Netconsole

started by Ingo Molnar <mingo@redhat.com>, 2001.09.17

2.6 port and netpoll api by Matt Mackall <mpm@selenic.com>, Sep 9 2003

IPv6 support by Cong Wang <xiyou.wangcong@gmail.com>, Jan 1 2013

Extended console support by Tejun Heo <tj@kernel.org>, May 1 2015

Please send bug reports to Matt Mackall <mpm@selenic.com> Satyam Sharma <satyam.sharma@gmail.com>, and Cong Wang <xiyou.wangcong@gmail.com>

Introduction:

This module logs kernel printk messages over UDP allowing debugging of problem where disk logging fails and serial consoles are impractical.

It can be used either built-in or as a module. As a built-in, netconsole initializes immediately after NIC cards and will bring up the specified interface as soon as possible. While this doesn't allow capture of early kernel panics, it does capture most of the boot process.

Sender and receiver configuration:

It takes a string configuration parameter "netconsole" in the following format:

```
netconsole=[+][src-port]@[src-ip]/[<dev>],[tgt-port]@<tgt-ip>/[tgt-macaddr]

  where
       +             if present, enable extended console support
       src-port      source for UDP packets (defaults to 6665)
       src-ip        source IP to use (interface address)
       dev           network interface (eth0)
       tgt-port      port for logging agent (6666)
       tgt-ip        IP address for logging agent
       tgt-macaddr   ethernet MAC address for logging agent (broadcast)
```

Examples:

```
linux netconsole=4444@10.0.0.1/eth1,9353@10.0.0.2/12:34:56:78:9a:bc
```

or:

```
insmod netconsole netconsole=@/,@10.0.0.2/
```

or using IPv6:

```
insmod netconsole netconsole=@/,@fd00:1:2:3::1/
```

It also supports logging to multiple remote agents by specifying parameters for the multiple agents separated by semicolons and the complete string enclosed in "quotes", thusly:

```
modprobe netconsole netconsole="@/,@10.0.0.2/;@/eth1,6892@10.0.0.3/"
```

Built-in netconsole starts immediately after the TCP stack is initialized and attempts to bring up the supplied dev at the supplied address.

The remote host has several options to receive the kernel messages, for example:

1. syslogd
2. netcat

   On distributions using a BSD-based netcat version (e.g. Fedora, openSUSE and Ubuntu) the listening port must be specified without the -p switch:

   ```
           nc -u -l -p <port>' / 'nc -u -l <port>

       or::

           netcat -u -l -p <port>' / 'netcat -u -l <port>
   ```

3. socat

```
socat udp-recv:<port> -
```

Dynamic reconfiguration:

Dynamic reconfigurability is a useful addition to netconsole that enables remote logging targets to be dynamically added, removed, or have their parameters reconfigured at runtime from a configfs-based userspace interface. [Note that the parameters of netconsole targets that were specified/created from the boot/module option are not exposed via this interface, and hence cannot be modified dynamically.]

To include this feature, select CONFIG_NETCONSOLE_DYNAMIC when building the netconsole module (or kernel, if netconsole is built-in).

Some examples follow (where configfs is mounted at the /sys/kernel/config mountpoint).

To add a remote logging target (target names can be arbitrary):

```
cd /sys/kernel/config/netconsole/
mkdir target1
```

Note that newly created targets have default parameter values (as mentioned above) and are disabled by default -- they must first be enabled by writing "1" to the "enabled" attribute (usually after setting parameters accordingly) as described below.

To remove a target:

```
rmdir /sys/kernel/config/netconsole/othertarget/
```

The interface exposes these parameters of a netconsole target to userspace:

enabled	Is this target currently enabled?	(read-write)
extended	Extended mode enabled	(read-write)
dev_name	Local network interface name	(read-write)
local_port	Source UDP port to use	(read-write)
remote_port	Remote agent's UDP port	(read-write)
local_ip	Source IP address to use	(read-write)
remote_ip	Remote agent's IP address	(read-write)
local_mac	Local interface's MAC address	(read-only)
remote_mac	Remote agent's MAC address	(read-write)

The "enabled" attribute is also used to control whether the parameters of a target can be updated or not -- you can modify the parameters of only disabled targets (i.e. if "enabled" is 0).

To update a target's parameters:

```
cat enabled                           # check if enabled is 1
echo 0 > enabled                      # disable the target (if required)
echo eth2 > dev_name                  # set local interface
echo 10.0.0.4 > remote_ip             # update some parameter
echo cb:a9:87:65:43:21 > remote_mac    # update more parameters
echo 1 > enabled                      # enable target again
```

You can also update the local interface dynamically. This is especially useful if you want to use interfaces that have newly come up (and may not have existed when netconsole was loaded / initialized).

Extended console:

If '+' is prefixed to the configuration line or "extended" config file is set to 1, extended console support is enabled. An example boot param follows:

```
linux netconsole=+4444@10.0.0.1/eth1,9353@10.0.0.2/12:34:56:78:9a:bc
```

Log messages are transmitted with extended metadata header in the following format which is the same as /dev/kmsg:

```
<level>,<sequnum>,<timestamp>,<contflag>;<message text>
```

Non printable characters in <message text> are escaped using "xff" notation. If the message contains optional dictionary, verbatim newline is used as the delimeter.

If a message doesn't fit in certain number of bytes (currently 1000), the message is split into multiple fragments by netconsole. These fragments are transmitted with "ncfrag" header field added:

```
ncfrag=<byte-offset>/<total-bytes>
```

For example, assuming a lot smaller chunk size, a message "the first chunk, the 2nd chunk." may be split as follows:

```
6,416,1758426,-,ncfrag=0/31;the first chunk,
6,416,1758426,-,ncfrag=16/31; the 2nd chunk.
```

Miscellaneous notes:

Warning

the default target ethernet setting uses the broadcast ethernet address to send packets, which can cause increased load on other systems on the same ethernet segment.

Tip

some LAN switches may be configured to suppress ethernet broadcasts so it is advised to explicitly specify the remote agents' MAC addresses from the config parameters passed to netconsole.

Tip

to find out the MAC address of, say, 10.0.0.2, you may try using:

```
ping -c 1 10.0.0.2 ; /sbin/arp -n | grep 10.0.0.2
```

Tip

in case the remote logging agent is on a separate LAN subnet than the sender, it is suggested to try specifying the MAC address of the default gateway (you may use /sbin/route -n to find it out) as the remote MAC address instead.

Note

the network device (eth1 in the above case) can run any kind of other network traffic, netconsole is not intrusive. Netconsole might cause slight delays in other traffic if the volume of kernel messages is high, but should have no other impact.

Note

if you find that the remote logging agent is not receiving or printing all messages from the sender, it is likely that you have set the "console_loglevel" parameter (on the sender) to only send high priority messages to the console. You can change this at runtime using:

```
dmesg -n 8
```

or by specifying "debug" on the kernel command line at boot, to send all kernel messages to the console. A specific value for this parameter can also be set using the "loglevel" kernel boot option. See the dmesg(8) man page and Documentation/admin-guide/kernel-parameters.rst for details.

Netconsole was designed to be as instantaneous as possible, to enable the logging of even the most critical kernel bugs. It works from IRQ contexts as well, and does not enable interrupts while sending packets. Due to these unique needs, configuration cannot be more automatic, and some fundamental limitations will remain: only IP networks, UDP packets and ethernet devices are supported.