1 Introduction and motivation

Component-based software development consists of developing a software system by assembling it from pre-existing or reusable components rather than building it from scratch. It offers several advantages over the traditional life-cycle software development, the two most notable ones being a reduction in the application development cost and an improvement in the application quality. Cost reduction is enabled by reusing components in multiple applications, whereas quality improvement is achieved by testing components more thoroughly in different setups and environments.

Many component-based software applications are used in life and mission-critical distributed real-time (DRE) systems. These systems have to meet stringent performance (response time) and availability requirements. Replication of the application components may be an effective way to achieve the availability objectives of the application. Accordingly, the Corba Component Model (CCM) has defined two replication models, namely active and passive replication models for this purpose. The advantages and disadvantages of these replication models have been discussed extensively. These replication models have a distinct influence on the performance of the application. Although DRE systems impose strict performance requirements,
an extensive quantitative analysis of these replication models on the application performance has not been conducted.

In this paper we describe a methodology to analyze the application performance employing active and passive replication models. The application performance metric we consider is the end-to-end response time. The methodology can be used to assess the influence of different configuration parameters of each replication model on the response time. It can also be used to compare and contrast the replication models for a given application. We illustrate the value of the performance analysis methodology with examples.

The layout of the paper is as follows: Section 2 describes the characteristics of the system under consideration. Section 3 presents the two alternative architectures of the system, one for each replication model. Section 4 discusses the performance analysis of the two architectures presented in Section 3.

2 Application description

We consider an application with two types of components, namely component A and component B. These components are organized in a tier or a cascade structure with components of type A residing in tier A and components of type B residing in tier B as shown in Figure 1. In general, to provide fault-tolerance and for load balancing, multiple instances of each type of component may be present in each tier.
A client request is processed sequentially by this cascade of components in the following steps:

1. The client request is received by component of type A and it processes the request. During the processing determines that it needs a service from component of type B. Component type A then sends a request to component type B.

2. Component type B processes the request from component type A and sends a response back to component type A.

3. Component type A completes processing and sends a response back to the client.

The processing steps depicted in the form of a sequence diagram are shown in Figure 2.

3 Alternative architectures

In this section we describe the alternative architectures and configurations along with the replication models used to enhance the application availability. In this paper, we consider failures of the components and these configurations are employed to guard the application against such failures. The two architectures described
in the next two sections can withstand exactly one failure of each one of the component types A and B while serving a single client request.

3.1 Passive replication

In the passive replication scheme, an active instance of component A resides on host #1 and a passive instance resides on host #2. Similarly, an active instance of component B resides on host #2 and a passive instance resides on host #1. The passive instance of each type of component periodically synchronizes its state with the active instance of that type of component on the other host and this consumes a fraction of the processing capacity of each host. When the active instances of the components do not fail during the processing of a client request, the processing steps are the same as the ones reported in Section 2. However, if an active instance of a component fails during the processing of a client request, then a failover to the passive instance is necessary and this requires the passive instance to bring its state up to date from the last synchronization point. A pictorial depiction of the passive replication strategy is shown in Figure 3. In the figure, instances $A_1$ and $B_2$ are active instances and instances $A_2$ and $B_1$ are passive replicas of components $A$ and $B$ respectively.
3.2 Active replication

In the active replication scheme, an active instance of each component resides on each one of the hosts. A processing request for a given component type is sent to all the instances of the component. However, duplicate responses are suppressed and exactly one response is forwarded further for subsequent processing or back to the client. A management software residing on host #3 is responsible for processing the requests and sending to all the active component instances and suppressing duplicate responses. A certain portion of the processing capacity of the hosts is consumed in sending and receiving messages from the manager. A pictorial depiction of the active replication strategy is shown in Figure 4.

A client request is processed by the architecture employing the active replication strategy in the following steps:

M1. The client request is processed by the manager and sent to all the replicas of component A.

1. All the replicas of component A process the client request.

M2. The request from all the replicas of component A are processed by the manager to suppress duplicates
and exactly one request from component A is sent to all the replicas of component B.

2. All the replicas of component B process the response from component A.

M3. The responses from all the replicas of component B are processed by the manager to suppress duplicates; a single response from component B is sent back to each replica of component A.

3. All the replicas of component A complete processing.

M4. The responses from all the replicas of component A are processed by the manager to suppress duplicates; a single response from component A is sent back to the client.

Steps M1, M2, M3 and M4 represent processing by the management software.

4 Performance analysis

In this section we discuss the performance analysis of the active and passive replication architectures presented in Section 3. The performance metric of interest is the end-to-end response time of a client request as it is processed by the cascade of components. In order to compute the end-to-end response time of a client request we follow the trajectory of the request through the processing steps outlined for each replication model. Since we are interested in comparing the impact of replication strategies on the end-to-end response time of a single request, we do not consider simultaneous processing of multiple requests.

The random variables denoting the processing times of the components which are common to both the replication models are as follows:

- $T_{A,1}$ - Execution time of component type A, up to the point where it determines that it needs a service provided by component type B (Step #1).

