Dijkstra-Scholten and Shavit-Francez Termination detection Algorithms

Rajesh Yadav Kanakabandi
Department of Computer Science
Kent State University
Kent, Ohio, USA
rkanakab@kent.edu

Abstract— **This paper discusses about Dijkstra-Scholten's termination detection algorithm for centralized networks and Shavit-Francez's generalization to decentralized networks. Also it discusses the results that are produced when the algorithm is implemented on various topologies with different number of processes and different number of initiator processes to measure the overhead to detect termination (*i.e.* number of control messages).**

Keywords-process; termination; event; computation;

I. INTRODUCTION

A computation is said to in terminal state when there is no enabled guarded command and no messages in transit (i.e. no further steps can be taken by the algorithm). It is important to detect termination of a computation since, no process will be aware of the global state of the computation.

Termination detection is done in two phases- *Detection* and *Announcement*. The former is an algorithm to detect the termination of the computation and the later one is to announce to all the processes that the computation has come to an end. However we will not be discussing about the "*Announcement"* in this paper. Dijkstra-Scholten's approach to detect termination is to detect Message termination and enforce proper termination on the basis of message termination. Dijkstra-Scholten's algorithm detects termination only on centralized network where only one initiator exists. A computation tree is maintained for all the processes that are active and involved in the computation. Each process maintains additional local parameters "father, children, number of children". The initiator is initially made the father of itself. Here two types of messages are involved: Basic message "MSG", Control message "SIG".

The algorithm goes like:

1. When a process sends a basic "MSG" message to another process, it makes the receiving process as its child.
2. When a process receives a basic "MSG" message, if the process does not have a father and is not involved in the computation, it makes the sender as its father. Else if it is already involved in the computation and has a father. It sends back a "SIG" message.
3. Upon reception of a control message ("SIG"), the process removes the sender from its children. When the number of children of the process becomes zero, it sends a control "SIG" message to its father.
4. The computation terminates when the initiator gets a signal from all of its children including it-self.

This algorithm detects termination only on networks with a single initiator. Shativ-Francez generalized the above algorithm to work on arbitrary networks with multiple initiators. Here each initiator maintains its own computation tree. All such computation trees together are called a forest. Termination is detected by another single wave that terminates when all the initiators at a terminal state.

II. EXPERIMENTAL SETUP

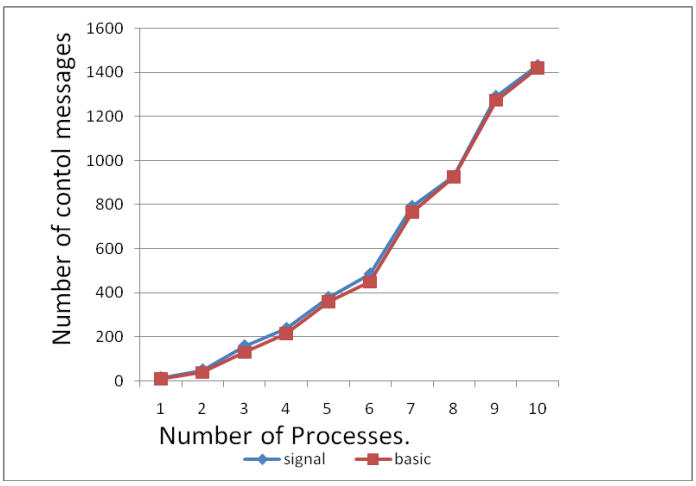
To analyze the overhead between message termination and proper termination, I chose to measure the number of control messages that have been transmitted over the network. The whole termination algorithm works on another base algorithm that broadcasts messages. Here my base algorithm is Random flood. Where, each process broadcasts at most once. Since random flood works on arbitrary networks I have a choice of selecting any kind of network but I chose to work on fully connected, ring and star topologies. The reason I selected ring and star topologies is cause of their interesting properties when

random flood algorithm runs on them. I have varied the number of processes between 5 and 50 in multiples of 5. For each case I ran tests for 1 initiator, half of the total number of processes as initiators, all the processes as initiators.

III. REASULTS AND ANALYSIS

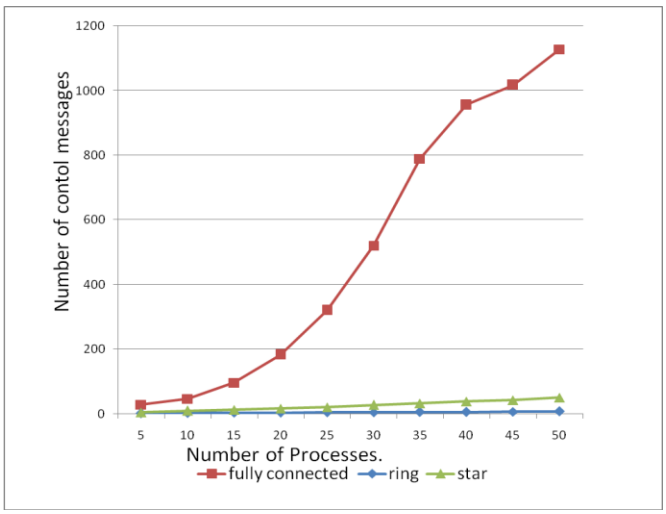
The graphs below depict the number of control messages that have been sent for different topologies with variable number of processes and variable number of initiators.

