

Doing Right Reboot by Yourself

Dan Hong, Maoke Chen
Network Research Center, Tsinghua University
{hongdan, mk}@cernet.edu.cn

Abstract

Distributed network testbed is extremely important to developing and evaluating novel network services and innovative Internet architectures. Keeping such a testbed in high availability is a big challenge, and fast healing from trouble is extremely important. We propose and implement a self-healing mechanism for nodes in a testbed, called SCOL(Self-COntrol), and implements it on 6PlanetLab, an IPv6 network experiment platform in China. The real data that we collected over 2 months suggests that the proposed self-healing model can overcome drawbacks of most ever-existing mechanisms. Similar conclusion is also applicable for PlanetLab and other distributed testbeds.

1 Introduction

PlanetLab¹ has created a successful paradigm for global network research testbed that supports the development of new network services. Since its deployment in 2003, it has over 800 nodes all over the world now. However the operation team of PlanetLab has never stopped fighting with the instability of the testbed, which is caused by over-consumption of node resources, link failures, or impairment of physical nodes themselves. As most of the nodes are distant from the PLC with complicated network conditions between them, and many administrators are not aggressive enough to maintain local availability, it usually costs a lot to recover a broken node. According to the operating data collected by CoMon²[6] at 15:00 on May. 1st, 2008, there were at least 344 nodes in 849 ones could not be logged on, the available rate was only 59.5%. The 50 PlanetLab nodes on CERNET in China are even worse, because fewer than 18 ones are available for use on average since 2006.

6PlanetLab³ is another sample of network experiment platform, which was developed in 2006 and contains 52 nodes in China on the date of writing. It follows the slice

computing model from PlanetLab, but is realized with user-oriented addresses (UOA[2]) in native operating systems rather than virtual machines over customized OS kernels. With native host OS, 6PlanetLab gains the scalability in functionality. Any new features can be deployed as long as the native operating systems have supported. Moreover, addresses being assigned to users rather than machines give 6PlanetLab users a freedom to use the whole 64K port range without conflict, and have full privilege over the configuration for their own addresses, routes and tunnels. We believe this openness and flexibility is greatly beneficial to the research community, but it also increases the difficulty in distributed platform maintenance.

This paper is organized as follows. Section 2 discusses ever-existing mechanisms. Section 3 proposes our self-healing mechanism SCOL. Section 4 evaluates the SCOL. Section 5 makes a conclusion and raises the future work.

2 Related Works and Limitations

To address the problem of system healing, PlanetLab developed Ping-Of-Death (POD)[1], a kernel patch that can reboot the node immediately when receiving a properly signed ICMP packet. When administrator finds the node unable to be logged on, but still reachable through network, he can send a POD packet to reboot the node in question. This mechanism is efficient in the case where crash caused by user-space troubles, but it doesn't work if a node's network services or configurations are damaged. More importantly, the failed node will never reboot until some administrator notices and sends POD messages to it. Depending on human intervention costs a lot of time to recover a broken node.

Having the motivation of enabling nodes of 6PlanetLab in trouble be recovered automatically, Wang et al. designed the KLAB6[4] protocol based on the assumption that the unreachable node can go back into work after rebooting. KLAB6 uses a Keep-Alive mechanism to trigger reboot when the node is out of reach for a definite period. A kernel module, which accepts the Keep-Alive message from so-called super nodes, is installed into any 6PlanetLab node. The Keep-Alive messages are sent by the super node to

¹PlanetLab Homepage. <http://www.planet-lab.org/>.

