Revisit Byzantine Agreement in Hybrid Fallible Virtual Subnet Network

K.Q. Yan kqyan@cyut.edu.tw
S.C. Wang* **(Corresponding author)** scwang@cyut.edu.tw
G.Y. Zheng s9414606@cyut.edu.tw
Chaoyang University of Technology, Taiwan, R.O.C

Abstract

A Mobile Ad-hoc Network (MANET) may suffer from various types of processor failure. In order to enhance the fault-tolerance and reliability of the Mobile Ad-hoc Networks, the Byzantine Agreement problem in the virtual subnet network of MANET is revisited in this paper. The proposed protocol is called Hybrid Agreement Protocol for processor (HAP_p) which can make each correct mobile processor reach an agreement value to cope with the faulty component in the virtual subnet network.

Keywords: Byzantine agreement, fault-tolerance, distributed system, virtual subnet, mobile ad-hoc network

1. Introduction

The mobile ad-hoc networks have attracted significant attentions recently due to its features of infrastructure less, quick deployment and automatic adaptation to changes in topology. Therefore, mobile ad-hoc network suits for military communication, emergency disaster rescue operation, and law enforcement [1].

The reliability of the mobile processor is one of the most important aspects in mobile ad-hoc networks. In order to provide a reliable environment in a mobile ad-hoc network, we need a mechanism to allow a set of mobile processors to agree on an agreement value [8]. The Byzantine Agreement (BA) problem [2.3.6-9] is one of the most fundamental problems to reach an agreement value in a distributed system.

The original BA problem defined by Lamport et al. [3] is assumed as follows: (BA_1): There are n processors in a synchronous distributed system, where n is a constant and $n \geq 4$. (BA_2): Each processor can communicate with each other through reliable fully connected network. (BA_3): One or more of the processors might fail, so a faulty processor may transmit incorrect message(s) to other processors. (BA_4): After message exchange, all correct processors should reach a common agreement, if and only if the number of the faulty processors t is less than one-third of the total number of processors in the network ($t \leq (n-1)/3$). Based on these assumptions, the BA requirement can be satisfied when the following constraints are met:

Agreement: All correct processors agree on an agreement value.

Validity: If the source processor is correct, then all correct processors agree on the initial value sends by the source processor.

The traditional BA problem was focused on the fixed and well-defined network [2.3.6-9]. However, the network structure of mobile ad-hoc network is not fixed, and it can change its topology at any time by the feature of mobility. Hence, the tradition solutions for the BA problem were not suited for the mobile ad-hoc network.

In this study, the BA problem in the virtual subnet network of MANET is revisited. The proposed protocol is called as the Hybrid Agreement Protocol for processor (HAP_p). HAP_p can make each correct mobile processor in the virtual subnet network of MANET reach an agreement value.

The remainder of this paper is organized as follows. Section 2 discusses the virtual subnet network and the failure type of a fallible mobile processor. Section 3 illustrates the concept of HAP_p by an example. Finally, we conclude in Section 4.

2. Related work

Recent advances in technology have provided portable processors with wireless interfaces that allow networked communication among mobile users. The computing environment, which refers to as mobile computing, no longer requires users to maintain a fixed and universally known position in the network and enables almost unrestricted mobility.

Furthermore, each processor has highly mobility in MANET, it will causes network topology changes of the wireless mobile network. In addition, the limitation of power leads processors disconnects mobile unit frequently in order to save power consumption. On the other hand, processors may immigrate into the network or move away from the network at any time. In MANET, virtual subnet is composed of several groups by overlay network approach [1.5]. Fig. 1 is a topology of virtual subnet under the MANET. There are two situations that the processors communicate underlying virtual subnet:

Situation 1. Processors in the same group communicate to each other directly by virtual backbone.

Situation 2. Processors in different groups are exchanging messages with each other via virtual subnet or physical communication media (such as the agent-based).

