

1 Staying Alive...or Online, Anyway

2 Dick Munroe

3 **I**run a small consulting company from a home office. I have a
4 decent number of machines online constantly running every-
5 thing from OpenVMS to Macintosh OS 9 and X to Windows to
6 Linux. These machines support my client's software development
7 efforts as well as my own products, marketing, etc. Additionally,
8 my house is fairly well wired, both with wireless and wired net-
9 works and a number of workstations used by the rest of my family.
10 Unfortunately, none of my family members have been particularly
11 interested in understanding the communication infrastructure in the
12 house and thereby hangs this tale.

13 The Usual Suspects

14
15 The basic networking infrastructure is pretty much the same
16 everywhere and nothing is too different at my installation. I have a
17 router/firewall/server built with an old 500-MHz AlphaServer 800
18 running Debian Linux, the testing version. It's my personal favorite
19 Linux distribution; try it if you like a lot of control over what goes
20 onto your system and you don't want to hassle with installations
21 too much.

22 My ISP is Verizon and at the time I wrote this article, I was
23 using their 768-Kbs ADSL service. The modem they sold me with
24 the service was a Westell WireSpeed and, for reasons that neither
25 Verizon nor I ever discovered, the modem would periodically lock
26 up and have to be power-cycled to restart. There seemed to be some
27 correlation between wet weather and the modem locking up but
28 nothing that was particularly reliable in predicting the problem. I
29 travel quite a bit and, of course, the modem would behave perfectly
30 for months while I was home. When I would leave, of course, it
31 would lock up and cut off my clients from their support services.

32 Clearly, I needed something to automate the things that I would
33 do to get my communications infrastructure online in the face of a
34 variety of flaky environmental factors.

35 Requirements

36
37 Whatever the solution to staying connected was, it had to satisfy
38 the following requirements:

- 39
40 1. Robust in the face of power outages. The router/firewall and any
41 servers had to come back up automatically, reboot themselves,
42 and restart communications. The connection to Verizon's DSL
43 network had to come back automatically.
- 44 2. Insensitive to individual piece failures. While a completely failed
45 component (you try getting a DSL modem or a computer that's
46 taken a lightning strike to work again) would have to be
47 replaced, the firewall/router should be able to detect, repair, and
48 complain if repair isn't possible with all pieces of the "plumb-
49 ing" between itself and the intra- and inter- network.
- 50 3. No external intervention. In the event that anything had to be
51 done to recover connectivity, I shouldn't need to be there nor
52 should there be a need for a network connection to fix the prob-
53 lem (if at all possible).
- 54 4. Flexible. I had to be able to manipulate each piece of the solution
55 individually. If I had to power cycle something, I didn't want to
56 power cycle everything.
- 57 5. Inexpensive. As a small businessman, I'm sensitive to price. I
58 tend to trade "sweat" for money in my office, buying used gear,
59 or creating solutions to problems out of free software compo-
60 nents and a little in-house glue.

61
62

1 All of my computers reboot themselves automatically, and the only
2 time I saw a situation in which the DSL network didn't reconnect
3 after power-cycling was when a traffic accident put a car into the
4 local Verizon office, so I felt reasonably certain that the hardware I
5 had satisfied the first requirement.

6 With the Debian Linux distribution's PPPoE client (by Roaring
7 Penguin), I had noticed the PPP interface disappeared every time
8 the WireSpeed modem needed to be power-cycled. Because I could
9 reliably spot the problem, it was possible to try to fix it if I could
10 actually do the power cycling. A little research in "home automa-
11 tion" turns up any number of possibilities for cycling power on an
12 appliance. However, qualifying the research to those products sup-
13 ported by Linux drives you, more or less immediately, to the X10
14 product line.

15 The X10 product line is a "hobbyist" collection of home
16 automation products ranging from motion sensors to Web-enabled
17 cameras, to light dimmers, appliance controllers, and (most impor-
18 tantly) transceivers for the X10 protocol.

19 Basically these products work by using the 60-cycle line current
20 in your wiring as a carrier and sending a very simple protocol from
21 a control station to one or more slave devices. Their C17A product
22 (a.k.a., "Firecracker") connects to a standard RS-232 serial port and
23 emits the X10 protocol on radio frequencies (essentially acting as
24 one of the X10 "remote control" units). The transceiver takes the
25 radio frequency version of the X10 protocol and converts it to the
26 line current carrier wave version of the protocol. The AlphaServer
27 800 I use for a router has an RS-232 port suitable for use with
28 "Firecracker" and doesn't have anything else, so if it were to be
29 possible to cycle power on my modem, it would have to be the
30 "Firecracker" that would do the job.