²CoMon.<http://comon.cs.princeton.edu/>

³6PlanetLab Homepage. <http://core.cgeni.edu.cn/>

3.2 Local Keep-Alive

The self-diagnose flow will run periodically through crontab, which can overcome the typical troubles mentioned above within 10 minutes. However, if the crontab daemon does not work for some reason, or even the user space is crashed at all, the self-diagnosis cannot take effect any more. To address this problem, we utilize the KLAB6 protocol again, but in a local fashion. We ask the node send Keep-Alive messages to itself, destining at its local address. In other words, the Keep-Alive messages are sent from user space to kernel on a same node(Shown as Fig. reffig:keepalive). So if the user space is crashed, the KLAB6 kernel module will not receive Keep-Alive messages any more, and will reboot the node automatically. As the loss rate of UDP between user space and kernel is negligible, such a mechanism never brings in false reboots. The bidirectional communication is not needed here either, so the KLAB6 module can be greatly simplified.

The Keep-Alive sending process will be periodically executed by crontab in user space. If the crontab is broken, the node will also reboot.

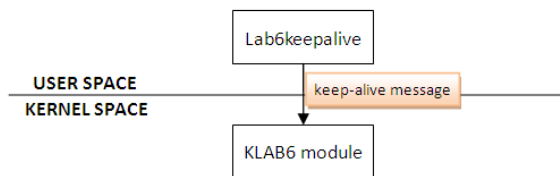


Figure 3. Local keepalive

4 Evaluation

4.1 Quantitative Testing

This section checks if the node can recover from failures of its own within 10 minutes, and if the node can get rid of unnecessary reboots.

We installed our SCOL module on a server in our lab, which has the same configuration with other 6PlanetLab nodes. We prepared 10 testing cases, which are classified into 3 classes. TC1 focuses on the capability of identify outer failure and inner failure. TC2 focuses on the failures reasoned from mis-configuration of node network. TC3 focuses on the failures caused by crash in user-space. The testing cases and results are shown as table 1.

From the result we can see that SCOL can differentiate the external failures(TC1) and internal failures(TC2 and TC3). To the cases in TC2, node can recover without reboot within 10 minutes in most cases(TC2.1~TC2.4). Even to the problem that can not be fixed(TC2.5), the node will not reboot frequently. To the failures in TC3, the node

Table 1. Module Testing

TC	Input(failure,duration)	Output(result)
1.1	unplug the network cable (10min)	never reboot; user cannot login
1.2	shut down switch (10min)	never reboot; user cannot login
1.3	drop the route to the node (10min)	never reboot; user cannot login
2.1	drop the ip address (10min)	user can login without reboot (6min)
2.2	drop the default gateway (10min)	user can login without reboot (6min)
2.3	ifconfig eth0 down (10min)	user can login without reboot (7min)
2.4	stop sshd service (10min)	user can login without reboot (7min)
2.5	uninstall sshd (30min)	the node reboot once; user cannot login(8min)
3.1	stop crontab (10min)	user can login after reboot (2min)
3.2	killall5 (10min)	user can login after reboot (2min)

can recover by rebooting within 2 minutes. So the results matches the design requirements of SCOL.

4.2 Operational data

The SCOL was deployed on 6PlanetLab since Mar. 15th, 2008. The operating data of 6PlanetLab from 2008-3-15 to 2008-5-23 is summarized as Fig 4.

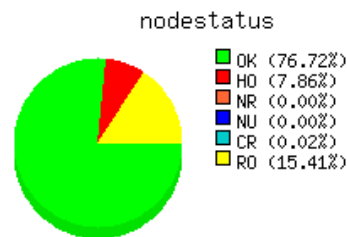


Figure 4. The available rate of 6PlanetLab from 2008-3-15 to 2008-5-23

Our network management system(nms) probes each 6PlanetLab node every 10 minutes, and use different tokens to present the states of node.

- OK:the node can be logged in by ssh
- HO:the node is unreachable by ICMP
- RO:the gateway of node is unreachable by ICMP
- NR:no route to host

- NU:node unreachable
- CR:connection refused by sshd

So each node was working well in 76.72% of this period of time on average. Fig 5 shows the node states on each day, and the statistical result shows that in about 97.2% of time there are over 40 nodes in 52 working well. As some nms failures cause concurrent false RO sometimes, the real average available rate of 6PlanetLab node is over 76.72%.