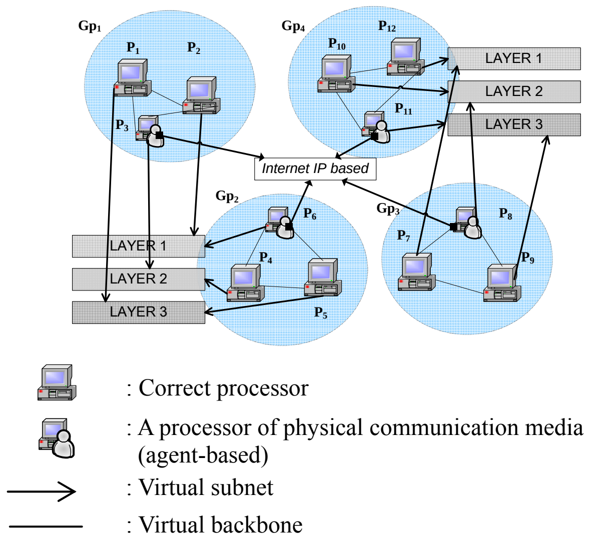


Fig. 1. The topology of virtual subnet

A mobile processor is said to be correct if it follows the protocol specifications during the execution of a protocol; otherwise, the mobile processor said to be faulty. The symptoms of mobile processor failure can be classified into two categories: dormant fault and malicious fault (also called as Byzantine fault) [6]. The dormant faults of a fallible mobile processor are crashes and omission. A crash fault happens when a processor is broken. An omission fault takes place when a processor fails to transmit or receive a message on time or at all. However, the behavior of a mobile processor with malicious fault is unpredictable. A mobile processor with malicious fault may work in coordination with other faulty mobile processor to prevent other correct mobile processors to reach an agreement value. In this study, we will solve the BA problem with mobile processor with malicious and dormant fault.

3. The proposed protocol

This paper proposes a new protocol, Hybrid Agreement Protocol for processor, called HAP_p, to solve the BA problem due to faulty processor(s), which may send wrong messages to influence the system to reach agreement in a virtual subnet.

The notations and parameters of our protocols HAP_p for the virtual subnet network are shown in follow:

- Let x, y be the group identifier where $1 \leq x, y \leq g$ and $g \geq 4$.
- Let f_m be the total number of malicious faulty processors.
- Let f_d be the total number of dormant faulty processors.
- Let F_{Gm} be the maximum number of malicious faulty groups allowed.
- Let F_{Gd} be the maximum number of dormant faulty groups allowed.
- Let T_{FP} is the total number of allowable faulty processors, $T_{FP} = f_m + f_d$.
- Let T_{FG} is the total number of allowable faulty groups, $T_{FG} = F_{Gm} + F_{Gd}$.
- Let η_x is the number of processors in G_x, $0 \leq x \leq g$.
- Let c be the connectivity of the virtual subnet network, where c is $g - 1$.

The proposed protocol HAP_p is organized as two phases with the procedure TRANSMISSION. In the first round of the message exchange phase, the source processor sends its initial value to all processors by using TRANSMISSION, and then receiver processor stores the received value in the root of its message-gathering tree (mg-tree). The mg-tree is a tree structure that is used to store the received message [9]. If the source processor is dormant fault, then the value "λ^0" has replaced the initial value from source processor. After the first round of message exchange phase ($\theta > 1$), each processor without source processor uses TRANSMISSION to transmit the value at level $\theta - 1$ in its mg-tree to all processors. If the value at level $\theta - 1$ is λ^i (the value λ^i is used to report absent value), then the value λ^i will be replaced by λ^{i+1}, where $0 \leq i \leq T_{FG} - 1$. At the end of each round, the receiver processor uses the function RMAJ on it received VMAJ values, which are from the same group by TRANSMISSION, to get a single value. Moreover, each receiver processor stores the received messages (VMAJ values) and function RMAJ value in its mg-tree.

Subsequently, in the decision making phase, each processor without the source processor reorganizes its mg-tree into a corresponding information-collecting tree (ic-tree). The ic-tree is a tree structure that has used to store a received message without repeated group names [9]. Therefore, the common value VOTE(s) has obtained by using function VOTE on the root s of each processor's ic-tree. The function VOTE counts the non-value λ^0 (expert the last level of the ic-tree) for all vertexes at the θ-th level of an ic-tree, where $1 \leq \theta \leq T_{FG} + 1$. The conditions in the function VOTE are similar to conventional majority vote [7]. The detail steps of the proposed protocol HAP_p has presented in Fig. 2.

