Journal of Network and Computer Applications 32 (2009) 1–30

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Bottleneck Active Node Detouring for capsule-based active network

Tzu-Chi Huang [a,*], Ce-Kuen Shieh [a], Yu-Ben Miao [b]

[a] Department of Electrical Engineering, National Cheng Kung University, Lab No. 92579, No. 1, Ta-Hsueh Road, Tainan, Taiwan
[b] Information and Communications Research Laboratories, Industrial Technology Research Institute, Taiwan

ARTICLE INFO

Article history:
Received 30 October 2007
Received in revised form
27 March 2008
Accepted 23 April 2008

Keywords:
Bottleneck Active Node Detouring
BAND
Active Node
Active network
Capsule
Programmable network
Bottleneck
Throughput
Detouring

ABSTRACT

Active Node is a network device capable of forwarding packets and giving them the computation service in the meantime. It plays a critical role in capsule-based active networks to speed up the development of a protocol and facilitate the deployment of a service inside networks. When getting overloaded, however, it becomes a throughput bottleneck to all Active Applications whose packets traverse the Active Node. It can enable the Bottleneck Active Node Detouring (BAND) proposed in this paper to free Active Applications from the penalty of poor throughput because not all Active Applications need the computation service in the bottleneck Active Node. Besides, it can enable the BAND to give Active Applications other benefits identified in this paper.

1. Motivation

Today's Internet succeeds tremendously because of the simplicity. It works in best-effort, unreliable, and automatic manners. In the best-effort manner, it routes packets via consulting the routing table and forwards them to the destination as soon as possible. In the unreliable manner, it has no guarantee of packet delivery, provides no error recovery when delivering packets, and discards packets in network congestion. In the automatic manner, it builds and corrects the routing table automatically through an exchange of the routing information without many administration costs.

* Corresponding author. Tel.: +886 6 275 7575x62400x1779; fax: +886 6 234 5486.
E-mail addresses: huang@hpds.ee.ncku.edu.tw, tzuchi@ee.ncku.edu.tw (T.-C. Huang), shieh@ee.ncku.edu.tw (C.-K. Shieh), miao@itri.org.tw (Y.-B. Miao).

1084-8045/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2008.04.005

Thanks to the simplicity, it makes a solid global infrastructure whereby people can transmit packets at will without acquiring permission in prior. Given the simplicity like "the waist of the hourglass", it requires people to utilize extra protocols or services such as encryption (Kent and Seo, 2005), flow control (Postel, 1981), reliable data delivery (Postel, 1981), proxy (Cooper and Dilley, 2001), packet filter and translation (Freed, 2000; Srisuresh and Egevang, 2001), etc., in order to meet various application needs in communication. However, it often takes a long time to accommodate people with a new protocol or service due to the slow protocol standardization progress, the costly upgrade of network devices, and the troublesome deployment and management of world-wide services.

Capsule-based active network (Tennenhouse et al., 1997; Calvert, 2006) is a promising way to speed up the development of a protocol and facilitate the deployment of a service inside networks. It needs network devices not only to forward packets like conventional network devices do, but also to support the computation service by means of evaluating user codes; the network device capable of evaluating user codes is referred as the Active Node (AN). In ANs, it evaluates user codes that are carried in packets or downloaded from a network server according to references (Wetherall, 1999) in packets. In security considerations, it evaluates user codes safely in an Execution Environment (EE), e.g., the Java Virtual Machine (JVM), like a "sandbox" rather than evaluates them directly in the native environment offered by the operating system (OS) in the AN. With the use of EEs, it not only greatly improves the user code portability among ANs, but also efficiently controls the user code's access to resources in ANs when evaluating user codes to use CPU cycles and probably other resources such as memory space, disk volume, network bandwidth, etc. Although leveraging weaknesses of time-consuming protocol development and service deployment in the Internet, however, it gives Active Applications (i.e., the applications providing or utilizing user codes) poor throughput due to overheads of supporting the computation service in a bottleneck AN.

Compared with a conventional network device, the AN in capsule-based active networks has more chances to become a throughput bottleneck to an Active Application. First, it may naturally have limited computation power incapable of supporting the computation service to meet Active Application requirements for a certain degree of throughput. Second, it may be poorly implemented to incur a high overhead while processing packets that request the computation service, e.g., having the context switch overhead by copying packets between the user-mode EE and the kernel-mode OS in the AN. Third, it may have the packet queue overloaded by numerous packets requesting the computation service, e.g., due to greedy Active Applications or denial-of-service attacks, so certain Active Applications cannot get required throughput. Fourth, it may serve malicious Active Applications' user codes that are programmed to exhaust available resources in the AN to affect throughput of other Active Applications. When throughput of an Active Application is degraded, the bottleneck AN may have negative impacts on people, the suffering Active Application, or other Active Applications because not all above reasons affect merely one Active Application. It may increase network latency of Active Applications, abort communication of Active Applications, prohibit Active Applications from functioning, and hurt people experiences in network surfing through Active Applications.

Currently available solutions against poor throughput in a bottleneck AN in the capsule-based active network can be divided into the hardware-based solution, the sophisticated software-based solution, the resource-based solution, and the security-based solution. The hardware-based solution uses powerful or special hardware (Decasper et al., 1999; Wolf and Turner, 2000) to implement a high-performance AN. The sophisticated software-based solution implements the kernel-mode EEs (Ruf et al., 2003), adopts special kernel-mode data path (Decasper et al., 2000), evaluates user codes directly in the native environment of the OS (Nygren et al., 1999), or provides user codes with a lightweight programming language (Moore et al., 2001). The resource-based solution uses different packet queuing (Williams et al., 1999; Ramachandran et al., 2000) or CPU scheduling algorithms (Sabrina and Jha, 2003; Nguyen et al., 2003; Qie et al., 2000) to make the fairness of serving Active Applications' packets, so throughput can be fairly distributed among Active Applications. The security-based solution uses various protection mechanisms enforced by the programming language of the user code or by the Application Programming Interfaces (APIs) of the EE (Wetherall, 1999; Braden et al., 2002; Tullmann et al., 2001; Kulkarni et al., 1998) in order to monitor and control user codes' access to AN resources in a fine-granularity manner. Accordingly, the security-based solution