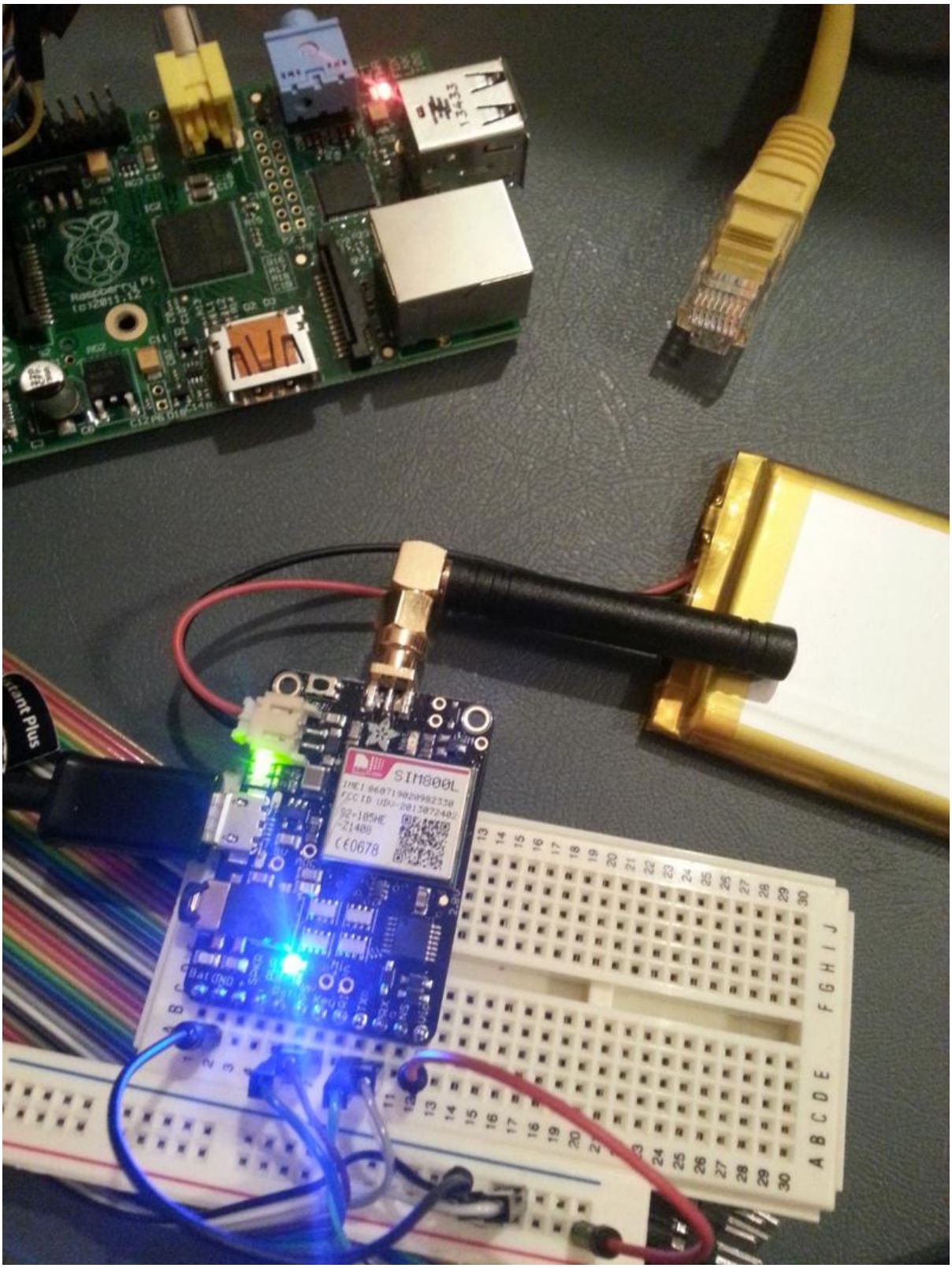
Network Interface Failover using FONA

Created by Adam Kohring

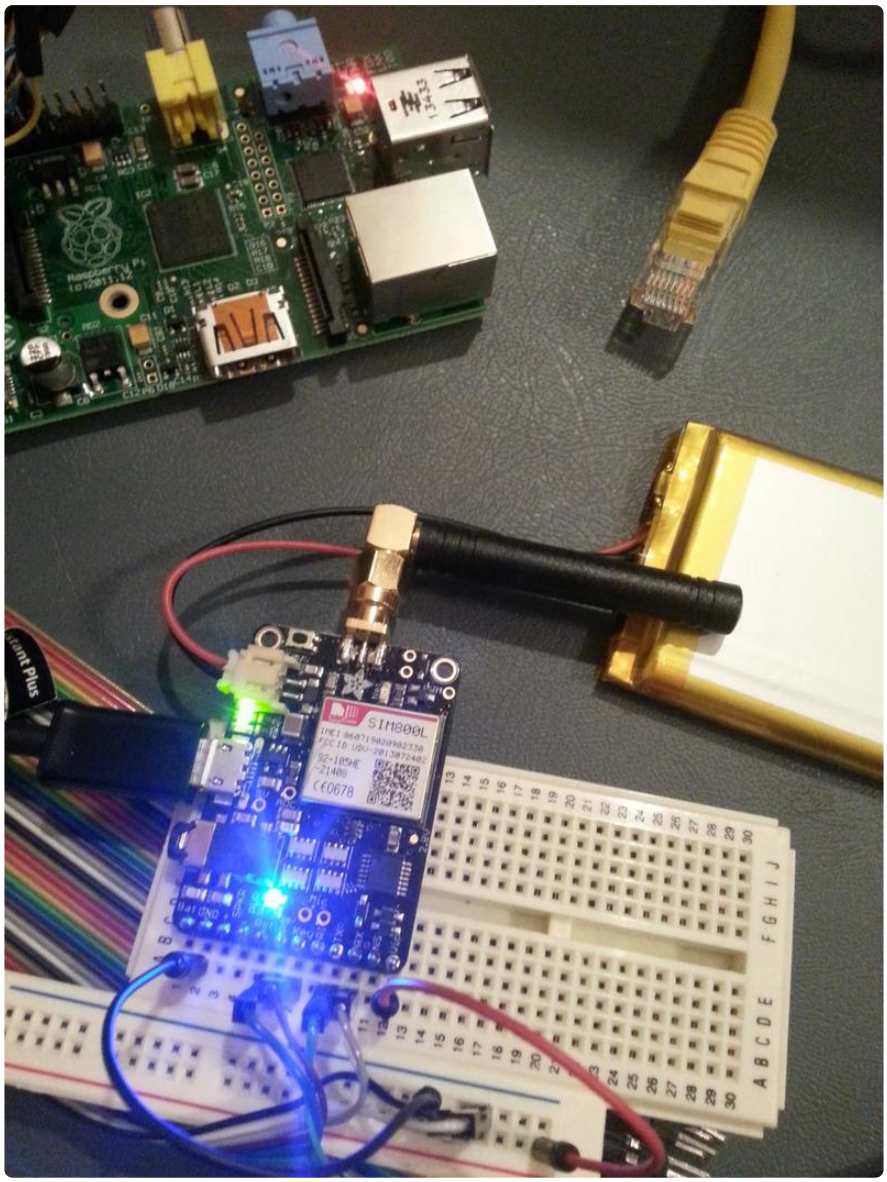


https://learn.adafruit.com/network-interface-failover-using-fona

Last updated on 2023-08-29 02:38:49 PM EDT

Table of Contents

Overview



Do you hate it when the internet goes down in your area? Do you want to be notified when your internet goes down when you are away from home? Do you have a project that needs 100% internet availability where you would need to be notified of service disruption?

This guide will show you how use FONA as a secondary cellular connection to the internet with automatic failover and real time notifications.

I built a home security system to set off an alarm and notify me in the event of an intruder break in. The home security system requires an internet connection to route a message to my phone when the house is under attack. In the event of an internet outage someone could break into my house without my knowledge. To resolve this issue, I wrote a service to monitor my primary internet connection. When the service

detects an internet outage it will automatically reroute all outbound traffic over a cellular network using the Adafruit FONA. A message will then be sent to my phone over the data plan to let me know my that my data plan is actively being used to route traffic. Once the primary internet connection is available, the data plan will no longer be used and all traffic will be automatically routed back to the primary interface.

