Hannes Tietz, Fotolia

Reconfigure your network connection with isp-switch

LIFELINE

When an Internet provider goes down, users suffer. Alternatively, users can immediately switch to another ISP. We'll show you a Perl script that can help you reconfigure your computer to make the switch.

BY MICHAEL SCHILLI

In California, two major competitors currently offer Internet access to private households: AT&T with broadband DSL and Comcast with broadband Internet via the TV cable. A DSL or cable modem picks up data off the wire, and a router connects the machines on the home network with the Internet (Figure 1). Both providers have their advantages and disadvantages.

The cable solution suffers from too many kids in the neighborhood playing World of Warcraft because bandwidth drops as more users join the network. The Web Hog! advertisement [1] by DSL provider Southwestern Bell gives a humorous illustration of how peaceful neighbors in a small town can go nuts overnight because they all suspect each other of slowing down the Internet service. DSL access via the phone company is cheaper, assuming you go for the slow basic version, and everyone gets the same bandwidth. However, availability and throughput depend on the distance to the next node.

Neither of these providers is 100 percent reliable and problems do occur. A power outage, a sys admin asleep on the job, a construction worker drilling into a cable, and down goes the Internet – it's not a good situation if you need access in a hurry.

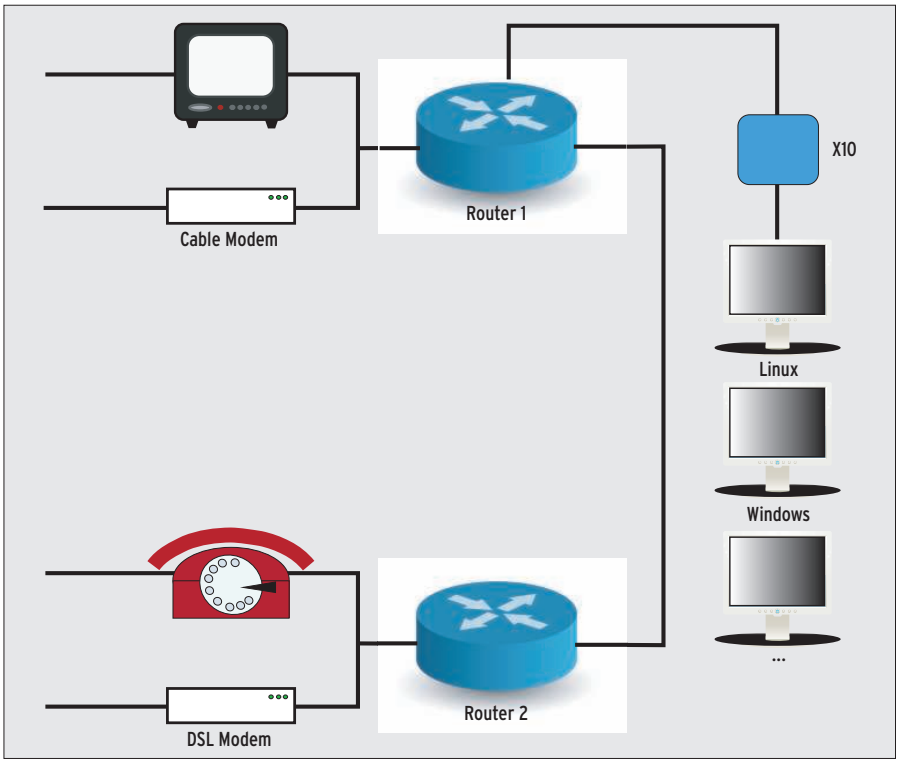

Figure 1: A view of the home network at the Perlmeister Studio.

Considering that both DSL and cable-based Internet cost about US$ 20 a month, I decided to order both – if one provider goes down, I just switch to the other one on the fly.

The isp-switch script (Listing 1) [2] expects either *cable* or *dsl* as a command-line parameter and performs the steps needed for switching. In the isp-switch script, line 12 determines the values these parameters can assume.

