Selecting a distributed agreement algorithm

Robert John Walters
University of Sothampton
Highfield, Southampton,
United Kingdom
rjw1@ecs.soton.ac.uk

Peter Henderson
University of Sothampton
Highfield, Southampton,
United Kingdom
ph@ecs.soton.ac.uk

Stephen Crouch
University of Sothampton
Highfield, Southampton,
United Kingdom
stc@ecs.soton.ac.uk

ABSTRACT

When component parts of distributed systems need to reach agreement, arriving at consensus is difficult if some components don't behave properly. The Byzantine Generals Problem described by Lamport and others exemplifies the difficulty.

In a real situation, components don't know which of their peers are faulty and hence they cannot apply the algorithms of Lamport et al, nor even decide if a suitable algorithm exists.

This paper discusses options available in this situation and describes how a good expectation of arriving at a consensus can be achieved without knowing for certain which or how many participants are behaving badly.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *Distributed Applications;* D.4.5. [**Operating Systems**]: Reliability – fault tolerance

General Terms

Algorithms, Design, Reliability.

Keywords

Distributed agreement, Byzantine Generals Problem.

1. INTRODUCTION

We have mature middleware for building such applications [5, 7-9] from interacting components [6]. Components generally assume their information is correct but the size of modern systems means this is no longer reasonable.

Some problems can solved with approaches like belief maintenance systems [2] or DataWarp [3] but sometimes inconsistencies have to be eliminated.

The Byzantine Generals Problem (BGP) described by Lamport et al [4] exemplifies the difficulty of arriving at a consensus in the presence of bad behaviour. It describes the conditions which have to be met if there is to be a solution and algorithms which provide achieve a solution for a specified number of rogues. However, in reality the participants do not know how many rogues are present and hence they can never *guarantee* they will arrive at a consensus.

Fortunately, the fact that it cannot be guaranteed doesn't imply that agreement will not be achieved.

SAC'07, March 11-15, 2007, Seoul, Korea.

2. BGP AND THE OM ALGORITHM

BGP concerns Generals surrounding a city who are trying to decide whether to attack. Each judges whether a concerted attack would succeed and they then have to distribute these opinions in such a way that a few rogue Generals can neither influence the decision (unduly) nor cause disagreement.

Lamport et al [4] provide a comprehensive analysis of the problem. They prove loyal Generals must outnumber the rogues by more than two to one for it to be possible to *guarantee* a correct decision They also provide algorithms (OMx) which guarantee the result in the face of no more than x rogues.

3. THE REAL SITUATION

The critical factor in selecting the algorithm is number of rogue participants but this number is not known. The participants could use the algorithm which is proof against the most rogues their number is able to defeat but this doesn't guarantee a correct result and the algorithms become greatly more complex with the number of rogues.

Alternatively they can estimate the number of rogues using knowledge of the likely failure rates of their peers. Assuming the probability of failure is reasonably small, the likelihood of two or more rogues amongst the group is very small suggesting that using an algorithm which is proof against two might a waste of effort.

4. USING OM1 IN PLACE OF OM2

Considering a group of seven, OM1 ensures a consensus in the presence of zero or one rogues and it is impossible to *guarantee* a correct result if there are three or more. OM2 guarantees the result if there are two rogues.

These algorithms use layers of message forwarding and voting. In OM1, there is one layer of message forwarding: each participant sends a message to each of the others and forwards each original message they receive. Each participant receives six messages in the style of, "I say x" and thirty messages which say, "He told me y", a total of 6 + 6x5 = 36 each, 252 messages in total. For OM2, the total is 1092, more than four times as many. Since the probability that this extra assurance will apply is small and what the group really wants

to know is how often they *actually* achieve the correct result, it appears this could be a waste of effort.