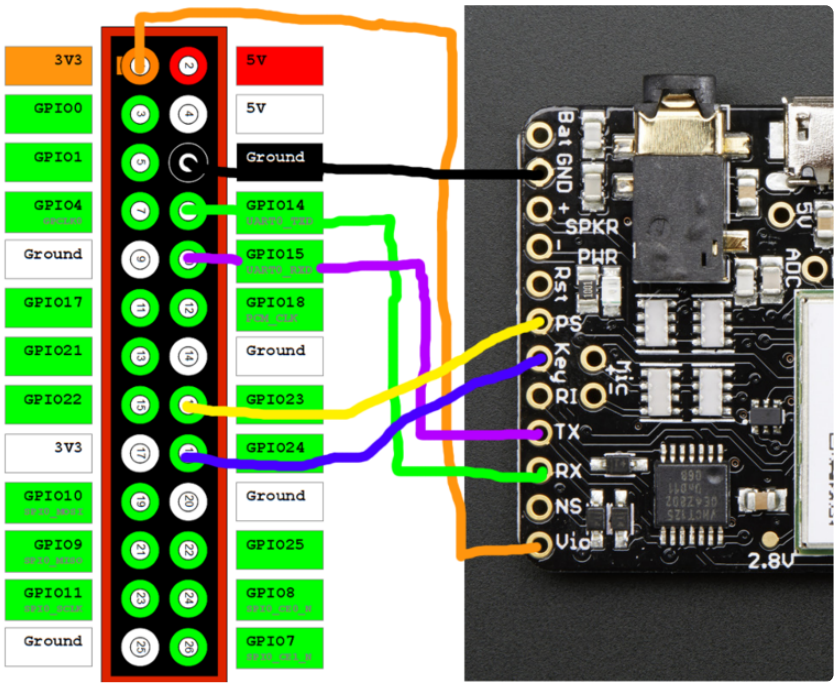
Prerequisites

For Raspberry Pi and BeagleBone Black, follow the FONA Tethering Setup Guide () to install and configure the Point-to-Point Protocol daemon for Fona.

If you are already familiar with the tethering setup guide and installing ifacefailover on a fresh Raspberry Pi, in the next section you can execute the bin/raspberryPiInstall.sh script instead of the bin/install.sh script to automatically disable the kernel's use of the hardware serial connection, install ppp, and download the Fona ppp peer prior to installing the ifacefailover service.

Wiring

Raspberry Pi to Fona



Power Status (PS) and Key pins are tied to GPIO pins so the Fona can be turned on and off programmatically. Your data plan is only used in the event of a failover.

ifacefailover Service

Install the Source Code

We will be using the open source python service ifacefailover to monitor a network interface and reroute all outbound network traffic to Fona in the event of an outage. Clone the ifacefailover repository from GitHub and execute the install.sh script from the bin directory. This will create the appropiate directory structure under /opt, link to the recently checked out source code, install the required python dependencies, and create the startup service /etc/init.d/ifacefailover. The following commands should be executed on the device you wish to install ifacefailover.

```
sudo mkdir -p /opt/git/ifacefailover
sudo git clone https://github.com/shellbit/ifacefailover.git /opt/git/ifacefailover
sudo /opt/git/ifacefailover/bin/install.sh
```

Routes

Routes are how your device connects to the internet and other devices. Routes are established on a network interface through an ethernet cable, wireless card, or even UART pins. ifacefailover routes are configured in the ifacefailover.properties file. You can define as many failover routes as you wish. The order you define the routes in the properties file matters! The first route defined in the properties file is your primary route. The ifacefailover service will always try to restore connection to your primary route in the event of a failover. If your primary route goes down, ifacefailover will try to failover to your secondary route. If your primary route and secondary route are down, ifacefailover will try to failover to your tertiary route, and so on and so forth until it detects an available route. You can view the current routes on a Linux device by running the following command. ifacefailover simply rewrites your default route (0.0.0.0) to a different gateway and interface in the event of a failover.