31 I have to emphasize the "hobbyist" aspect of the X10 product
32 line. The X10 hardware design is very inexpensive and not suitable
33 for industrial or other harsh environments. X10's customer support
34 isn't all that consistent, and they have a mixed reputation for
35 refunds for defective equipment. However, there is a decent user
36 group if you are more comfortable asking users about solutions to
37 problems rather than the company's technical support. The bottom
38 line is that X10 home automation solutions can be made to work in
39 specific limited environments if you are willing to spend the time.
40 Test your environment and your specific X10 hardware thoroughly
41 before committing your business. In my lab, for example, although
42 the modem solution described here worked like a dream, a substan-
43 tially more complicated lighting control scheme would not (or at
44 least not for the money I was interested in spending).

45 At the time, there was a "special" going on (X10 is always run-
46 ning specials) for a "Firecracker", a transceiver for the
47 "Firecracker" signals, and several lighting control units. The total
48 cost for the X10 hardware was less than \$40 so the price was cer-
49 tainly right.

50 A little further research turned up a shell-scripting interface for
51 the "Firecracker" called Bottle Rocket, which had been ported to
52 Debian and was available as a Debian package. As of this writing,
53 the author of Bottle Rocket is looking for a maintainer for the pack-
54 age. However, it is functional in its current form. So there appeared
55 to be a collection of stuff that would satisfy my requirements if I
56 were clever enough to put all the pieces together.

57 The Hardware Solution

58
59 The first step was to buy the necessary X10 hardware and test
60 whether the solution could be made to work. At this point, the
61 worst case was that I would be out \$40 for the hardware.

62 The kit that I bought consisted of:

- 1
- 2 • Firecracker Control Module
- 3 • Remote Control Unit
- 4 • Transceiver Module
- 5 • Lamp Module

6

7 This was just enough equipment to put together the solution I had
8 in mind and test it with and without software. Once the hardware
9 arrived, it took very little time to plug the transceiver and lamp
10 module into a power strip, plug the WireSpeed modem into the
11 transceiver module, and see whether the X10 hardware would control
12 the modem state.

13 After that, I fired up aptitude, the Debian package manager of
14 choice at my shop, and installed the source package of Bottle
15 Rocket (version 0.05b3), built it, tested it, and found it didn't work.
16 So, I rebuilt the package using the debug mode, which, among
17 other things, turns off all compiler optimizations, tried again, and
18 this time it worked. Your mileage may vary, but if Bottle Rocket
19 doesn't work for you out of the box, try rebuilding in debug mode.

20 So, now I had a hardware-only solution. The WireSpeed could
21 be controlled individually without any other hardware, the solution
22 was inexpensive, and if necessary, flexible. Now, I just needed to
23 make it smart.

24 The Software Solution

25

26 Here comes the fun stuff — writing the software. Because
27 Bottle Rocket came with an interface that could be used from a
28 shell script, I decided to write a daemon that would:

- 29
- 30 • Start/Stop with the usual init.d interface
- 31 • Be configurable to accommodate the different delays and restrictions
32 imposed by ISPs and modem builders
- 33 • Attempt a reasonable power-cycle strategy (not simply madly
34 flip the power switch)
- 35 • Warn me if the network stayed down after trying “everything”

36

37 The configurable parameters of the check-pppoe daemon are
38 shown in Listing 1. The parameters are:

39

40 CYCLETIME — The existence of the PPP interface of interest will be
41 checked every CYCLETIME seconds. The number is specified
42 in “sleep” format.

43 DSL_PROVIDER — The Debian PPP infrastructure provides for a
44 variety of named DSL providers. The default is “dsl-provider”.

45 HOUSECODE — The X10 address of the module that controls power
46 to the modem. Since your modem can generally be plugged into
47 the transceiver, only a single X10 module is generally necessary.

48 HOUSECODE_OFF_PAUSE — To allow the modem hardware time to
49 shut down properly, check-pppoe waits for the specified time
50 before proceeding after powering off the modem.

51 HOUSECODE_ON_PAUSE — To allow the modem hardware time to
52 come up completely and detect both the wide area and local networks,
53 this amount of time is allowed to elapse before bringing
54 up the PPP interface. You'll have to figure out how long your
55 modem takes by inspection. Time it, then add a few seconds to
56 make sure that the modem is really up.

