International Journal of Digital Information and Wireless Communications (IJDIWC) 2(1): 82-85
The Society of Digital Information and Wireless Communications, 2012(ISSN 2225-658X)

A new converge cast algorithm: Application of procedural distributed recursive wave model and feedback function

Ha Quoc Trung

School of ICT, Hanoi University of Science and Technology. Vietnam
trunghq@soict.hut.edu.vn

Abstract— *The distributed recursive wave is a programming model based on distributed procedure call, allowing distributed algorithm definition using central algorithm. For returning result to calling process, a feedback function is used. Introducing feedback function makes the DRW more complex, but it allows more control on the execution of the DRW depending on the application needs. With this solution, DRW can be applicable for collecting, not only for propaganda of information. This DRW with feedback function is applied in the case of network convergecast problem. Convergecast is one of group communication tasks in distributed systems: convergecast, broadcast, multicast and unicast. ConvergeCast algorithms allow concentrating network information for calculating of global parameters. When the Converge Cast is required, information about the network itself such as routing table, network topology is not available. Distributed recursive wave is a way to construct algorithms that can be customized only by local parameters and functions. Distributed recursive wave then is suitable for distributed systems having network information dynamically changing without global information. Convergecast can be raw information collection or aggregation, so the feedback function DRW is suitable to apply to adapt both cases.*

This article presents a new converge cast algorithm. This algorithm is experimentally implemented using Shell Script.

Index Terms— *Distributed Converge Cast, Distributed Recursive Waves*

1. INTRODUCTION

Converge cast is one of group communication tasks used when the communication network of distributed systems is installed or reconfigured. In this case, the global information of the system is not ready, so the protocol can only rely on the local information in each process. This feature is becoming more important for the continuously changing systems, depending on the objective and natural factors such as sensor networks and mobile networks.

Distributed recursive wave is a model enabling the construction of distributed algorithms with simplicity of the recursive algorithm, and allowing customizing the implementation of recursive algorithms with local parameters. Recursive wave is based on the principle of echo, being made in two phases: forward phase and echo phase. In [1] a distributed recursive wave based broadcast algorithm uses only forward phase because of the nature of the broadcast algorithm. In [4,5] the authors have proved that distributed recursive wave is consistent in distributed systems with no global information.

This article presents the analysis and design of an algorithm to collect information (forward phase and echo phase) in a distributed system with the assumption that global information (network topology) is not available. The distributed system is assumed to be asynchronous.

The paper is structured as follows: Section 1 gives the general introduction. Section 2 presents the model of distributed recursive wave and the way of analysing and designing distributed algorithms using distributed recursive wave. Section 3 presents the algorithm to collect new information. Section 4 describes the implementation of the algorithm. Finally, the conclusions are found in section 5.

2. DISTRIBUTED RECURSIVE WAVE

Programmatic model of distributed recursive wave

The above introduced model of distributed recursive wave allow us to design and evaluate the algorithms base on distributed recursive wave. In order to build a program implementing the algorithms, we need more programmatic model. The model is shown on Figure 1.

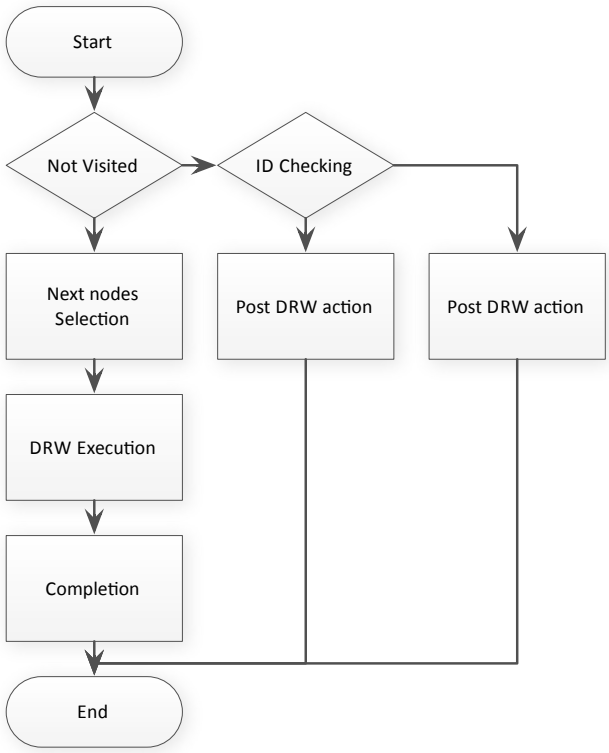

Figure 1: Programmatic Model of Distributed Recursive Wave

In this model, the collection of result can be executed at the initial node, or partly at the intermediate nodes. This possibility allows us to customize the computation of the final result: it can be done at the end on the initial node, or it can be executed partly when each node finished all outgoing calls. In this model of the distributed recursive wave, the distributed call is a function, so in the returning phase, we face a problem of collecting results.

In [18] presents a feedback function for collecting results. The collection can be raw, i.e stores all results in an array. The final result is compute at the root node from all collected intermediate results. Another way is to collect information and process computation of partial results at each intermediate note, send only the results to the parent. In the general case of ConvergeCast, we don't know which aggregate function will be process on the collected information, so we use the raw collection approach. In this article, we use the feedback function allowing customization the distribution of the computation quantity on nodes.

In [19], to accelerate the processing of the distributed recursive wave, to avoid waiting, a non-blocking recursive call is used and compared with a blocking call solution.

