DISTRIBUTED PROCESSING SYSTEMS
ANALYSIS OF RESPONSE TIME

Nick Redding
Bell-Northern Research
Ottawa, Ontario
Canada.

1 Introduction

This paper addresses response time analysis of distributed communications processing systems. Such systems may be modelled directly as queueing networks, but the analysis of these networks is often computationally difficult or impossible. The method of analysis described here requires that some simplifying assumptions be made to obtain a queueing network model that can be solved by decomposition; each subnetwork in the decomposition is characterized by a simple response time approximation which is generally quite accurate. The subnetwork response time approximations are either formulas that can be evaluated in constant time, or the roots of equations involving these types of formulas, and are therefore easy to evaluate.

Queueing network analysis by decomposition has been studied by several authors. The paper by Kuehn [1] describes an approach that is similar in flavor to that proposed here. The primary distinction is that while Kuehn's method is more general in many respects, it is also considerably more complex and does not account for structures arising from round-robin task allocation to multiple nodes, synchronous interactions between nodes, and finite population sources.

Figure 1 illustrates the architecture of the types of distributed processing systems we wish to consider.

Figure 1: Distributed processing system architecture.
Work enters and exits the system via communications processors, and moves between nodes via the local area network.
Multiple processors (of different types) are connected by a local area network, and incoming work may require service from several processors before departing the system. The arrival process for incoming work may be infinite source (Poisson) or finite population (for example, online terminals).

The distributed processing systems that we are concerned with impose largely deterministic service time requirements on incoming work. Furthermore, since capacity is increased by adding processors, for moderately large systems it is reasonable to assume that individual processors perform single types of work (ie. they are segregated according to the type of work performed). These two simplifications lead to a queueing network model in which each node is a processor and jobs arriving at each node have a constant service time requirement.

Analysis of systems that do not exhibit segregation of work is a second-order modelling problem, since such systems are obtained from segregated systems by merging sets of nodes into one. It is conjectured that an approach to dealing with segregated systems will yield a basis for developing a second-order modelling approach for non-segregated systems, and this will be a topic for future research.

A final simplification is to remove feedback paths from the queueing network. Since the systems we are concerned with do not, in general, exhibit feedback loops that may be traversed a variable (or random) number of times, we can eliminate such loops by ensuring that a job receives all of the service it would eventually require during the first visit to each node in its path. Any edges that feed back into the network of queueing systems then become exit points from the network.

The result of the above simplifications is a queueing network model with the property that it is open with no feedback paths, and all service times are constant. As a result, the arrival processes to internal nodes in the network are quite regular (especially under conditions of moderate to high processor utilizations).

2 Queueing Network Structure

The queueing networks that we consider may be represented by graphs without circuits. Edges that connect nodes may be asynchronous or synchronous. In the former case, a job completing service at the source node frees that node to serve another job immediately. In the latter case, the source node is not freed until the destination node is freed by the completing job. Clearly, the relationship of nodes being connected by synchronous edges may be extended transitively. It is commonly used to prevent multi-part transactions becoming interleaved.

As a job progresses through the network, the path followed is determined by the allocation policy at each node encountered. The set of all paths that a job could take is called a routing tree, and the set of successors to a node in a routing tree is called a routing set.

Nodes in a routing set receive jobs in a round-robin fashion. In order that this allocation policy may be used to quantify
minimum interarrival times from one specific node to another, it is necessary to restrict routing sets at the same depth in a routing tree to be identical or disjoint.

A second restriction on the routing trees enables the performance of the network to be modelled by studying a single path through any of the routing trees. Define two routing trees or subtrees to be isomorphic if their nodes and edges are in one-one correspondence with identical parameters (arrival rates, service times, asynchronous versus synchronous relationships). Furthermore, define a routing tree to be symmetric if for each node in the tree, all interactions with its successor nodes are either all synchronous or all asynchronous, and all of its subtrees are mutually isomorphic. We assume that all routing trees are symmetric and mutually isomorphic.

3 Queueing Network Parameters

We consider queueing networks which have a finite number of arrival sources, and we assume that all of the arrival sources are of the same type (finite population or infinite population) and have the same job arrival rate, denoted by \( \lambda \). An arrival source will be represented by a special type of node in the queueing network; these source nodes have the property that they have exactly one edge leading away from them and no edges leading into them.