HAP_p (source processor with initial value v_s)

Pre-Execute. Computes the number of rounds required $\theta = \lfloor (g-1)/3 \rfloor + 1$

Message Exchange Phase:

***Case* $\theta = 1$**, run

A) The source processor transmits its initial value v_s to each group's processors by using TRANSMISSION.

B) Each receiver processor obtains the value and stores it in the root of its mg-tree.

C) If the source processor is dormant fault, the value "λ^0" has replaced the initial value from source processor.

***Case* θ** > 1, run

A) Each processor without the source processor uses TRANSMISSION to transmit the values at level $\theta - 1$ in its mg-tree to each group's processors. If the value at level $\theta - 1$ is "λ^i", and the value λ^i will be replaced by λ^{i+1}, where $0 \leq i \leq T_{FG} - 1$.

B) Each receiver processor applies RMAJ on its received messages and stores RMAJ value in the corresponding vertices at level θ of its mg-tree.

```
Decision Making Phase:
Step 1:
  Reorganizing the mg-tree into a corresponding ic-tree. (The vertices with repeated group names are deleted).
Step 2:
  Using function VOTE on the root s of each processor's ic-tree, then the common value VOTE(s) has obtained.
Function RMAJ(V)
  1. The majority value in the vector $V_i = [v_1, \ldots, v_{\eta x-1}, v_{\eta x}]$, if it exists.
  2. Otherwise, choosing a default value ($\phi$).
Function VOTE($\mu$)
  1. If the $\mu$ is a leaf or the number of value $\lambda^0$ is $3 * (T_{FG} - \theta + 1) + (g - 1) \% 3$, then output $\mu$.
  2. Else if the majority value is not existed, then output $\phi$.
  3. Else if the majority value is $\lambda^i$, where $1 \le i \le T_{FG}$, then output $\lambda^{i-1}$.
     Otherwise, output m, where $m \in \{0, 1\}$
```

Fig. 2. HAP_p protocol

The procedure TRANSMISSION used to transmit messages in HAP_p. Based on the distinctions of virtual subnet network [1.5], TRANSMISSION can provide a virtual channel for each processor to transmit messages to each other without influence from faulty inter-communication medium. The description of TRANSMISSION is shown in Fig. 3.

```
Procedure TRANSMISSION
Definition:
  1. For the virtual subnet, each processor has the common knowledge of entire graphic information $\hat{G} = (E, Gp)$, where Gp is the set of groups in the network and E is a set of group pairs $(Gp_x, Gp_y)$ indicating a physical communication medium (the sensing is covered) between group $Gp_x$ and group $Gp_y$.
  2. Each processor communicates with all other processors via virtual subnet, virtual backbone or physical communication media [1][5].
  3. The processor plays sender, receiver or agent, depends on the behavior of which kinds of transmission [1].
  4. The agent-based processor cannot garble the message between the sender processor and receiver processor; this assumption has achieved by the technology of encryption (such as RSA [4]).
Step 1:
  The sender processor i ($1 \le i \le n$) transmits the value $v_i$ to the receiver group.
Step 2:
  If the group-disjoint path from sender processor to destination group passes through any dormant faulty processor or if the sender processor has dormant faults, then stores $\lambda^0$ itself.
Step 3:
  The processors in the receiver group take the local majority value from the same group paths and then construct the vector $V_i = [v_{path\ 1}, v_{path\ 2}, \ldots, v_{path\ c-1}, v_{path\ c}]$.
Step 4:
  The processors in the destination group apply VMAJ on vector $V_i$.
Function VMAJ (for each vector $V_i$)
  1. Count the received value: Take the majority
  2. If the majority value is $\lambda^0$ and the number of value $\lambda^0$ is greater than or equal to $c - \lfloor (g-1)/3 \rfloor$, then output the value $\lambda^0$.
  3. Else set majority value m, where $m \in \{0, 1\}$
       If the majority value does not exist, then
           Output the majority value $\lambda^0$.
     Otherwise, output the majority m, where $m \in \{0, 1\}$.
```

