

Error Detection in Distributed Systems

Naghmeh Ghafari, Alexander Lau, Barry Pekilis, Rudolph Seviora

Bell Canada Software Reliability Laboratory, University of Waterloo

Waterloo, Ontario, Canada, November 2002

{nghafari, alexlau, bpekilis, seviora}@swen.uwaterloo.ca

I. INTRODUCTION

Experience shows that external failures of software systems are often preceded by deterioration in their internal state. From an operational perspective, the capability to provide an external indication of a growing corruption of the internal state of key program entities (the presence of error) would be very valuable [1]. A major class of errors (corruption) in the internal state is indicated by state inconsistencies. The state of two entities is consistent with each other if their relationship is legal under the design specification. The state consistency relationships are defined as boolean predicates on the states of individual processes. Inconsistencies in the program's state may lead the program on an erroneous execution path and result in an external failure. To detect state inconsistencies, the state information must be collected and the relationship predicates must be checked.

A distributed system consists of spatially separated processes and communication channels. The local state of a process is characterized by the state of its local memory. The local state of a channel is characterized by the messages in transit on that channel. The global state of a distributed system is the collection of the local states of its processes and communication channels.

The state information of a process is very large. To reduce the computational cost of dealing with this amount of information, the process state information can be abstracted. The techniques for determining the granularity of abstraction are not addressed in this paper.

This paper considers two approaches for checking the state consistency relationships between the abstracted states of a pair of processes in a distributed system. The first approach is based on sampling the state of the distributed processes. In this approach, a perspective of the partial system state is captured by ignoring the states of the communication channels. Consistency relationships can be evaluated by using a *state relationship distribution matrix*. The evaluation is statistical, not deterministic. The second approach captures a precise (undistorted) partial system state through the use of marker messages and creation of *simultaneous regions* [2]. In both approaches the communication channels are assumed to have infinite capacity, to be error-free, and to deliver messages in the order sent.

II. STATISTICAL SAMPLING

Collecting the states of processes in a distributed system without considering the states of the communication chan-

nels may result in a distorted view of the partial system state. However, in many circumstances, the distortions can be filtered out by applying statistical techniques.

In the statistical sampling approach, a local state monitor is associated with each process in the distributed system whose state appears in the consistency predicates being checked. The local state monitor contains the procedure **sample_state**, shown in Fig. 1.

```

procedure sample_state ();
{
    state := associated_process.get_state ();
    send_parent_state (state);
}
    
```

Fig. 1, **sample_state** procedure.

A parent monitor is responsible for collecting process states from the local monitors and checking their state consistency relationships. It periodically sends a request message to the local monitors. When a local monitor receives the request message, it invokes its **sample_state** procedure to obtain the state of the associated process. The sampled state is sent back to the parent monitor. The parent monitor builds a state relationship distribution matrix, which contains the frequency of observing different state relationships.

	P ₁	...	P _{i-1}	P _i	P _{i+1}	...	P _n
Q ₁				⋮			
⋮				0.02			
Q _i		0.05	0.82	0.07	
⋮				0.04			
Q _m				⋮			

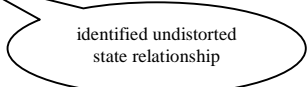


Fig. 2, State relationship distribution matrix.

Fig. 2 illustrates a state relationship distribution matrix. The $n \times m$ matrix corresponds to n and m states of processes p and q respectively. Through comparing with a threshold or using other statistical techniques, the distortions in the state relationships due to messages in transit can be filtered out. For example, if the maximum communication channel delay is D_{max} , the maximum number of possible distortions of a specific state relationship during the sampling time is $D_{max} \cdot f$, where f is the state sampling frequency. The undistorted state relationships are those observed with frequencies higher than the threshold $(D_{max} \cdot f) / T_{obs}$, where T_{obs} is the observation time. After filtering out the distortions in the state relationships, the state consistency relationships can be evaluated on the resulting state relationship distribution matrices.