Let us consider seven Generals using OM1, label them A, B, … G and assume B and C are rogues. Suppose A sends b to B, c to C, etc. The good participants pass on values faithfully so A's messages to D,E,F,G are reported accurately and the messages b,c,..g will have the same value so each of the other good participants receives the correct value from A four times (once direct and three accurate reports from others), plus two reports via the rogues. Clearly, there is nothing the rogues can do to induce D,E,F,G to any value other than that truly sent by A – they would be outvoted by at least four to two. See Table 1.

	A	B	C	D	E	F	G
A							
B	b		C_1	D	e	f	g
C	c	b_1		D	e	f	g
D	d	b_2	C_2		e	f	g
E	e	b_3	C_3	D		f	g
F	f	b_4	C_4	D	e		g
G	g	b_5	C_5	D	e	f	

Table 1: Messages of OM1

The rogues can only create mayhem with their own values but as there are just two of them, they can only do so if the others are evenly divided. If the good participants are divided three to two, the rogues can achieve a pyrrhic victory by swinging the decision but the good participants still achieve a consensus. To really disrupt the process, the rogues need to induce disagreement in the others which is more difficult.

5. EXPERIMENTAL RESULTS

We have carried out experiments using implementations of the OM algorithms in which participant applications communicate using message passing [1, 5].

Probabilities for bad behaviour were selected to create experiments in which the long run mean number of rogues is 0, 1, 2 or 3. For the experiment reported, the rogues behaviour was to insert some chosen value into every message they send regardless of what they receive. Table 2 shows the results from 1000 runs and confirms that, for this style of bad of our rogues, there is no advantage to using OM2.

Probability of Traitor A	Mean No. Traitors	Correct Results	
		OM1	OM2
0	0	1000	1000
1/7	1.03	1000	993
2/7	2.01	992	954
3/7	2.96	996	914

Table 2: Results of 1000 runs

These results confirm that OM1 provides much better protection against the bad behaviour in a distributed agreement exercise than its guarantee to frustrate just one rogue would suggest and further suggests that there is no worthwhile return for the considerable additional effort of using OM2.

6. CONCLUSION

In their study, Lamport et al, consider the implications of the presence of badly behaved participants in a distributed agreement exercise. They show how many participants there need to be in total for it to be possible to eliminate the possible disruption of a given number of badly behaved participants and give algorithms which guarantee success.

However, in a real system participants don't know how many will behave badly. They can't even place an upper bound on the number. In this situation, they cannot select an algorithm which is guaranteed to succeed nor even decide if one exists. However, they can estimate the number of faulty members present in a group and using this it is possible to achieve a reasonable expectation of a correct result, even when it cannot be guaranteed.

For the particular circumstances examined, OM2 appears to offer better protection by guaranteeing a correct result if there are as many as two faulty processes where OM1 only protects against one. However this small assurance comes at a cost of more than four times the effort and in a practical experiment we found it was overwhelmed by other circumstances making OM1 a better choice.

7. REFERENCES

[1] A. Dickman, *Designing Applications With Msmq: Message Queuing for Developers*: Addison Wesley Publishing Company, 1998.

[2] N. Friedman and J. Y. Halpern, "Belief Revision: A Critique," *Journal of Logic, Language and Information*, vol. 8, pp. 401-420, July 1999.

[3] P. Henderson, R. J. Walters, S. Crouch, and Q. Ni, "DataWarp: Building Applications which make Progress in and Inconsistent World," in *4th IFIP WG 6.1 International Conference, Distributed Applications and Interoperable Systems (DAIS 2003)*, Paris, 2003, pp. 167-178.

[4] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382-401, 3rd July 1982.

[5] Microsoft, "Microsoft Message Queuing Services," Microsoft, 2001.

[6] L. Nicolle, "John Taylor - The Bulletin Interview," The Computer Bulletin: British Computer Society, 1999.

[7] R. Sessions, *COM and DCOM - Microsoft's Vision for Distributed Computing*: Wiley Computer Publishing, 1998.

[8] C. Szyperski, *Component Software*: Longman, 1998.

[9] A. Thomas, "Enterprise JavaBeans Technology," Patricia Seybold Group, White Paper prepared for Sun Microsystems Inc December 1998.