transition by 0.002 second. The reduction of the queue wait time is equal to the original value minus the new value.

The reduction in mean queue wait time was recorded in Column 6. This procedure is repeated for all the transitions. From the table, we can see that reducing the service time of the transition with the largest weight causes the largest reduction in the mean queue wait time of that class of incoming messages. This result confirms the analysis given in this section.

More than 20 experiments with different work load parameters had been performed for several protocols. In most cases, the results from simulation agreed with the analytic results. Only 3 exceptions were found. However, even then, the reduction of the mean queue wait time by reducing the service time of the bottleneck transition was very close (within 15%) to the largest reduction. The reason why the proposed procedure occasionally does not correctly identify the bottleneck transition is that both the queue wait times and the transition times have variance and we use only the mean value to compute the weights for simpilicity.

5 RELATED WORK

Performance bottleneck detection and removal have received less attention than performance prediction. This is not only because the problem itself is hard but also because there is a lack of adequate formal definitions and effective analysis methods. Only a few methods have been proposed to locate the performance bottleneck of a software system.

The concept of *critical path* was first introduced in the context of *project management* and used to manage the progress of projects [Lock84]. It was later adopted for parallel processing systems [Yang89] where there are both parallel events and synchronization points. The critical path of a parallel program is the path in the program activity graph[4] which determines the program performance (e.g. shortest execution time). The possible domain of the potential bottleneck is that of the critical path. Other techniques are used for locating the bottleneck within the critical path.

Lockyer's *critical path analysis* [Lock84] is often used to identify bottlenecks in parallel or distributed systems which are modeled as acyclic directed graphs [Yang89, Wagn93]. However, only one transition in a PEFSM can be executed at a time. The execution of the transitions in a PEFSM are sequential and follows a certain order. There is no synchronization with other transitions in a single PEFSM. Therefore, the method of critical path analysis can not be directly applied to PEFSMs in identifying bottlenecks.

Although intuitively we all know what a bottleneck is, historically, the term *bottleneck* has had various definitions. They can be classified into the following two categories according to usage :

- analytical definitions
- measurement based definitions

Using derivatives is a common analytical approach to identifying a performance bottleneck. For example, in [Ferr78], the *derivatives* of the mean throughput rates with respect to the service rates of the constituent servers of the system are used to define performance bottlenecks analytically. If

$$\frac{\partial \overline{T}}{\partial \mu_i} > \frac{\partial \overline{T}}{\partial \mu_j} \; (j = 1, 2, ..., s; j \neq i)$$

then server Σ_i is the performance bottleneck of a system with s servers, where $\overline{T}$ is the mean throughput rate of the object system; μ_k is the service rate of server Σ_i (k=1,2,...,s).

However, this definition can not be used if $\overline{T}$ is not differentiable.

---

[4] A program activity graph is a directed graph which depicts the synchronization points of the whole system.

Utilization based techniques constitute another analytical way for determining *performance bottlenecks* [Leung88, Allen90]. Among the servers in a queuing network model, the one with the highest utilization or the one which first achieves 100% utilization with increasing workload on the system is considered to be the bottleneck of the system. However, this approach is not appropriate for a PEFSM because it is assumed to have only one service center.

Generally, the analytical definition is applied to a model of the system. When an implementation of the system already exists, *analyses* of data from *measurement* can be used to identify the bottleneck. In [ZiEt92], the bottleneck is defined as the performance parameter which is most sensitive to performance. The sensitivity of a parameter is defined as

$$sensitivity \stackrel{\text{def}}{=} \frac{\%ChangeInPerformance}{\%ChangeInParameter}$$

Intuitively, the sensitivity is similar to the *weight* to a certain extent. Both can be used in analytical approaches and measurement approaches.

6 CONCLUSION

We have proposed a methodology to identify performance bottlenecks based on a performance extended FSM model PEFSM. *Weights* are used to measure the impact of the reduction of each transition time on the improvement of a specific performance metric. The bottleneck with respect to a performance metric is defined to be the transition in the PEFSM with the maximum *weight*.

The methods to compute the *weights* of the transitions in a PEFSM with respect to two performance metrics are presented. The first method makes use of the closed-form expression of a performance metric such as *throughput*. This depends on the existence of both the closed-form expression of a performance metric and the partial derivatives of the performance metric with respect to each transition time. The second method uses an approximate recursive algorithm to compute the weights with respect to a performance metric. This method is used when no closed-form expression of the performance metric or derivatives exists.