A finite population source will be represented as a collection of unit population sources, corresponding to populations of one. Since a unit population source is inactive while it has a job in the network, its arrival rate is the rate at which a new job is generated after the previous job has completed service. It follows that if the mean response time of a job from a unit population source is \( r \) then the mean interarrival time from that source to the network is \( r+1/\lambda \), while for infinite population sources it is \( 1/\lambda \) regardless of the response time. In the case of unit population sources, we associate a minimum interarrival ('think') time \( D \) with the edge leading away from each source node.

4 Network Decomposition

The performance analysis of a network has reduced to that of a path \( N_0 \rightarrow N_1 \rightarrow \ldots \rightarrow N_k \) where \( N_0 \) is an infinite or unit population source. We are interested in evaluating the response time through the path (both in terms of mean and percentiles), and the approach is to deal with each node in the chain independently. Thus, assuming that jobs arriving at node \( N_i \) will have response statistic \( T_i \), the response statistic for the path will be

\[ r = T_1 + \ldots + T_k \]

(Note that for the mean response time, this decomposition into a sum is exact, but for percentile response times it is an approximation.)

If \( N_0 \) is an finite population source, the statistic \( T_i \) is a
function of \( r \), and the individual node statistics may be evaluated after solving the following equation for \( T \):

\[
r = T_1(r) + \ldots + T_k(r)
\]

In this case, the path is called a *finite population chain*.

For both types of source node, the analysis of the path requires that closed-form expressions be developed for the individual node statistics.

If we consider the decomposition of the path \( N_0 \rightarrow \cdots \rightarrow N_k \) we see that each service node falls into one of several categories:

1) It is the first node in a path with an infinite population source, and has (a) an asynchronous or (b) synchronous interaction with next node in the path.

2) It is a service node (at least the second such node in a path with an infinite population source) with several arrival sources, a non-zero minimum interarrival time per source, and (a) an asynchronous or (b) synchronous interaction with the next node in the path.

In the first case, the node statistics may be evaluated using standard M/G/1 queueing theory results for the mean and general distribution of response time. In case (1a), the M/D/1 formulas may be used. In case (1b), the effective service time of a job at the server will include the response time at the subsequent node. In order to apply the result for mean waiting time in a M/G/1 queue, the mean and variance of the service time are required. The mean is clearly \( d_1 + r \) where \( d_1 \) is the service time at \( N_1 \) and \( r \) is the approximation of the mean response time at the subsequent node. The variance may be approximated using the following result:

**Proposition V:** (Estimated Variance) Let \( R \) be a non-negative random variable with mean \( r \). Assume that \( p \) is a percentile response time statistic, i.e. \( \Pr[R \leq p] = \alpha \) where \( p \) is large relative to \( r \) (i.e. \( p \leq 2r \)). Then \( \sigma^2_R \geq (1-\alpha) (p-r)^2 \).

The variance of the random variable \( d_1 + R \), where \( d_1 \) is constant, is simply the variance of \( R \). Thus, Proposition V enables us to estimate the variance of the service time at the first node (by adopting the lower bound) using the mean and percentile response time approximations of the subsequent node. The percentile response times in case (1b) may be approximated using the M/D/1 approximation with the constant service time set to \( d_1 + r \).

In cases (2a) and (2b), the node statistics may be approximated using the following results:

**Proposition A:** (Asynchronous Response) Consider a single-server queueing system with \( k \) independent arrival streams and constant service time \( y \). Assume that the arrival processes for each stream are arbitrary, except that each stream has arrival rate \( \xi \), and the minimum interarrival time in any stream is at least \( ky \). The mean response time \( T \) of a job from any stream may be approximated by

\[
4.18.4.4
\]
The response time percentiles may be evaluated using the same method as in Proposition A.

In case (2b), it is necessary to consider a maximal subpath of nodes with consecutive synchronous interactions, say

N_a \rightarrow_{\text{sync}} N_{a+1} \rightarrow_{\text{sync}} \ldots \rightarrow_{\text{sync}} N_B

and evaluate the response time statistics starting at the end of the subpath. This analysis yields a response statistic for the entire subpath, and therefore only one summand would contribute to the expression for network response time.

5 Constraints on Node Utilizations

For a large subclass of networks satisfying the required assumptions, equivalent conditions in terms of node utilizations exist. If there is only one successor routing set for every internal node in the network, then we refer to the network as converging.