57 POWERCYCLETIME — The integer number of seconds to wait once
58 the need to cycle power to the modem has been determined. The
59 failure modes I've seen with the DSL network are either that the
60 network connection comes back after the first power cycle (the
61 problem was with the modem) or the network is down and will
62 be back up “soon”. To keep from blindly flipping the power on

1 the modem, check-pppoe implements a back-off strategy such
2 that it waits for subsequently more time (to a configurable maxi-
3 mum) between attempts to cycle power.
4 POWERCYCLETIMEINCREMENT — The integer number of seconds to
5 increment POWERCYCLETIME if the PPP connection fails to
6 come up after a power cycle.
7 POWERCYCLETIMEMAXIMUM — The integer number of seconds that is
8 the maximum amount of time between power-cycle attempts.
9 PPP — The PPP interface to be checked. It is a simple existence
10 check. If the PPP interface exists, the network connection is
11 assumed to be up.
12 PPP_PAUSE — The number of seconds (in “sleep” format) that it
13 takes the PPPoE client to bring up the new connection. You’ll
14 have to time this on your system.
15 SYSTEMSTARTUP — The number of seconds (in “sleep” format) to
16 wait after starting the daemon before testing for the network con-
17 nection. If you start this daemon at boot time, you should always
18 start it after the network interfaces are up. This makes sure that
19 check-pppoe doesn’t get into any unfortunate interaction with
20 the network startup scripts.
21 TRY — The integer number of times that check-pppoe will attempt
22 to restart the PPPoE client in the hopes that it won’t be necessary
23 to actually cycle power on the modem.

24
25 Listing 2 contains the code of the check-pppoe daemon. It’s pretty
26 straightforward. Note that the `logger` command is used to emit sta-
27 tus information. If additional notification of a system manager is
28 needed, then modification of `syslog.conf` to provide the necessary
29 hooks to the appropriate notification mechanism will be necessary.
30 The last piece is Listing 3, which is the daemon startup shell script.

31 Installing check-pppoe

32 I don’t have enough “round tuits” left to do the right thing and
33 write a Debian package for this, but it’s on my list. Installation is
34 pretty straightforward:
35

- 36 1. Copy Listing 2 to `/usr/local/sbin/check-pppoe.daemon`. If you
37 prefer, you can copy it to `/usr/sbin`, `/sbin`, or `/opt/sbin`.
- 38 2. Copy Listing 1 to `/etc/check-pppoe/check-pppoe.conf`.
- 39 3. Copy Listing 3 to `/etc/init.d/check-pppoe`.

40
41 You should make symbolic links as appropriate from the `/etc/rc`
42 directories for the run levels, which should have check-pppoe
43 started and killed.
44

45 Conclusion

46 For \$40 in hardware and less than a day of time, I solved a prob-
47 lem that had driven me nuts for years. This isn’t the most robust or
48 general solution possible, but it works far better than anything else
49 I could have had for the money. So far, I’ve had no further need for
50 automation of this sort of systems administration task, but given
51 how easy this solution was, the next time I won’t hesitate. Now I
52 can leave the office and not worry about the state of the network.
53 It’s all being taken care of for me.

54 This code is available for download from the *Sys Admin* Web
55 site and from:

56 <http://www.csworks.com/download/StayingAlive.tar.bz2>

59 Resources

60 Bottle Rocket software — <http://mlug.missouri.edu/~tymm/>
61 Debian Linux — <http://www.debian.org>
62

1 X10 “Firecracker” Kit —
2 http://www.x10.com/software/automation_software.html
3
4 *Dick Munroe is a software engineer, architect, and consultant with nearly*
5 *40 years of software and project experience ranging from the sublime to the*
6 *ridiculous. He grinds code and wood from his offices at Cottage Software*
7 *Works in Belmont, Massachusetts, Havana, Florida, and Guanaja,*
8 *Honduras. When playing, he can frequently be found at the top of moun-*
9 *tains wondering whether they will find the pieces come springtime or deep*
10 *in the water worrying whether that shark is really as hungry as it looks. He*
11 *can be contacted at: munroe@csworks.com.*

Please provide
Listing captions.

Listing 1 ???

```
#
# Copyright Dick Munroe, munroe@csworks.com, 2005 all rights reserved.
#
# Use of this program is granted under the Gnu Public License provided
# this copyright remains in place.
#

CYCLETIME=30s
DSL_PROVIDER=dsl-provider

#
# The X10 module address that controls power to the modem.
#

HOUSECODE=A1

#
# These numbers have been adjusted to work with the WireSpeed Westlink
# DSL modem provided by Verizon.
#
# Adjust HOUSECODE_OFF_PAUSE so that the off power cycle is long
# enough to guarantee that your equipment is reset.
#
# Adjust HOUSECODE_ON_PAUSE so that the equipment has enough time to
# synchronize with your line before attempting to start PPPoE.
#

HOUSECODE_OFF_PAUSE=15s
HOUSECODE_ON_PAUSE=30s

#
# The daemon is rigged to delay more and more time after failing an
# attempted power cycle. This keeps it from simply flipping the
# switch incessantly BUT it has the effect of slowing down response to
# the line coming up. I figured that 1/2 hour was the maximum time I
# would delay. Adjust it to suit your needs. Most line outages are
# pretty short where I am (the longest I've seen so far is 1.5 hours)
# which got me to about a 15 minute delay for the next powercycle.
#

POWERCYCLETIME=0
POWERCYCLETIMEINCREMENT=60
POWERCYCLETIMEMAXIMUM=1800

#
# The ppp interface to be tested.
#

PPP=ppp0

#
# The amount of time it takes for the ppp daemon to get a line up.
#

PPP_PAUSE=20s

#
# The number of seconds to wait after the system starts before
# actually attempting to check for the connection.
#

SYSTEMSTARTUP=600s

#
# The number of tries to make before giving up and either trying a
# power cycle or waiting for a while to try the next power cycle.
#

TRY=3
```