If the provider name is unknown, the script quits after a call to *pod2usage()* to output an error message along with a short HOW-TO.

The soft reference – *$switch_to->()* – in line 39 calls one of the functions defined farther down: *cable()* or *dsl()*.

To keep the Perl interpreter from complaining about this dirty trick in strict mode, the script has to reduce the strictness level with *no strict 'refs'*. The *eval* construct surrounding the call handles any errors and allows the script to run even if it cannot complete every single step successfully. The script uses *Log::Log4perl* for status and error messages; the *ALWAYS* macro used in line 60 was added in version 1.13, so line 9 asks for at least this version.

Patch Me!

My Linux machine has a static 192.xxx IP on the local network; that is, it does not use DHCP. Fedora configures the default gateway – the router that sends requests out onto the Internet – in */etc/sysconfig/network*. Figure 2 shows the original content of the file. The gateway is a G54-type device by Buffalo, which originally cost US$ 30 and has reliably connected the internal network in the Perlmeister Studio with the DSL modem, and thus the Internet, for years.

If I need to switch to the cable connection because of a DSL outage, the default gateway setting needs to change to point to a different router with the static IP 192.168.10.98.

How can I modify the configuration files and save the original state to be able to restore it at any time?

The CPAN *Config::Patch* module adds patches to system configuration files and stores the original content as a Base 64-encoded comment inline. If you remove the patch later, the module reads the Base 64 encoding, extracts the original text, and restores it.

Figure 3 shows the *network* file after applying the patch. The *gateway_patch* function defined in line 74 expects the IP address for the gateway as a parameter. The name of the file to patch and a key are passed to the *Config::Patch* object's constructor. Typically, this string is the project or application name.

The keys allow the module to distinguish between different patches for the same file and to process them separately. The module marks patched areas as no-go areas to prevent overlapping patches.

Config::Patch can optionally apply patches at the start of a file, between two lines, or at the end of the file. The *replace* method will even replace a line or part of a line that you reference by means of a regular expression with a replacement line. The method call

```
$patcher->replace(
 qr(^GATEWAY=.*)m,
 "GATEWAY=$ip");
```

searches for a line that starts with *GATEWAY=*. Because the file might contain

```
mschilli@mybox:/etc/sysconfig
NETWORKING=yes
HOSTNAME=mybox

  # DSL
GATEWAY=192.168.10.1
~
~
~
~
~
~
"/etc/sysconfig/network" 5L, 60C   1,1          All
```

Figure 2: The original version of /etc/sysconfig/network before applying the patch.

```
mschilli@mybox:/etc/sysconfig
NETWORKING=yes
HOSTNAME=mybox

  # DSL
#(Config::Patch-isp-switch-replace)
GATEWAY=192.168.10.98
#(Config::Patch::replace)
# R0FURVdBWT0xOTIuMTY4LjEwLjEK
#(Config::Patch::replace)
#(Config::Patch-isp-switch-replace)
~
"network" 10L, 216C                1,1          All
```

Figure 3: Modified configuration file after patching via the Config::Patch module.