Fig. 3. TRANSMISSION

An example is given to execute our protocol HAP_p, the virtual subnet network is shown in Fig. 4(a). There are 24 processors falling into seven groups. Gp_1 includes source processor P_s, P_1 and P_2. Gp_2 includes P_3, P_4, P_5 and P_6. Gp_3 includes P_7, P_8, P_9 and P_{10}. Gp_4 includes P_{11} and P_{12}. Gp_5 includes P_{13} and P_{14}. Gp_6 includes P_{15} and P_{16}. P_{17}, P_{18}, P_{19}, P_{20} and P_{21} belong to Gp_7, P_{22} and P_{23} belong to Gp_8.

In BA problem, the worst situation [3] is that the source does not honest anymore. Simply, here the worst case of the example, suppose the source processor P_s is malicious fault, which means P_s may send arbitrarily different values to different groups. Therefore, in order to solve the BA problem among correct processors of the example, HAP_p requires θ ($\lfloor (g-1)/3 \rfloor + 1$) rounds of message exchange phase.

In HAP_p, Pre-Execute counts the number of rounds required before message exchange phase. There is in need of three rounds to message exchange for the example.

The source processor P_s uses TRANSMISSION to transmit messages to all other processors in the first round of the message exchange phase. The message obtained of each correct processor is listed in Fig. 4(b). In the σ-th ($1 < \sigma \le \theta$) round of message exchange, except for the source processor, each processor uses TRANSMISSION to transmit RMAJ values at the ($\sigma - 1$)-th level in its mg-tree to all the others and itself. Subsequently, each receiver processor applies RMAJ to its received messages and stores the received messages (VMAJ values) and RMAJ values at the corresponding vertices at level σ of its mg-tree. The mg-tree of correct processor P_1 at the second and final round in the message exchange phase is shown in Fig. 4(c) and 4(d).

After the message exchange phase, the tree structure of each correct processor is converted from mg-tree to ic-tree by deleting the vertices with duplicated names. The example ic-tree has showed in Fig. 4(e). Eventually, using the function VOTE to root the value s for each correct processor's ic-tree {VOTE(s) = VOTE(s1), …, VOTE(s8) = 1}, an agreement value 1 can be obtained, as shown in Fig. 4(f), and the decision making phase has completed.

4. Conclusion

In the previous work, the complex networks had studied in a branch of mathematics known as graph theory. The network topology developed in recent years [1.5] shows a mobile feature. The previous protocols such as [2.3.6-9] cannot adapt to solve BA problem in MANET, and none of the BA protocol is designed for the virtual subnet of MANET. Therefore, the BA problem in virtual subnet network with dual failure mode on fallible processor has revisited. The proposed protocol can tolerate the most damaging failure type of fallible processors. The proposed HAP_p can take the minimum number of required rounds to achieve an agreement, and tolerate the maximum number of faulty components.

Furthermore, in a generalized case, there are not only processors may be crash, omission or malicious, but also communication medium. On other hand, our protocol will be extend to solve when dormant or malicious communication media or processors are existed simultaneously underlying virtual subnet network of MANET in future work.

Reference

[1] T.C. Chiang, H.M. Tsai and Y.M. Huang, A Partition Network Model for Ad Hoc Networks, IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Vol. 3, pp.467-472, 2005.

[2] M. Fischer, The Consensus Problem in Unreliable Distributed Systems (A Brief Survey), Technical report, Department of Computer Science, Yale University, 2000.

[3] L. Lamport, R. Shostak and M. Pease, The Byzantine General Problem, ACM Transactions on Programming Language and Systems, Vol. 4, No. 3, pp. 382-401, 1982.

[4] B. Lehane and L. Doyle, Shared RSA Key Generation In A Mobile Ad Hoc Network, in the Military Communications of IEEE Conference, Vol. 2, pp 814-819, 2003.