A. Number of basic messages vs number of control messages:



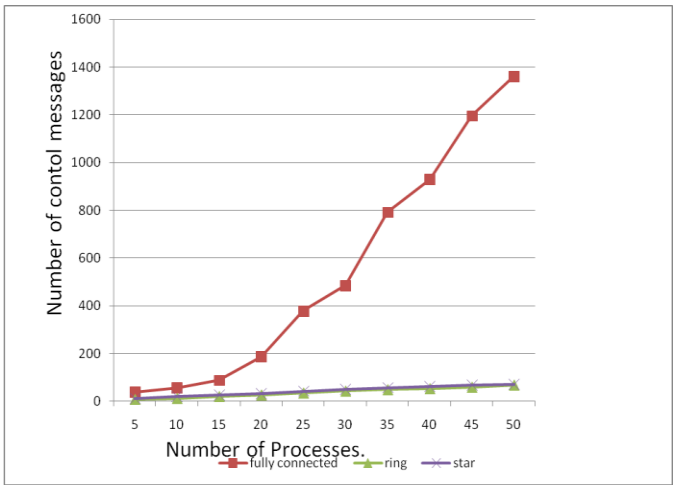
The number of control messages and number of basic messages are comparable(i.e. number of control messages depend on the number of messages send by the base algorithm). The number of basic messages sent in star and ring topology is considerably very low compared to the number of basic messages sent in fully connected networks.

B. One Initiator on fully connected, ring and star topologies:



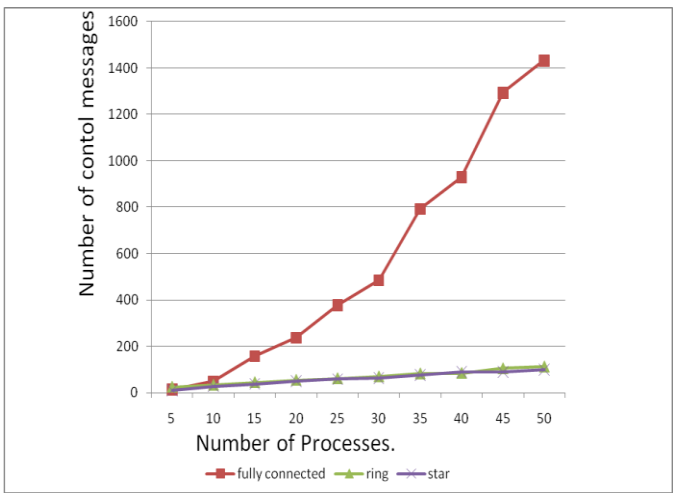
With a single initiator in the network, the number of control messages increases with increase in the number of processes. For ring and star topologies the number of messages transmitted could be very less cause of its topology. (if the initiator chooses to send message to only one neighbor and the receiver sends only 1 message back to the initiator, total number of messages will be only 2). The numbers of basic messages transmitted and number of control messages are comparable.

C. Half of the processes as initiators:



With half of the processes as initiators, the number of messages sent on ring topology and star topology increases when compared to one initiator. Consider if every alternate process is an initiator for ring topology and the central process and n/2-1 other processes are initiators for a network with star topology.

D. All the processes acting as initiators on fully connected star and ring topologies:



Even in the case of all initiator processes, the number of signal messages sent in networks with star and ring topologies is more than that in case of a single initiator.

IV. CONCLUSION

From the data analyzed, we understand that the number of signal messages is always comparable with the number of basic messages. The behavior of the base algorithm caused larger impact on the number of signal messages which is the overhead in detecting Proper termination. Star and ring topologies have interesting topologies that restrict the number of base messages sent by the base algorithm. And thus restricting the number of control messages. Number of control messages is comparable to the number of basic messages.

V. FUTURE WORK

I would like to test the algorithm in other topologies like tree. I would also like to test the termination detection algorithm on algorithms other than random flood. In order to measure a better over head I also want to measure the number of control messages that are sent after the occurrence of message termination.

REFERENCES

[1] Edsger W. Dijkstra, C. S. Scholten “Termination Detection for Diffusing Computations” 1980.

[2] Gerard Tel “Introduction to Distributed Algorithms”, Cambridge University Press, 1st edition.

[3] Nir Shavit, Nissim Francez “A new approach to detection of locally indicative stability” 1986.

[4] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.