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     2      0        0 wlan0
```

Route Verifiers

ifacefailover uses Route Verifiers to determine if a route is healthy and capable routing packets. Route verifiers are implemented as python classes and must return a boolean value to determine if the route is available. We will be using an ICMP route

verifier to send ping echo request packets to target servers outside our network. Route verifiers are not limited to ICMP requests. We could have used HTTP, a raw socket connection, or even implemented our own custom protocol to verify the route. Checkout the ifacefailover wiki () on GitHub for more examples and if you write a route verifier that could be used by others, please consider contributing it to the ifacefailover open source project.

Route Handlers

Route Handlers provide hooks into the ifacefailover service. They allow you to log failover events, send event notifications, start or stop running processes, and anything else you want to do throughout the lifecycle of the route. You can associate as many route handlers as you wish to a route. We will be using the Log Handler to log events, the HTTP Handler to send failover events to the Google Cloud Messaging service, and the Fona Raspberry Pi Handler to turn the Fona on and off pre and post failover. Checkout the ifacefailover wiki () on GitHub for more examples and if you write a route handler that could be used by others, please consider contributing it to the ifacefailover open source project.

Property Configuration

Routes, Route Handlers, and Route Verifiers must be configured in the ifacefailover.properties file. To configure a route, open the /opt/ifacefailover/config/ifacefailover.properties file in your favorite editor. This property file contains key/value pairs of properties used by the ifacefailover service. The route configuration format is route.name.property=value. So the property route.primary.gateway=192.168.1.1 means configure a route named primary and set its gateway property to 192.168.1.1. Below are a list of configurable route properties.

- gateway (required): This is the IP address used to access other networks internally or across the internet. For home networks, this is typically 192.168.1.1. For the Fona route, use 0.0.0.0.
- iface (required): This is a string value representing the network interface associated with the route. If your device is connected to the internet via an ethernet cable, this will probably be eth0. The number may vary based on how many network interfaces are on your device. For a wireless connection, you will probably use wlan0. For Fona, use ppp0.
- targets (optional): This is a comma delimited list of IP Addresses used to verify if the route is available. The IP Addresses will be created as static routes (pegged to the defined route interface) on startup. This allows ifacefailover to verify the route by sending requests to the target on the specific interface that failed. For our example, we will be using the Google primary and secondary DNS servers of

8.8.8.8 and 8.8.4.4. You can specify any number of IP addresses. The ICMP Verifier we will be using will only failover if all targets are unavailable. Verifiers are 100% configurable, so you can change this functionality as needed by updating the isRouteAvailable method on the verifier or by creating your own custom verifier.

- verifier (optional): This is a route verifier (as described above) that has been serialized to a Python pickle file. This value should be the absolute path to the serialized pickle file of the route verifier. Check under /opt/ifacefailover/src/verifier for a list of preserialized verifiers.
- verifierDelay (optional): This is the interval in seconds to wait between verification requests. If not specificed, this will default to 0 seconds.
- verifierKwargs (optional): This is a Python dictionary of Keyword Arguments (Kwargs) that will be passed to the route verifier. Use verifierKwargs to externally configure your route verifiers.
- handlers (optional): This is a Python list of route handlers (as described above) that have been serialized to Python pickle files. Each item of the list should be an absolute path to a pickle file. You can specify as many handlers as you want and they will be executed in the order defined. Check under /opt/ifacefailover/src/handler for a list of preserialized handlers.
- handlerKwargs (optional): This is a list of Python dictionaries of Keyword Arguments (Kwargs) that will be passed to the route handler. Use handlerKwargs to externally configure your route handlers. Each dictionary item in the list will passed to the corresponding handler defined in the handler property.

You will need to create the following ifacefailover.properties file under /opt/ifacefailover/config. This will define a primary route on eth0 (ethernet connection) and a secondary route on ppp0 (fona). You can specify any number of routes in this file, they will be loaded in the order specified. Please be sure to update each property value accordingly. If you change the location or name of this file, you will need to update the /etc/init.d/ifacefailover startup script with the new value.