- $T_{A,2}$ - Execution time of component type A after it receives response from component type B (Step #2).
• $T_B$ - Execution time of component type B (Step #3).

4.1 Passive replication

In the passive replication strategy, a fraction of the processing capacity of each host is expended in the synchronization of the passive replica with the state of the active replica. This slows down the actual execution of the active instance on the host. Let $\alpha$ denote the fraction of the processing capacity of each host that is consumed in the synchronization process. Then, the execution of the active instance residing on the host will be slowed down by a factor of $(1 - \alpha)$. Further, when an active instance of a component fails, failover to the passive replica entails a recovery phase which consists of synchronization of the passive replica with the current state of the active replica. The exact time spent by the application in the recovery phase will depend on the time elapsed from the last synchronization operation. We let the random variable $T_m$ denote the time spent by the application in recovery and failover.

As described in Section 3, since the architecture can withstand one failure of each component type while serving a client request, the following four possibilities arise: (i) Components of type A and B complete processing without failure, (ii) Component of type A fails, component of type B completes processing without failure, (iii) Component of type A completes processing without failure, component of type B fails, and (iv) Both type A and B components fail during processing. The end-to-end response time of a client request in each one of the four cases can be obtained as follows:

**Case I: Component types A and B do not fail:**

In this case, the expected response time denoted $E[R_1]$ is the sum of the processing times of steps summarized in Section 2 and is given by Equation (1).

$$E[R_1] = E\left[\frac{T_{A,1}}{1 - \alpha}\right] + E\left[\frac{T_{A,2}}{1 - \alpha}\right] + E\left[\frac{T_B}{1 - \alpha}\right]$$

**Case II: Component type A fails, component type B does not fail:**
In this case, the expected response time denoted \( E[R_2] \) is given by the sum of the processing times of the steps summarized in Section 2 and the time taken to recover from the failure of component type A. \( E[R_2] \) is given by:

\[
E[R_2] = E\left[\frac{T_{A,1}}{1-\alpha}\right] + E\left[\frac{T_{A,2}}{1-\alpha}\right] + E\left[\frac{T_B}{1-\alpha}\right] + E[T_{rec}]
\]  

(2)

**Case III: Component type A does not fail, component type B fails:**

In this case, the expected response time denoted \( E[R_3] \) is the same as the response time in Case II.

**Case IV: Component types A and B fail:**

In this case, the application recovers from two failures and hence the response time denoted \( E[R_4] \) is given by:

\[
E[R_4] = E\left[\frac{T_{A,1}}{1-\alpha}\right] + E\left[\frac{T_{A,2}}{1-\alpha}\right] + E\left[\frac{T_B}{1-\alpha}\right] + 2E[T_{rec}]
\]  

(3)

If \( R_A \) and \( R_B \) denote the reliabilities of component types A and B respectively, then the average response time can be computed as the weighted sum of the expected response times in the above four cases, with the weights given by the probability of occurrence of each one of the four cases. The expected response time \( E[R] \) is thus given by:

\[
E[R] = R_A R_B \left( E\left[\frac{T_{A,1}}{1-\alpha}\right] + E\left[\frac{T_{A,2}}{1-\alpha}\right] + E\left[\frac{T_B}{1-\alpha}\right]\right)
\]

\[+(R_A(1 - R_B) + R_B(1 - R_A))(E\left[\frac{T_{A,1}}{1-\alpha}\right] + E\left[\frac{T_{A,2}}{1-\alpha}\right] + E\left[\frac{T_B}{1-\alpha}\right] + E[T_{rec}]\)
\]

\[+(1 - R_A)(1 - R_B)(E\left[\frac{T_{A,1}}{1-\alpha}\right] + E\left[\frac{T_{A,2}}{1-\alpha}\right] + E\left[\frac{T_B}{1-\alpha}\right] + 2E[T_{rec}]\)
\]  

(4)

### 4.2 Active replication

In this case, the time incurred by the management software in processing the requests, sending the requests to all the active instances and suppressing the duplicates adds to the response time. We let the random
variable $T_m$ the processing time of the manager for each one of the processing steps M1, M2, M3 and M4 in Section 3.2. Further, since a fraction of the processing capacity of the hosts is spent in interacting with the management software to send and receive messages, the actual execution of the component instances is slowed down. We let $\beta$ denote the fractional processing capacity of the hosts consumed in the sending and receiving messages from the management software. The execution time of the active component instances residing on each host would be slowed down by a factor of $(1 - \beta)$. As mentioned earlier, since we do not consider simultaneous processing of multiple requests, on each host only one active instance is actually processing the request. The expected end-to-end response time is obtained as the sum of the processing steps outlined in Section 3.2 and is given by Equation (5).

$$E[R] = 4E[T_m] + E\left[\frac{T_{A,1}}{1 - \beta}\right] + E\left[\frac{T_{A,2}}{1 - \beta}\right] + E\left[\frac{T_B}{1 - \beta}\right]$$  \hspace{1cm} (5)

This is the response time of the architecture with active replication model in each one of the four cases discussed in Section 4.1. Hence, this is the expected end-to-end response time of the architecture.