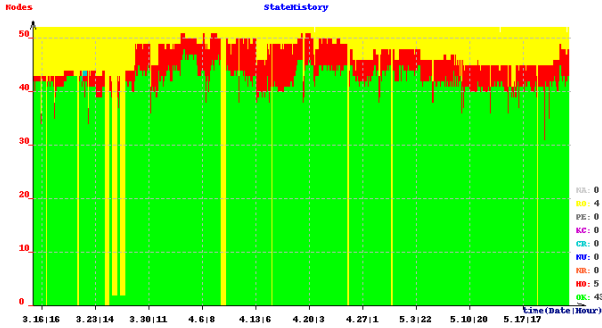


Figure 5. The operating data of 6PlanetLab from 2008-3-15 to 2008-5-23

All the RO problems and part of the HO problems are caused by link or router failures between 6PlanetLab nodes. In these cases, the node should do nothing except waiting for maintenance of administrators. Part of failures like HO, CR and NU are caused by mis-configuration of node network and problems of sshd service, so the reboot caused by these inner reasons are shown as Table 2.

Compared with Fig. 1, the frequency of reboot count is significantly reduced because false reboots which are frequent in KLAB6 protocol are avoided in SCOL, and most of inner failures are fixed without reboot.

5 Conclusion

SCOL is a full automatic solution, which can recover node with least cost timely, so it saves quite a lot of cost of node maintenance for testbed. It is able to differentiate inner and outer failures, so it can get rid of unnecessary reboots. Being non-centralizing and lightweight, it is scalable, and has no overheads of end-to-end communication.

To PlanetLab, the network configuration of node is fixed, so the self-diagnosis is not needed in most cases. But the local keepalive is still useful for user-space crashes, by which the broken node can recover by itself shortly, rather than waiting for POD messages.

However, SCOL can not do everything. When the kernel is crashed, the KLAB6 module can not work either, so the node can not be rebooted. Reboot can not address serious problems like KERNEL PANIC as well. Also if

Table 2. Reboot count caused by SCOL from 2008/3/15 to 2008/4/30

date	cpu1	cpu2	csu1	csu2	dlut1	dlut2	fudan1	fudan2	hit	neu1	pku1	pku2	sdul	seu1	thu1	thu2	tju1	tongji2	uestc1	xjtu1	thusec2	sum	
3.23																					1	1	
3.24																		1					1
3.28						1																	1
3.3			1	1	5	1						1											9
4.01																						1	1
4.02			1					1															2
4.03														2									2
4.06											1	1											2
4.07			1										1							1			3
4.09	1						1								1								3
4.1		1	1															1					3
4.12									2														2
4.2											1	1											2
4.24															2	2	1						5
4.27							1				1	1											3
5.12																					1		1
5.21														1			1						2
5.23									2														2
sum	1	2	3	1	5	2	2	1	2	2	3	4	2	2	3	2	3	1	1	1	2	45	

the sshd is uninstalled viciously or some critical libraries are destroyed, the node can not be fixed even by rebooting. So something like automatic reinstallation may be needed, which is our future work.

References

- [1] R. Adams. Distributed system management: Planetlab incidents and management tools. Technical report, PDN-04-18, November 2003.
- [2] M. K. Chen, T. Y. Li, X. Li, N. M. Guire, and Q. G. Zhou. Prototyping user-oriented addressing model in linux kernel. In *9th Real-time Linux Workshop (RTLWS 2007)*, Linz, Austria, 2007.
- [3] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking(TON)*, 2(2):122–136, April 1994.
- [4] D. Hong, Y. Wang, J. X. Lu, M. K. Chen, and X. Li. Building a robust and economical internet testbed: 6planetlab. In *15th IEEE International Conference on Networks, ICON 2007*, pages 289–294, Adelaide, Australia, November 2007.
- [5] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, November 1991.
- [6] L. Wang, K. S. Park, R. Pang, V. P., and L. Peterson. Reliability and security in the codeen content distribution network. In *annual conference on USENIX Annual Technical Conference*, pages 14–14, Boston, MA, 2004.