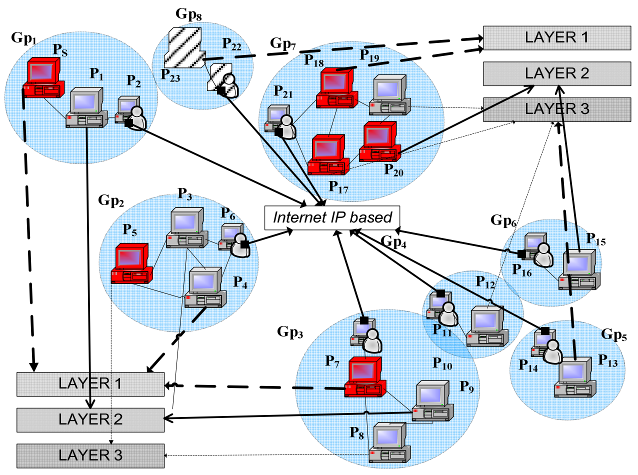
[5] M. Min, F. Wang, D. Z. Du and P. M. Pardalos, A Reliable Virtual Backbone Scheme in Mobile Ad-hoc Networks, IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 60-69, 2004.

[6] M. Pease, R. Shostak and L. Lamport, Reaching Agreement in Presence of Faults, Journal of ACM, Vol. 27, No. 2, pp. 228-234, 1980.

[7] S. C. Wang and K. Q Yan, Revisit Consensus Problem on Dual Link Failure Modes, in the Proceedings of International Computer Software & Applications Conference, Vienna, Austria, pp. 84-89, 1998.

[8] S.C. Wang, K.Q. Yan and G.Y. Zheng, Reaching Consensus Underlying Fallible Virtual Subnet of Mobile Ad-Hoc Network, Twelfth Mobile Computing Workshop, pp. 257-263, 2006.

[9] K.Q. Yan, S.C Wang, Group Byzantine Agreement, in Computer Standards & Interfaces, pp. 75-92, 2005.



The symbol notations:

🖥	: Correct processor
🟥	: Malicious faulty processor
▨	: Dormant faulty processor
👤	: Physical communication media (agent-based)

The messages sent from the source processor by TRANSMISSION and then start to execute HAP_p.

- The source processor, P_s is a malicious faulty processor.
- P_s sends value 1 to Gp_2, Gp_4, Gp_5, Gp_6, Gp_7 and Gp_8.
- P_s sends value 0 to Gp_1 and Gp_3.

Fig. 4(a). The initial status of executing HAP_p in a virtual subnet of MANET.

	Level 1 *root* *s*
Gp_1's correct processors	0
Gp_2's correct processors	1
Gp_3's correct processors	0
Gp_4's correct processors	1
Gp_5's correct processors	1
Gp_6's correct processors	1
Gp_7's correct processors	1
Gp_8's correct processors	1

→VMAJ values

Fig. 4(b). The mg-tree of each processor at the 1st round

Level 1 *root* *s*	*Level 2*	Function RMAJ	
Val(S)=1	*s1*	0	(0,0)
	s2	1	(1,1,***0***,1)
	s3	0	(0,***0***,0,0)
	s4	1	(1,1)
	s5	1	(1,1)
	s6	1	(1,1)
	s7	0	(***0***,***0***,1,***0***,1)
	s8	λ^0	(λ^0, λ^0)