In the case of a converging network (note that this includes all networks with only one job class), consider a representative path N_0 \rightarrow \ldots \rightarrow N_k. Let P_i denote the server utilization of service node N_i. If N_i has an asynchronous interaction with its successor, then define t_i=r_i. Otherwise, N_i is a member of a maximal subpath N_a \rightarrow_{\text{sync}} \ldots \rightarrow_{\text{sync}} N_B of nodes with synchronous interactions. In this case, we define
where $m_i$ is the fan-in of $N_i$.

Then the following equivalent conditions exist for the premises of propositions A and S to be satisfied:

1) **Infinite population sources**: The premises will be satisfied if and only if:

$$\pi_i \geq \tau_{i+1} \text{ for } 0 < i < k$$

2) **Unit population sources**: The premises will be satisfied if and only if in addition to condition (1) above, the nodes in the path satisfy

$$m_1 d_1 \leq D + d_1 + \ldots + d_k$$

(recall that $D$ is the minimum 'think' time of a unit population source).

Note that if all interactions are asynchronous, then the conditions are simply $\rho_i \geq \rho_{i+1}$. If there are synchronous interactions, but only one customer class, then

$$\pi_i = \rho_i + \ldots + \rho_b$$

If a network is not converging, then the premises of Propositions A and S become much stronger restrictions.

**6 Elimination of Nonconforming Nodes**

A simplification may be made to deal with a service node $N_i$ where $\rho_i \ll \rho_{i+1}$. In this case, the premises required by Propositions A and S will not be satisfied by $N_{i+1}$ and we may consider $N_{i+1}$ to be a node with arrival streams directly from the nodes represented by $N_{i-1}$. This strategy of bypassing $N_i$ is based on the fact that if $\rho_i \ll \rho_{i+1}$ then $N_i$ does not introduce any significant regularity into the arrival stream to $N_{i+1}$, and therefore discounting it from the calculation of $T_{i+1}$ will not introduce significant error. Now if $\rho_{i-1} \geq \rho_{i+1}$ then the premises are satisfied; otherwise, $N_{i-1}$ may be bypassed, and this process may be continued until either the source node is encountered or the premises of the propositions are satisfied.

**7 Treatment of Bottleneck Nodes**

Although bottleneck nodes (defined as a node through which all customers from a job class must pass) should be avoided in the design of distributed systems, they may arise in the
consideration of specific system configurations. Although Propositions A and S remain valid in the context of bottleneck nodes, the premises can sometimes be relaxed by eliminating or applying special considerations to degenerate bottleneck configurations.

Assume that node $N_i$ is a bottleneck node. Several cases may be considered:

$N_i \rightarrow_{\text{async}} N_{i+1}$

In this case, only one of the successors of $N_i$ can be processing a job at any point in time. Therefore, the successors to $N_i$ may be removed if the service time at $N_i$ is changed to $d_i + d_{i+1}$.

$N_i \rightarrow_{\text{async}} N_{i+1} \rightarrow_{\text{async}} N_{i+2}$

In this case the subnetwork containing $N_i$ and its immediate successors can be rearranged according to a result from Tembe and Wolff [2] which states that it is equivalent to the network obtained by inverting the order of $N_i$ and its immediate successors.

$N_i \rightarrow_{\text{async}} N_{i+1} \rightarrow_{\text{sync}} N_{i+2}$

In this case, if $d_i > r_{i+1}$ then no queue will ever form past $N_i$ and we can set $T_j$ accordingly for $j > i$.

It is possible to design a network with more servers at a given stage than could be occupied at any time. The limitation on the number of servers that may be occupied is determined by the number of unit population sources present, and/or the number of synchronous nodes of any degree. In the latter case, synchronous nodes impose a limit on the number of jobs that may be present in the synchronous subpaths of which they are members. In the former case, unit population sources place an overall limit on the number of jobs that may be present in the network. If the fan-in $m_i$ at a node $N_i$ exceeds the limit imposed by synchronous nodes or unit population sources, then it may be replaced by that limit in the application of Propositions A and S.

8 Application of the Method

The methods described here have been successfully applied to the analysis of a distributed processing system for the handling of Signalling System #7 queries (telephony and advanced services processing). Validation of these and more general applications by simulation has demonstrated that the approximations are accurate within 10 per cent for most load levels of practical interest.
REFERENCES