```
# Defines the location of the external Python logging configuration file
log.properties=/opt/ifacefailover/config/log.properties

# The following properties configure the primary route

# This is the gateway of your router
route.primary.gateway=192.168.1.1
# This will probably be eth0 for wired connections and wlan0 for wireless
connections
route.primary.iface=eth0
# We are using the Google primary and secondary DNS servers for route verification
route.primary.targets=8.8.8.8,8.8.4.4
# Wait 10 seconds between verification requests
route.primary.verifierDelay=10
# We are using an ICMP route verificer to send ping echo requests
route.primary.verifier=/opt/ifacefailover/config/icmpRouteVerifier.pkl
```

```
# The ICMP verifier is configured with a response timeout of 1 second and will
retry 2 times
route.primary.verifierKwargs={'timeout':1, 'maxRetry':2}
# We will use the logHandler to log the lifecycle of the route and the httpHandler
to send failover events to a web app
route.primary.handlers=['/opt/ifacefailover/config/logHandler.pkl','/opt/
ifacefailover/config/httpHandler.pkl']
# The logHandler is configured with a name property to log the route name
# The httpHandler is configured with an authentication tuple, a url, and url
suffixes to send different types of event notifications
route.primary.handlerKwargs=[{'name':'primary'}, {'auth':('username', 'password'),
'url':'https://localhost:5000/myapp/contextroot/',
'onConnectionFailedUrlSuffix':'primaryconnectionfailed',
'onConnectionRestoredUrlSuffix':'primaryconnectionrestored'}]


# The following properties configure the secondary route used for failover

# Fona's gateway is 0.0.0.0
route.fona.gateway=0.0.0.0
# Fona is running on the ppp0 interface
route.fona.iface=ppp0
# Fona is configured with a logHandler to log lifecycle events
# A fonaRaspberryPiHandler to turn the Fona on and off and start and stop the point-
to-point protocol daemon
# And an httpHandler to send failover events
route.fona.handlers=['/opt/ifacefailover/config/logHandler.pkl','/opt/ifacefailover/
config/fonaRaspberryPiHandler.pkl','/opt/ifacefailover/config/httpHandler.pkl']
# The logHandler is configured with a name property to log the route name
# The fonaRaspberryPiHandler is configured with the GPIO pin number connected to
the Fona PS pin and the GPIO pin number connected to the Fona Key pin
# The httpHandler is configured with an authentication tuple, a url, and url
suffixes to send different types of event notifications
route.fona.handlerKwargs=[{'name':'fona'},{'powerStatusPin':23, 'keyPin':24},
{'auth':('username', 'password'), 'url':'https://localhost:5000/myapp/
contextroot/', 'onConnectedUrlSuffix':'fonaconnected'}]
```

Logging Configuration

ifacefailover loads its Python logging configuration from an external property file. The location of the property file must be specified as a property in /opt/ifacefailover/config/ifacefailover.properties. The property name is log.properties and the value should be the absolute path to the logging configuration file. The default file is /opt/ifacefailover/config/log.properties. Read the Python Reference Documentation () for more information on the logging configuration file format.

General Packet Radio Service (GPRS) Configuration

If you followed the prerequisites section of the guide, you should already have chatscripts installed with a gprs config at /etc/chatscripts/gprs. Use the root user to edit this file and reduce the default timeout value from 12 to 1 for a quicker timeout.

```
# cease if the modem is not attached to the network yet
ABORT           "+CGATT: 0"

""              AT
TIMEOUT         1
```

```
OK              ATH
OK              ATE1
```

Point-to-Point Protocol (PPP) Configuration

If you followed the prerequisites section of the guide, you should already have Point-to-Point Protocol Daemon (pppd) installed with a fona peer listed under /etc/ppp/peers. Use the root user to add the following configuration properties to /etc/ppp/peers/fona.

```
# If this option is given, pppd will send an LCP echo-request frame to the
# peer every n seconds. Normally the peer should respond to the echo-request
# by sending an echo-reply. This option can be used with the
# lcp-echo-failure option to detect that the peer is no longer connected.
lcp-echo-interval 5

# If this option is given, pppd will presume the peer to be dead if n
# LCP echo-requests are sent without receiving a valid LCP echo-reply.
# If this happens, pppd will terminate the connection.  Use of this
# option requires a non-zero value for the lcp-echo-interval parameter.
# This option can be used to enable pppd to terminate after the physical
# connection has been broken (e.g., the modem has hung up) in
# situations where no hardware modem control lines are available.
lcp-echo-failure 1

# Terminate after n consecutive failed connection attempts.
# A value of 0 means no limit. The default value is 10.
maxfail 0

# Specifies how many seconds to wait before re-initiating the link after
# it terminates.  This option only has any effect if the persist or demand
# option is used.  The holdoff period is not applied if the link was
# terminated because it was idle.
holdoff 1
```

Restart the Service

Once you have updated all the ifacefailover properties, you will need to restart the service.