→VMAJ values

Fig. 4(c). The mg-tree of correct processor P_1 at the 2nd round

Level 1 *root*	*Level 2*	*Level 3*	Function RMAJ	
s *0*	*s1* 0(0)	*s11*	0	(0,0)
		s12	0	(0,0,***0***,0)
		s13	0	(0,***1***,0,0)
		s14	0	(0,0)
		s15	0	(0,0)
		s16	0	(0,0)
		s17	1	(***1***,***1***,1,***0***,1)
		s18	λ^0	(λ^0,λ^0)
	s2 1(1,1,1,1)	*s21*	1	(1,1)
		s22	1	(1,1,***1***,1)
		s23	1	(1,***1***,1,1)
		s24	1	(1,1)
		s25	1	(1,1)
		s26	1	(1,1)
		s27	0	(***0***,***0***,1,***0***,1)
		s28	λ^0	(λ^0,λ^0)
	s3 0(0,0,0,0)	*s31*	0	(0,0)
		s32	0	(0,0,***1***,0)
		s33	0	(0,***1***,0,0)
		s34	0	(0,0)
		s35	0	(0,0)
		s36	0	(0,0)
		s37	0	(***0***,***0***,1,***0***,1)
		s38	λ^0	(λ^0,λ^0)
	s4 1(1,1)	*s41*	1	(1,1)
		s42	1	(1,1,***0***,1)
		s43	1	(1,***1***,1,1)
		s44	1	(1,1)
		s45	1	(1,1)
		s46	1	(1,1)
		s47	1	(***1***,***1***,1,***0***,1)
		s48	λ^0	(λ^0,λ^0)
	s5 1(1,1)	*s51*	1	(1,1)
		s52	1	(1,1,***1***,1)
		s53	1	(1,***0***,1,1)
		s54	1	(1,1)
		s55	1	(1,1)
		s56	1	(1,1)
		s57	0	(***0***,***0***,1,***0***,1)
		s58	λ^0	(λ^0,λ^0)
	s6 1(1,1)	*s61*	1	(1,1)
		s62	1	(1,1,***1***,1)
		s63	1	(1,***1***,1,1)
		s64	1	(1,1)
		s65	1	(1,1)
		s66	1	(1,1)
		s67	1	(***1***,***1***,1,***1***,1)
		s68	λ^0	(λ^0,λ^0)
	s7 0(0,0,1,0,1)	*s71*	0	(0,0)
		s72	1	(1,1,***1***,1)
		s73	0	(0,***0***,0,0)
		s74	1	(1,1)
		s75	0	(0,0)
		s76	1	(1,1)
		s77	0	(***0***,***0***,1,***0***,1)
		s78	λ^0	(λ^0, λ^0)
	s8 $\lambda^0(\lambda^0, \lambda^0)$	*s81*	λ^1	(λ^1,λ^1)
		s82	λ^1	$(\lambda^1,\lambda^1,\lambda^1,\lambda^1)$
		s83	λ^1	$(\lambda^1,\lambda^1,\lambda^1,\lambda^1)$
		s84	λ^1	(λ^1,λ^1)
		s85	λ^1	(λ^1,λ^1)
		s86	λ^1	(λ^1,λ^1)
		s87	0	(***0***,***0***, λ^0,***0***,λ^0)
		s88	λ^0	(λ^0,λ^0)

Fig. 4(d). The final mg-tree of processor P_1 after the message exchange phase

The tree structure has converted from mg-tree to ic-tree by erasing the vertices with repeated names.

Level 1 *root*	*Level 2*	*Level 3*	Function RMAJ	
s *0*	*s1* 0(0)	*s12*	0	(0,0,***0***,0)
		s13	0	(0,***1***,0,0)
		s14	0	(0,0)
		s15	0	(0,0)
		s16	0	(0,0)
		s17	1	(***1***,***1***,1,***0***,1)
		s18	λ^0	(λ^0,λ^0)
	s2 1(1,1,1,1)	*s21*	1	(1,1)
		s23	1	(1,***1***,1,1)
		s24	1	(1,1)
		s25	1	(1,1)
		s26	1	(1,1)
		s27	0	(***0***,***0***,1,***0***,1)
		s28	λ^0	(λ^0,λ^0)
	s3 0(0,0,0,0)	*s31*	0	(0,0)
		s32	0	(0,0,***1***,0)
		s34	0	(0,0)
		s35	0	(0,0)
		s36	0	(0,0)
		s37	0	(***0***,***0***,1,***0***,1)
		s38	λ^0	(λ^0,λ^0)
	s4 1(1,1)	*s41*	1	(1,1)
		s42	1	(1,1,***0***,1)
		s43	1	(1,***1***,1,1)
		s45	1	(1,1)
		s46	1	(1,1)
		s47	1	(***1***,***1***,1,***0***,1)
		s48	λ^0	(λ^0,λ^0)
	s5 1(1,1)	*s51*	1	(1,1)
		s52	1	(1,1,***1***,1)
		s53	1	(1,***0***,1,1)
		s54	1	(1,1)
		s56	1	(1,1)
		s57	0	(***0***,***0***,1,***0***,1)
		s58	λ^0	(λ^0,λ^0)
	s6 1(1,1)	*s61*	1	(1,1)
		s62	1	(1,1,***1***,1)
		s63	1	(1,***1***,1,1)
		s64	1	(1,1)
		s65	1	(1,1)
		s67	1	(***1***,***1***,1,***1***,1)
		s68	λ^0	(λ^0,λ^0)
	s7 0(0,0,1,0,1)	*s71*	0	(0,0)
		s72	1	(1,1,***1***,1)
		s73	0	(0,***0***,0,0)
		s74	1	(1,1)
		s75	0	(0,0)
		s76	1	(1,1)
		s78	λ^0	(λ^0, λ^0)
	s8 $\lambda^0(\lambda^0, \lambda^0)$	*s81*	λ^1	(λ^1,λ^1)
		s82	λ^1	$(\lambda^1,\lambda^1,\lambda^1,\lambda^1)$
		s83	λ^1	$(\lambda^1,\lambda^1,\lambda^1,\lambda^1)$
		s84	λ^1	(λ^1,λ^1)
		s85	λ^1	(λ^1,λ^1)
		s86	λ^1	(λ^1,λ^1)
		s87	0	(***0***,***0***, λ^0,***0***,λ^0)

