Media Link failure detection for Modern routers

Alexandre Cassen
Linux Virtual Server OpenSource Project
Paris, France, December 2003
acassen@linux-vs.org, http://www.LinuxVirtualServer.org/~acassen/

Introduction

This paper presents a strategy to extend the current Linux kernel code for better detection of media link failure. All modern routing software needs to be able to diagnose media link activity in order to drive its own internal finite state machine (FSM). This FSM drives state transitions and must has low level event abstraction to guarantee stability. This small paper will be focused on Keepalived/VRRP implementation and needs.

Modern Networking software needs

To speed developement, much routing software is implemented in user space e.g. Keepalived/VRRP. As well, routing software such as VRRP also must access low level information, especially network interfaces. On loading, VRRP queries the kernel for this information producing a user space abstraction of the interfaces, which can be later queried by userspace processes. The NETLINK messaging subsystem provides kernel to userspace communication through subscription to a netlink broadcast socket. By a process known as, kernel netlink reflection, the userspace information is updated with changes to internal data structures.

There are several subscription groups: the RTMGRP_LINK group handles media link related kernel events. Through a userspace netlink socket registering RTMGRP_LINK group. On LINK event (eg: IFF_UP|DOWN), the kernel generates a netlink broadcast catchable userspace through a netlink socket registering RTMGRP_LINK group. Thus, when some other piece of software brings an interface down, then Keepalived/VRPP can catch this event.

The problem for Keepalived/VRRP and most routing software is that media link state changes don't generate a kernel netlink broadcast preventing user space routing software from acting on these events. Thus, IFF_UP|DOWN generate netlink broadcast, but not a media link state change notification.

Current media link failure detection

Facing the problem described in the last section, there is a need to catch NIC state changes in userspace. There are severals ways to determine NIC link status. All modern NICs have a dedicated chipset in charge of link media. This chipset provides a NIC driver register abstraction that can be polled both from kernel or userspace applications. The most widely used registers are the MII registers which provide information on the state of the NIC hardware. In our case the most interresting MII register is the BMSR (Basic Mode Status Reg) which contains the media link status. All NICs drivers implement a timer on this MII BMSR to update the kernel with the content of this register. This contents of this register can then be accessed in userspace through a ioctl() system call.

The current strategy for media link failure detection is to register a userspace a timer thread periodically polling this BMSR through ioctl() system call. This is a basic polling design which does not scale well. For 10 NICs it works acceptably well, but for 20 NICs, the CPU will be tied up with ioctl() calls.

LINKWATCH Design

In March 2002, Stefan Rompf, <sux@loplof.de> published a patch to the LKML that were approved for 2.5 inclusion in December 2002. The basic idea of the patch is to provide kernel netlink broadcast events on link media state changes. LinkWatch provides hooks in netif_carrier_on() and netif_carrier_off() to queue NIC link events. Then a timer thread will run the queue to generate netlink brodcast events. This timer thread will reflect userspace the IFF_RUNNING flag according to media link status.

With this new design userspace routing software can move from a polling design to an event design when testing for the IFF_RUNNING interface flag.

LINKWATCH & Keepalived/VRRP

For intensive VRRP uses or simply for cleaner way of use, I really recommand to use this new design. You will benefit a better performance reducing the CPU usage. This patch is now part of 2.5 branch, since I really consider it essential, I will maintain a 2.4 release that you will be able to download on the Keepalived website.

LINKWATCH NIC drivers Compatibility

Most common drivers are compatible with this design since they call netif_carrier_on() and netif_carrier_off() inside their BMSR timer thread. So drivers like 3c59x, eepro100, e100, e1000, tg3 are compatible with linkwatch design. Other drivers must be patched to call netif_carrier facilities functions to provide netlink broadcast. Since the Tulip driver is widely used we only provide a quick patch for it.