```
$ sudo service ifacefailover restart
Restarting ifacefailover
Stopping ifacefailover
Starting ifacefailover
```

Log Verification

You should now see a log statement every 10 seconds letting you know ifacefailover is monitoring your primary route.

```
$ tail -f /opt/ifacefailover/logs/out.log
2014-10-02 03:00:10,075 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
2014-10-02 03:00:20,163 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
```

```
2014-10-02 03:00:30,252 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
^C

Press ctrl+c to exit
```

Primary Route Verification

We also need to verify ifacefailover service setup our primary route correctly by running the following command. We should see the primary route with default destination of 0.0.0.0 on the correct interface and two static routes on this interface with destinations set to the Google primary and secondary DNS servers.

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
8.8.4.4         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
8.8.8.8         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
```

You should be able to ping adafruit through your primary interface.

```
$ ping -I eth0 adafruit.com
PING adafruit.com (207.58.139.247) from 192.168.1.143 eth0: 56(84) bytes of data.
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=1 ttl=55 time=38.3 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=2 ttl=55 time=38.1 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=3 ttl=55 time=38.0 ms
^C
--- adafruit.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 38.042/38.189/38.366/0.208 ms
```

Simulate a Network Outage

Unplug the ethernet cable from your modem. Within 10 seconds, ifacefailover should detect your primary route is down and failover to Fona. A blue light on the Fona will turn on indicating that Fona is booted and running. A red light will start blinking to indicate signal strength. You should also see the following output in the logs. This can take up to 5 minutes so be patient.

```
$ tail -f /opt/ifacefailover/logs/out.log
2014-10-03 01:18:01,179 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
2014-10-03 01:18:15,203 root        : INFO upRoutes=; downRoutes=8.8.8.8, 8.8.4.4
2014-10-03 01:18:15,206 root        : INFO Tearing down connection to primary
2014-10-03 01:18:15,210 root        : INFO Setting up connection to fona
2014-10-03 01:18:18,612 root        : INFO Adding route ppp0; attempt 1 of 120
2014-10-03 01:18:18,620 root        : ERROR (19, 'No such device')
2014-10-03 01:18:19,131 root        : INFO Adding route ppp0; attempt 2 of 120
2014-10-03 01:18:19,135 root        : ERROR (19, 'No such device')
2014-10-03 01:18:19,639 root        : INFO Adding route ppp0; attempt 3 of 120
2014-10-03 01:18:19,643 root        : ERROR (19, 'No such device')

...
```

```
2014-10-03 01:18:35,892 root        : INFO Adding route ppp0; attempt 35 of 120
2014-10-03 01:18:35,895 root        : ERROR (19, 'No such device')
2014-10-03 01:18:36,400 root        : INFO Adding route ppp0; attempt 36 of 120
2014-10-03 01:18:36,404 root        : INFO Connection failed on primary
2014-10-03 01:18:36,407 root        : INFO Attempting POST request 1 of 120
2014-10-03 01:18:36,734 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:18:56,762 root        : ERROR ('Connection aborted.', gaierror(-2,
'Name or service not known'))
2014-10-03 01:18:57,266 root        : INFO Attempting POST request 2 of 120
2014-10-03 01:18:57,287 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:19:17,309 root        : ERROR ('Connection aborted.', gaierror(-2,
'Name or service not known'))
2014-10-03 01:19:17,815 root        : INFO Attempting POST request 3 of 120
2014-10-03 01:19:17,836 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:19:37,859 root        : ERROR ('Connection aborted.', gaierror(-2,
'Name or service not known'))

...

2014-10-03 01:22:22,830 root        : INFO Attempting POST request 12 of 120
2014-10-03 01:22:22,868 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:22:39,225 requests.packages.urllib3.connectionpool: DEBUG "POST /app/
event/primaryconnectionfailed HTTP/1.1" 200 27
2014-10-03 01:22:39,260 root        : INFO Failing over to fona
2014-10-03 01:22:39,263 root        : INFO Attempting POST request 1 of 120
2014-10-03 01:22:39,285 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:22:51,506 requests.packages.urllib3.connectionpool: DEBUG "POST /app/
event/fonaconnected HTTP/1.1" 200 27
2014-10-03 01:22:55,129 root        : INFO upRoutes=; downRoutes=8.8.8.8, 8.8.4.4
2014-10-03 01:22:58,129 root        : INFO upRoutes=; downRoutes=8.8.8.8, 8.8.4.4
2014-10-03 01:23:01,129 root        : INFO upRoutes=; downRoutes=8.8.8.8, 8.8.4.4
```