Fig. 4(e). The ic-tree of processor P_1

- ✓ VOTE($s1$) = (VOTE($s12$), VOTE($s13$), VOTE($s14$), VOTE($s15$), VOTE($s16$), VOTE($s17$), VOTE($s18$))
  VOTE($s1$) = (0, 0, 0, 0, 0, 1, λ^0) = 0
- ✓ VOTE($s2$) = (VOTE($s21$), VOTE($s23$), VOTE($s24$), VOTE($s25$), VOTE($s26$), VOTE($s27$), VOTE($s28$))
  VOTE($s2$) = (1, 1, 1, 1, 1, 0, λ^0) = 1
- ✓ VOTE($s3$) = (VOTE($s31$), VOTE($s32$), VOTE($s34$), VOTE($s35$), VOTE($s36$), VOTE($s37$), VOTE($s38$))
  VOTE($s3$) = (0, 0, 0, 0, 0, 0, λ^0) = 0
- ✓ VOTE($s4$) = (VOTE($s41$), VOTE($s42$), VOTE($s43$), VOTE($s45$), VOTE($s46$), VOTE($s47$), VOTE($s48$))
  VOTE($s4$) = (1, 1, 1, 1, 1, 1, λ^0) = 1
- ✓ VOTE($s5$) = (VOTE($s51$), VOTE($s52$), VOTE($s53$), VOTE($s54$), VOTE($s56$), VOTE($s57$), VOTE($s58$))
  VOTE($s5$) = (1, 1, 1, 1, 1, 0, λ^0) = 1
- ✓ VOTE($s6$) = (VOTE($s61$), VOTE($s62$), VOTE($s63$), VOTE($s64$), VOTE($s65$), VOTE($s67$), VOTE($s68$))
  VOTE($s6$) = (1, 1, 1, 1, 1, 1, λ^0) = 1
- ✓ VOTE($s7$) = (VOTE($s71$), VOTE($s72$), VOTE($s73$), VOTE(74), VOTE($s75$), VOTE($s76$), VOTE($s78$))
  VOTE($s7$) = (0, 1, 0, 1, 0, 1, λ^0) = ϕ
- ✓ VOTE($s8$) = (VOTE($s81$), VOTE($s82$), VOTE($s83$), VOTE(84), VOTE($s85$), VOTE($s86$), VOTE($s87$))
  VOTE($s7$) = (0, 1, 0, 1, 0, 1, λ^0) = ϕ

- ✓ VOTE(s) = (VOTE($s1$), VOTE($s2$), VOTE($s3$), VOTE($s4$), VOTE($s5$), VOTE($s6$), VOTE($s7$), VOTE($s8$))
  VOTE(s) = (0, 1, 0, 1, 1, 1, ϕ, ϕ) = 1

Fig. 4(f). The common value VOTE(s) by correct processor P_1

Fig. 4. An example of HAP_p execution (cont')