III. SIMULTANEOUS REGIONS

In order to capture the exact relationship between the states of processes in a distributed system, a temporal relationship must be established between the corresponding states by some artificial means (due to the lack of an accurate global clock). Spezialetti et al. proposes in [2] a technique to establish a temporal relationship between the distributed processes for distributed event recognition. The same approach can be used to establish the notion of simultaneity between the corresponding states of distributed processes. The approach considered is based on a dynamic division of execution of processes into a sequence of numbered regions. The term *simultaneous regions* refer to regions with the same region number, which means that the activities within those regions can be considered to be concurrent.

A local monitor is associated with each process for capturing all state changes. It reports to a parent monitor who is responsible for checking the state consistency relationships. Each per process monitor implements the procedures **handle_state_change** and **receive_marker_message**. (For the clarity of presentation, the algorithm is presented for a process with one sibling.)

```

procedure handle_state_change (state, current_region_number);
{
    send_parent (update_msg);
    send_sibling_region_marker (current_region_number);
    current_region_number := current_region_number + 1;
}

```

Fig. 3, **handle_state_change** procedure.

The procedure **handle_state_change**, shown in Fig. 3, is invoked by each state change in a process. It informs its parent monitor of the state change and sends a marker message to its siblings (the processes which are involved in the state consistency relationship of interest). A process may have multiple siblings. However, the region numbers of a process relative to each of its siblings must be distinguished by a sibling-process ID.

```

procedure receive_marker_message (sibling_region_number);
{
    if sibling_region_number = current_region_number then
    {
        send_parent (update_msg);
        current_region_number := current_region_number + 1;
    }
}

```

Fig. 4, **receive_marker_message** procedure.

Upon receiving a marker message, as shown in Fig. 4, the local monitor of the sibling process sends an update message to the parent monitor only if the sibling's current region number and the message's region number are equal. If the message's region number is less than the current region number, it is an indication that the recipient of the marker message has already sent an updated message for corresponding region to the parent monitor. The sibling process will never receive a marker message with a region number greater than its own current region number [2].

The update message contains the fields shown in Fig. 5. The parent monitor checks the consistency relationship of

the states which pertain to the identically numbered regions. (In the case of multiple siblings, the checking procedure is more involved.)

process ID	sibling-process ID	region number	state
------------	--------------------	---------------	-------

Fig. 5, Update message format.

IV. COMPARATIVE EVALUATION

Given a set of state consistency relationships, the statistical sampling approach cannot detect every single violation of consistency. It can only provide a statistical indication of whether such violations are occurring. The simultaneous regions approach can detect all the violations since it captures a precise (undistorted) view of the partial system state.

To estimate the computational cost of consistency checking, using the number of messages as metric, assume that all the processes in a distributed system have a mean rate of state change r . The computational cost of the state monitoring of n processes based on statistical sampling approach has the asymptotic upper bound of $O(nrk)$ where k is the oversampling factor and rk is the sampling rate. The cost of the state monitoring by establishing simultaneous regions has the asymptotic upper bound between $O(nr)$ and $O(n^2r)$ depend on the number of siblings a process may have. If each process has just one sibling, the upper bound is $O(nr)$. If each process is a sibling of all the other processes, the upper bound is $O(n^2r)$.

V. CONCLUSIONS

Providing an external indication of a growing number of errors in the internal state of a distributed system can indirectly help improve its operational reliability and availability. This paper discussed two approaches for continuous monitoring of the state of the processes in a distributed system to detect state inconsistencies with regard to the design specification. The first approach is based on sampling of the state of the processes without consideration of the state of the channels. This approach results in some distortion of state relationships, which can be filtered out by statistical techniques. The second approach captures a precise view of the partial system state by establishing simultaneous regions in distributed processes through the use of marker messages. This allows the state of the communication channels to be taken into account.

ACKNOWLEDGEMENTS

This work was supported by the Ontario Ministry of Energy, Science and Technology, and the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

- [1] J.Thai, B. Pekilis, A. Lau, R. Seviora, "Aspect-Oriented Implementation of Software Health Indicators," Proc. of 8th Asia-Pacific Software Engineering Conference, 2001, pp. 97-104.
- [2] M. Spezialetti and J.P. Kearns, "Simultaneous Regions: A Framework for the Consistent Monitoring of Distributed Systems," Proc. of 9th International Conference on Distributed Computing Systems, 1989, pp. 61-68.