In next section, we'll present the convergecast algorithm with feedback function. The function uses blocking call mechanism.

3. CONVERGECAST ALGORITHM USING DISTRIBUTED RECURSIVE WAVE

With the requirements stated in section 1 and the ways to design the algorithm as in section 2, the convergecast algorithm will be developed. First, the local variables of each process will be determined including:

- A variable NB: presenting neighborhood nodes of the node being watched. This variable can also be expanded to contain more information about the link between the current node and neighbor nodes. The value of variables can be determined locally.

```
Function ConvergeCast(
                Data,
                ListOfProcessID) Of
        List
if not Visited then
          R=Converge(Select(Neighbo
 rhood));
          //Raw collection
          Visited=True;
          Else
          Return FeedBack(R);
          //Aggregation possible
          end if;
end procedure;

if ProcessID==root then
          Visited=False;
          Return ConvergeCast();
end if;
```

Figure 2: Convergecast with feedback function

- A visited variable is used to determine the status of recursive waves (finished / not finished).
- A variable R is used to store the results returned from the process of gathering information. This variable is only valid when the forward phase ends.

Forward phase is designed according to the strategies of application. On the basis of recursive waves, there can be the following possibilities:

- One selection: send information to only one process.
- All selection: send a message to all neighbor processes.
- Random selection.
- Selection according to network data of the local process.

The selection is generally expressed by a function Select. With the arguments above, information collecting algorithm is described as shown in Figure 2: Convergecast function.

The presence of feedback function allows customization of computing load distribution. In case of general

convergecast, the result and input dimension are almost the same, intermediate computing does not help to improve the bandwidth consummation; the feedback function will be a store-collect function. In case of aggregation convergecast, the result has a small tail, so this aggregation must be integrated in to the feedback function to decrease the data size sent.

The complexity of the algorithm depends on the function Select. In case of all selection, the complexity is $O(d)$ in which d is the diameter of the network. The function Select can be adjusted to get the implementation process more efficient, but if the ratio of the number of selected processes to all of the neighbor processes has a none-zero value as lower limits, the time complexity is still $O(d)$.

4. IMPLEMENTATION

In order to implement the algorithm we can use a simulated or real environment. However, in both cases, distributed recursive calls need to be supported by the system. We need to find a distributed system that allows:

- Sending a script to a remote computer
- Activating and executing the script.

For these reasons, the author has selected Shell Script and Linux operating system as execution environment. With the assumption that above actions are supported, function ConvergeCast() in Figure 2 becomes ConvergeCastImpl() with shell language as in Figure 3: Implementing ConvergeCast by Shell.

```
#!/bin/bash
if [ "$VISITED" = "YES" ]; then  echo "Collected
Information"
  cat neighbours >return.dat
  else
for i in $(cat neighbors); do
  scp convergecast root@$i:~/run/convergecast
  ssh root@$i 'chmod +x /run/convergecast'
  ssh root@$i '/run/convergecast'
  ssh root@$i '/run/feedback'
  ssh root@$i 'chmod +x /run/feedback'
   ssh root@$i 'cat return.dat   | /run/feedback' >
temp.dat
  scp root@$i:~/run/temp.dat temp
  cat temp>>return.dat
  export VISITED = "YES"
done
fi
```

Figure 3: Implementing ConvergeCast with FeedBack by Shell

The program is installed on computers running Linux operating system, which is configured to connect together using OpenSSL without password. An overlay network is built by creating on each computer a neighbor file containing a list of addresses of neighbor nodes. This is the only information about the configuration of the system. A return.dat file used to store returned results. The program runs correctly, as all the machines in the overlay network configurations are collected on the root machine. In experimenting this program, I have also constructed several scripts that:

- Accepts topology configuration text file as the input, configure the whole system conform to the topology;
- Accepts a shell script file as input for DRW
- Accepts command line option to run the algorithm with predefined number iteration, statistically compares the obtained results.

The program is tested on various types of network and function. We have test on topology and max id, computation. To test that the program is correct, a single machine is used with multiple IP addresses. Another scenario is to use 5 virtual machines on a physical machine. The program is executed with 5 topologies. The running time and the correctness of the result are metrics. The results are shown on the Table 1 (for max id) and Table 2 (for topology)

Table 1: Running performance for max id calculation

Topology	Diameter	Number of executions	Correct result?	App. Time (ms)
1	2	20	Yes	120
2	1	20	Yes	100
3	1	20	Yes	110
4	3	20	Yes	220
5	4	20	Yes	250

Table 2: Running performance for topology calculation

Topology	Diameter	Number of executions	Correct result?	App. Time (ms)
1	2	20	Yes	130
2	1	20	Yes	105
3	1	20	Yes	120
4	3	20	Yes	225
5	4	20	Yes	250

Unfortunately, we are not able to run the algorithm on a larger system, so the result can show only that the algorithm is correct. In addition, timing of the OpenSSL activities influence strongly on. We hope that in the future we'll be able to experiment this algorithm on a simulation environment such as Planet-Lab or G-Lab, so that the results will be more persuasive.

5. CONCLUSIONS

The article has presented the model of distributed recursive waves and how to design and analyze a distributed algorithm based on distributed recursive wave. A new ConvergeCast algorithm based on distributed recursive wave has been presented. The complexity of the algorithm does not change in comparison with other algorithms, but the