Listing 2 ???

```
#!/bin/bash
#
# Copyright Dick Munroe, munroe@csworks.com, 2005 all rights reserved.
#
# Use of this program is granted under the Gnu Public License provided
# this copyright remains in place.
#

PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin

scriptName=${0##*/}
scriptBaseName=${scriptName%.*}

. /etc/$scriptBaseName/$scriptBaseName.conf

#
# Give the system some "time" to get ppp up. It may take longer than
# the default cycle and we don't want to get two ppp interface started.
#

sleep $SYSTEMSTARTUP

#
# Try a simple ppp restart. Power cycling and other things are external
# to this logic. Call this only if the PPP interface really IS down.
#

function restartPPPoE()
{
    I=0

    while [ $I -lt $TRY ]
    do
        poff -r $DSL_PROVIDER 2>/dev/null || pon $DSL_PROVIDER

        sleep $PPP_PAUSE

        ifconfig $PPP >/dev/null 2>&1

        theStatus=?

        #
        # Bail if the interface is there
        #

        if [ $theStatus -eq 0 ]
        then
            return 0
        fi

        I=$((I + 1))
        done

        return 1
    }

#
# Record the pid in the run file
#

echo $$ >/var/run/$scriptBaseName.pid

powercycletime=$POWERCYCLETIME

while [ 1 -eq 1 ]
do
    sleep $CYCLETIME

    ifconfig $PPP >/dev/null 2>&1
    theStatus=?

    if [ $theStatus -ne 0 ]
    then
        logger -p local2.warn $PPP has dropped, attempting restart
        restartPPPoE
        theStatus=?
        if [ $theStatus -ne 0 ]
        then
            #
            # got here and the easy way didn't work, try cycling power,
            # then restarting it.

```

Listing 2 *continued*

```
#

logger -p local2.warn Cycling power on X10 interface $HOUSECODE.

br $HOUSECODE off

sleep $HOUSECODE_OFF_PAUSE

br $HOUSECODE on

sleep $HOUSECODE_ON_PAUSE

logger -p local2.warn $PPP still down after power cycle, \
    attempting restart.

restartPPPoE

theStatus=?
if [ $theStatus -eq 0 ]
then
    powercycletime=$POWERCYCLETIME
    logger -p local2.info $PPP is back up.
else
    logger -p local2.emerg $PPP is not back up after $TRY, \
        powercycle, $TRY attempts.
    powercycletime=$((powercycletime + $POWERCYCLETIMEINCREMENT))
    if [ $powercycletime -gt $POWERCYCLETIMEMAXIMUM ]
    then
        powercycletime=$POWERCYCLETIMEMAXIMUM
    fi

    sleep ${powercycletime}s
    fi
else
    powercycletime=$POWERCYCLETIME
    logger -p local2.info $PPP is back up.
fi
fi
done
```

Listing 3 ???

```
#!/bin/sh

PATH=/usr/local/sbin:/usr/sbin:/sbin:/opt/sbin
N=$0
CHECKPPPPOE=`type -p check-pppoe.daemon`

#
# Exit if the PPPoE checking daemon isn't installed.
#

[ -z "$CHECKPPPPOE" ] && exit 1

pppoeName=${CHECKPPPPOE##*/}
pppoeBaseName=${pppoeName%.*}

case "$1" in
restart)
    $N stop
    $N start
    ;;
start)
    echo "Starting daemon: check-pppoe"
    start-stop-daemon --start \
        --pidfile /var/run/$pppoeBaseName.pid \
        --background \
        --startas $CHECKPPPPOE
    ;;
stop)
    echo "Stopping daemon: check-pppoe"
    start-stop-daemon --stop \
        --quiet \
        --pidfile /var/run/$pppoeBaseName.pid \
        1>/dev/null
    ;;
*)
    echo "Usage: $N {start|stop|restart}" >&2
    exit 1
    ;;
esac

exit 0
```