You should now see the ppp0 interface (Fona) defined as your default destination and your static routes should still be visible.

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         0.0.0.0         0.0.0.0         U     0      0        0 ppp0
8.8.4.4         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
8.8.8.8         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
10.64.64.64     0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
```

You should be able to ping adafruit.com using your new default destination (Fona) but not by specifying your primary route interface (Ethernet).

```
$ ping adafruit.com
PING adafruit.com (207.58.139.247) 56(84) bytes of data.
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=1 ttl=46 time=1001 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=2 ttl=46 time=559 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=3 ttl=46 time=555 ms
^C64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=4 ttl=46 time=555 ms

--- adafruit.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3284ms
rtt min/avg/max/mdev = 555.530/668.125/1001.435/192.446 ms, pipe 2
```

```
$ ping -I ppp0 adafruit.com
PING adafruit.com (207.58.139.247) from 100.117.205.166 ppp0: 56(84) bytes of data.
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=1 ttl=46 time=959 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=2 ttl=46 time=560 ms
64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=3 ttl=46 time=560 ms
^C64 bytes from vps3.ladyada.net (207.58.139.247): icmp_req=4 ttl=46 time=560 ms

--- adafruit.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3638ms
rtt min/avg/max/mdev = 560.197/660.109/959.702/172.970 ms


$ ping -I eth0 adafruit.com
PING adafruit.com (207.58.139.247) from 100.117.205.166 eth0: 56(84) bytes of data.
From 192.168.1.143 icmp_seq=1 Destination Host Unreachable
From 192.168.1.143 icmp_seq=2 Destination Host Unreachable
From 192.168.1.143 icmp_seq=3 Destination Host Unreachable
^C
--- adafruit.com ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5146ms
pipe 4
```

Automatic Recovery from a Network Outage

Plug the ethernet cable back into your modem. Within 10 seconds, ifacefailover should detect your primary route is back online and turn off Fona. You should no longer see the solid blue or the red blinking lights on the Fona. You should see the following output in the logs.

```
$ tail -f /opt/ifacefailover/logs/out.log
2014-10-03 01:24:52,139 root        : INFO upRoutes=; downRoutes=8.8.8.8, 8.8.4.4
2014-10-03 01:24:53,970 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
2014-10-03 01:24:53,973 root        : INFO Tearing down connection to fona
2014-10-03 01:24:56,272 root        : INFO Setting up connection to primary
2014-10-03 01:24:56,279 root        : INFO Adding route eth0; attempt 1 of 120
2014-10-03 01:24:56,282 root        : INFO Connection restored to primary
2014-10-03 01:24:56,285 root        : INFO Attempting POST request 1 of 120
2014-10-03 01:24:56,318 requests.packages.urllib3.connectionpool: INFO Starting new
HTTPS connection (1): localhost
2014-10-03 01:25:03,350 requests.packages.urllib3.connectionpool: DEBUG "POST /app/
event/primaryconnectionrestored HTTP/1.1" 200 27
2014-10-03 01:25:13,544 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
2014-10-03 01:25:23,643 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
2014-10-03 01:25:33,734 root        : INFO upRoutes=8.8.8.8, 8.8.4.4; downRoutes=
```

And your default destination should now be pointed back to your primary route.

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
8.8.4.4         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
8.8.8.8         192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
```