Listing 1: isp-switch

```
001 #!/usr/bin/perl -w
002 use strict;
003 use Getopt::Std;
004 use Pod::Usage;
005 use Sysadm::Install qw(:all);
006 use Config::Patch;
007 use Buffalo::G54;
008 use X10::Home;
009 use Log::Log4perl 1.13
010   qw(:easy);
011
012 my @isps = qw(cable dsl);
013 my ($switch_to) = @ARGV;
014
015 Log::Log4perl->easy_init(
016   $INFO);
017
018 if ( !defined $switch_to ) {
019   pod2usage(
020   "Which ISP to switch to?");
021 }
022
023 if (
024   !grep { $_ eq $switch_to }
025   @isps ) {
026   pod2usage(
027     "Unknown isp, use ",
028     join( ", ", @isps )
029   );
030 }
031
032 my $x10 = X10::Home->new();
033
034 my $P =
035   password_read(
036   "Router password: ");
037
038 no strict 'refs';
039 eval { $switch_to->(); };
040 network_restart();
041
042 ################################
043 sub dsl {
044 ################################
045   gateway_patch(
046     "192.168.10.1");
047   $x10->send( "bridge",
048     "off" );
049   dhcp("on");
050 }
051
052 ################################
053 sub cable {
054 ################################
055   gateway_patch(
056     "192.168.10.98");
057   $x10->send( "bridge",
058     "on" );
059   dhcp("off");
060   ALWAYS "Waiting for ",
061     "bridge to start up";
062   sleep 20;
063 }
064
065 ################################
066 sub network_restart {
067 ################################
068   tap "sudo",
069 "/etc/rc.d/init.d/network",
070     "restart";
071 }
072
073 ################################
074 sub gateway_patch {
075 ################################
076   my ($ip) = @_;
077
078   my $patcher =
079     Config::Patch->new(
080     file =>
081     "/etc/sysconfig/network",
082     key => "isp-switch",
083     );
084
085   if ($patcher->patched()) {
086     # patched already?
087     # Remove old patch
088     $patcher->remove();
089   }
090
091   $patcher->replace(
092     qr(^GATEWAY=.*)m,
093     "GATEWAY=$ip"
094   );
095 }
096
097 ################################
098 sub dhcp {
099 ################################
100   my ($onoff) = @_;
101
102   DEBUG
103     "Setting dhcp to $onoff";
104
105   my $b =
106     Buffalo::G54->new();
107   DEBUG "Connecting";
108   $b->connect(
109     password => $P );
110
111   if ( defined $onoff ) {
112     INFO
113     "Setting DHCP to $onoff";
114     $b->dhcp($onoff);
115   }
116
117   INFO "DHCP is now ",
118     $b->dhcp()
119     ? "on" : "off";
120 }
121
122 __END__
123
124 =head1 NAME
125
126   isp-switch - Cable or DSL?
127
128 =head1 SYNOPSIS
129
130   isp-switch [dsl|cable]
131
132 =head1 DESCRIPTION
133
134 isp-switch switches between
135 Comcast cable and Pacbell
136 DSL.
137
138 =head1 AUTHOR
139
140 2007, Mike Schilli
141 <cpan@perlmeister.com>
```

Wherever you go...

Read Linux Magazine anywhere with a Digital Subscription.
Access articles by logging into our site and downloading PDF files.
Find the Linux solutions you need with an easy keyword search.
Maintain your own paperless archive for convenient offline reading.

...Linux Magazine goes with you!

http://www.linux-magazine.com/DigiSub

multiple lines, the */m* is important to make sure that the meta character ^ really does find all beginnings of lines and not just the first. *Config::Patch* encodes the gateway line as its way of "remembering" before going on to comment out the line and replace it with a gateway entry containing the new IP address.

Calling *$patcher->remove()* converts the file back into its original state; the patcher only needs the file name and the key, both of which are available in the object constructor. The *patched()* method checks whether the file has been patched with the specified key and, if so, returns a value of true.

Power on Command

The cable router is switched off while the Internet connection is managed via DSL. To switch the cable router on, I can use the X10 interface and the CPAN *X10::Home* module.

Figure 4 shows the entry in */etc/x10.conf*, which allows me to address the cable router as *cable_router*.

Both routers are the same type and both provide DHCP services.

When a computer logs on to my home network, it is automatically assigned a dynamic IP and knows which DNS server it has to contact to resolve hostnames. However, two DHCP servers offering the same service at the same time cause no end of trouble. Furthermore, I don't want the computers to use the old router for cable operations but, instead, to contact the new DHCP server with its cable gateway. For this reason, the *Buffalo::G54* CPAN module contacts the router's configuration interface and messes around with screen-scraping techniques with the *WWW::Mechanize* module until it has finally disabled the DHCP server on the DSL router. To do this, it needs the router's administrative password; isp-switch calls the *password_read()* function from the *Sysadm::Install* CPAN module to prompt the user to enter the password.