The second method was used to identify the bottleneck transition with respect to the mean queue wait time of a specific class of incoming messages. It is more general than the method of derivatives. This second method can be applied to the PEFSM in which the arrivals of asynchronous messages are not Poisson, and the mean queue wait time may be obtained either by measurement or computation.

The mean transition time of the bottleneck transition can be reduced in two ways: reducing the *mean transition wait time* or the *mean transition service time*. To reduce the mean transition wait time, one may increase the arrival

rate of the incoming messages associated with that transition. For example, we can increase the throughput rate of messages or decrease the queue wait time of messages in a specific workstation by shortening the *token* turnaround time for this workstation in a token ring network. To reduce the transition service time, one may try to improve the software implementation of that transition or use faster hardware to process the transitions.

7 REFERENCES

[Allen90] Arnold O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications.* Academic Press, Inc., second edition edition, 1990.

[Ferr78] Domenicao Ferrari. *Computer Systems Performance Evaluation.* Prentice-Hall, INC., Englewood Cliffs, New Jersey, 1978.

[Howa71] Ronald A. Howard. *Dynamic Probabilistic Systems*, volume Volume 1: Markov Models; Volume 2: SemiMarkov and Decision Processes. John Wiley & Sons, Inc., 1971.

[ISO2576] ISO TC97/SC16 N2576. *Formal specification of Transport protocol in Estelle.* ISO, 1986.

[ISO7776] ISO/TC 97, Information processing systems. *Information processing systems - Data communication - High-level data link control procedures - Discription of the of the X.25 LAPB-compatible DTE data link procedures.* ISO 7776 - 1986. ISO, first edition, 12 1986.

[ISO8807] ISO TC 97/SC 21. *Information processing systems - Open systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour.* ISO 8807. Interational Organization for Standardization, 1989.

[Leung88] Clement H.C. Leung. *Quantitative analysis of computer systems.* Chichester; New York : Wiley, 1988.

[Lock84] K.G. Lockyer. *Critical path analysis and other project network techniques.* Pitman Publishing, 1984.

[SDL] CCITT. *Specification and Description Language - Recommendation Z.100.* Geneva, Switzerland: CCITT press, 1986.

[Tane88] Andrew S. Tanenbaum. *Computer Networks.* Pretice-Hall, Inc., 1988.

[Wagn93] A. Wagner, J. Jiang, and S. Chanson. TMON - A Transputer Performance Monitor. *Concurrency: Practice and Experience*, 5(6):511–526, 1993.

[Yang89] Cui-Qing Yang and Barton P. Miller. Performance Measurement for Parallel and Distributed Programs: A Structured and Automatic Approach. *IEEE Transactions on Software Engineering*, 15(12), 12 1989.

[Zhang95] Sijian Zhang. *Performance Modelling and Evaluation of Protocols based on Formal Specifications.* PhD thesis, The Univ. of British Columbia, 1995.

[ZiEt92] John A. Zinky and Joshua Etkin. Troubleshooting throughput bottlenecks using executable models. *Computer Networks and ISDN Systems*, 24(1), 3 1992.

8 BIOGRAPHY

Dr. Sijian Zhang

Dr. Sijian Zhang graduated from Beijing Univ. with B.Sc and M.Sc. He received his Ph.D. degree from Univ. of British Columbia, Canada in 1995. He is currently working for Hughes Aircraft of Canada. Dr. Zhang's research interests include protocols, conformance and performance testing, formal specifications and distributed computing.

Dr. Samuel Chanson

Dr. Samuel Chanson received his Ph.D. degree from the University of California, Berkeley in 1975. He was a faculty member at Purdue University for two years before joining the University of British Columbia where he became a full professor and director of its Distributed Systems Research Group. In 1993 Professor Chanson joined the Hong Kong University of Science & Technology as Professor and Associate Head of the Computer Science Department. He has chaired and served on the program committees of many international conferences on distributed systems and computer communications, including IEEE ICDCS, IFIP PSTV and IWTCS.

Dr. Chanson's research interests include protocols, high speed networks, multimedia communication and load balancing in workstation clusters. He has published more than 100 technical papers in the above areas.