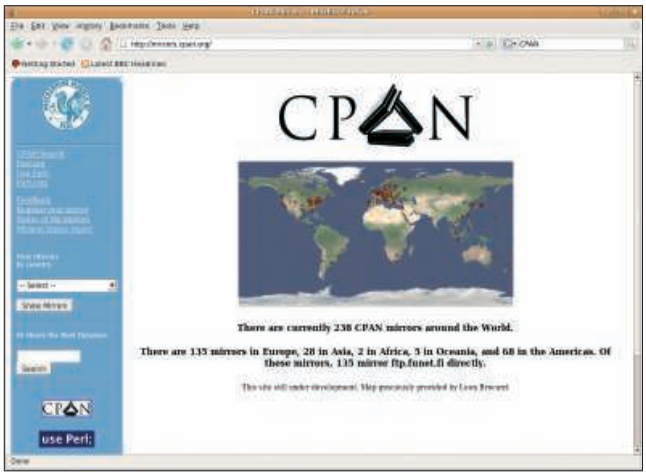

Figure 6: The Comprehensive Perl Archive Network (CPAN) is the source for many of the Perl modules discussed in this column.

In the opposite case, when the *dsl* option is provided to restore the DSL-based connection, the script enables the DHCP server on the DSL router and rigorously powers off the cable router using X10.

```
mschilli@mybox:/etc
mschilli ALL= NOPASSWD:/etc/rc.d/init.d/network
~
"/etc/sudoers " 1L, 48C                1,1           All
```

Figure 5: An entry in /etc/sudoers lets user mschilli start the network and bring it down.

Because it takes awhile for the new router to power up completely, isp-switch *sleeps* for 20 seconds before initiating any more activities.

Restart

Computers with dynamic addresses query the DHCP server to pick up a new IP address after a restart of the network system.

The Linux box with the static IP also has to relaunch its network initialization script to prevent it from sending requests to the old gateway.

To allow this to happen, isp-switch runs the Linux startup script */etc/rc.d/init.d/network* with the *restart* parameter.

This requires root privileges, but a *NOPASSWD* entry for the script in */etc/sudoers* (Figure 5) allows the non-privileged user *mschilli* to reinitialize the network. The *tap* function from the endless treasure troves of the *Sysadm::Install* CPAN module executes the command and absorbs its output.

These three measures allow isp-switch to switch back and forth between the two Internet providers. Although this makes Internet access twice as expensive, it is also twice as reliable.

As an alternative, you could deploy a third router that worked as a gateway and forward requests either to the cable router or DSL router. It could run on a Linux PC or on a WRT54GL-type Linksys router running on FreeWRT.

If you like, you could program a Nagios plugin to regularly check the Internet connection and automatically switch to another provider whenever a problem occurs, hopefully without the end user being any wiser. ■

```
mschilli@mybox:/etc
# x10.conf Configuration File
module: ControlX10::CM11
name:   "/dev/ttyS0"
receivers:
    - name:   cable_router
      code:   K12
      desc:   Cable Router

    - name:   dslmodem
      code:   K11
      desc:   DSL Modem

    - name:   dslrouter
      code:   K9
      desc:   DSL Router
~
"x10.conf " 15L, 292C  1,1            All
```

Figure 4: The cable_router entry in /etc/x10.conf lets X10::Home use an intuitive name to address the cable router.

INFO

[1] "Web Hog" by Southwestern Bell: *http://www.youtube.com/watch?v=ubc7zFSyEbg*

[2] Listings for this article: *ftp://www.linux-magazin.de/pub/listings/magazin/2007/11/Perl*

THE AUTHOR

Michael Schilli works as a Software Developer at Yahoo!, Sunnyvale, California. He wrote "Perl Power" for Addison-Wesley and can be contacted at *mschilli@perlmeister.com*. His homepage is at *http://perlmeister.com*.