Internet Monitor Service

Note 4-1-2025: this doc is slightly dated, but it gives the history and rational of the project.

Overview of Operation

I have been hearing people complain about their Internet access for years and years, especially in a remote community like the mountains we live in. There is really, really no way to address these problem complaints without a “performance monitoring” capability. All telecom equipment was built with “performance monitoring” built in – you could access the management system controlling it and ask for “performance” over the last hour, the last day, etc etc. It was extensive and built into the protocol of the transmission – whether DS1, DS3, OC3c, ATM etc.

There is also a built in “performance monitoring” of the ADSL/VDSL line as well, but the consumer is not able to access this in general. It IS possible in most ADSL/VDSL modems to dive into the modem and get some of these statistics, but interpreting them, much less accessing them and understanding them is far beyond the capability of the average consumer.

The Internet is a “statistically multiplexed” transmission environment, completely unlike the early “circuit-switched” transmission used by the old “POTS” telephone infrastructure. In the POTs infrastructure, when you called up your friend in CA, you had a “guaranteed” circuit, and a “guaranteed” bandwidth to carry your phone call. Your next door neighbor could have been downloading the encyclopaedia Britannica and it would have made no difference. The reason I have quoted the “guaranteed” is that even in this day and age, the “circuit-switched” phone calls are more than likely carried across the “statistically multiplexed” internet – they just build your “virtual circuit” to make it LOOK like a “circuit-switched” call.

If this is all very confusing, you should not feel alone. The Internet is much like the Post Office. It is an electronic post office! All your communications are chopped into “electronic” post cards and dropped into the “electronic” post office infrastructure [the Internet]. Sometimes your letter rides a jet plane, sometimes a truck, sometimes even a mail carrier. And your letters sometimes crash and burn and get lost. You can see that there is not a “circuit” that your electronic post cards follow to get to the other end, such as a water pipe or an electrical wire! They may get lost, they get stored overnight in mailboxes.

By the way, the electronic post cards are called “IP packets.” Each IP packet has a destination address and a return address, much like a real post card dropped in the mail. Each direction of transmission is entirely separated! The far end knows how to get back to the source by looking at the return address - called the source IP address. And then it drops the IP packet back into the same electronic post office to make its way [or not] back to the source.

There are all kinds of ways to try to monitor the success/failure of these electronic post cards. We can subscribe to “guaranteed delivery” or use “tracking” or we can hire couriers to deliver them personally -that is kind of like an old time circuit. The traditional way in the Internet to check the availability of a “circuit” to the other end is to send an ICMP Echo Request IP packet, otherwise known as a Ping request to the destination address. If the other end receives the Echo Request packet, he returns an ICMP Echo Response packet, otherwise known as a ping reply. The intermittent, statistical nature of the Internet, and its NON circuit transmission architecture makes this valid pretty much just for the single ping packet request/response. Nevertheless, if the intervening equipment behaves pretty much the same from minute to minute, this ping success/failure is a decent indication of Internet connectivity. And it is pretty much the only technique we have at the user/PC layer. Just consider how many billions of transistors and likely photons, and millions of lines of computer code have to be functioning correctly for your bits to make it from here to CA. Truly astonishing when you think about it. Most people certainly dont. But me, being a pioneer in this business, tend to think of all those lonely transistors, photons and all those BUGGY lines of code needed to get your packet from here to there. I am astonished every time I think about it.

So I have developed a series of scripts, one of which does simple pings and record the results for collection and display. Selecting the parameters for the testing is not any kind of a guarantee of results. You just don't want the background monitoring to interfere with the actual use of the Internet. The other scripts record change of (Internet) IP address, boot times, speedtests, and other tests yet to be determined.

This is certainly not a unique service, but I have not found a reliable way to do it without doing some work. I have done this work... in this work. It will continue to be a work in progress. It may or may not have ultimate value. People tend to completely ignore their Internet when it is functioning, and then when they have problems it is of course all interfering – they can get pretty much nothing done. It is becoming almost like losing electrical power, it is so important. This “service” hopes to do some “performance monitoring” on your Internet link.

To emphasize for a “final” time.... There is no way to accurately measure the performance of a service like the statistically multiplexed Internet, which can change instantaneously in its character. These tests I am doing are longer term tests, which assume that the Internet is performing in a similar way from micro moment to micro moment. We can gauge things like links repeatedly dropping or underperforming for longer periods of time [minutes-hours], and record some parameters which may

give us a clue as to the problem.

There are many parameters being collected and displayed [as of 11-4-2016].

There are three modes of operation for “imonitor.” *The initial implementation -mode 1-"imonitor"- was for the Raspberry Pi to independently collect statistics, copy them to the on-board web root for local interrogation, and also to establish connection to an external server where they could be collected from any Internet location*. This was meant as a development phase using about 20+ guinea pigs spread across the US on different ISPs. It is still ongoing.

Mode 2 -"Imonitorg" - will remove the external server requirement so that the collection and display of statistics is entirely local –there is no connection to an external server. This completely preserves the privacy of the collection and interrogation of the statistics. Users do not need to worry about any communication with outside sources. Of course, once the communication with the external server is deleted, there is no way to upgrade/enhance the service.

Mode 3 - "iotsnoop" - this was added in 2024. It uses a pi4 and turns the wifi into an APN so that IOT clients can authenticate to the SSID, and then the iotsnoop scripts perform capture of all IP packets on the APN to allow for interrogation and display of stats for the IOT devices. "Iotsnoop" contains all the previous imonitorg scripting [except pihole and wifi monitoring], so it represents a way to monitor/report on stats for both your Internet/Local networks, as well as your IOT APN.

All modes will have a way to deliver a daily email summary to a user specified address using your gmail address as a relay [gmail account necessary plus 2FA on the account].

There are basically three tests that are run each minute. The dailyping scripts “at*” are run via a user cron for each of 23 hours starting at 2AM. These dailyping tests ping a local Internet router by IP address, which is automatically determined each night by tracerouting to a far address and selecting the nearest pingable hop counting down from hop 5. These errors start from scratch each morning at 2AM and run till 1AM the next morning, when they are reset. This is a pretty low level error test, using the IP address [no DNS resolution used], and a local, highly available router on a very fast Internet connection. It is possible to "LOCK" this mode and specify your own IP address. This test represents a "near ICMP" or "ISP specific" target.

The second test is run every minute of the day, 24 hours. These counters will last for days, weeks -until the monthly reboot of the Raspberry Pi. These use "TCP pings" using a round robin of the top100 web addresses. It thus represent the fact that your DNS server is responding and the web site is responding. This represents a higher level type of test. This test also has a "LOCK" mode where you can specifiy a specific TCP ping target. The top100web file is the US top 100, but foreign users can modify this file to their own country /root/top100websites.txt. The number of entries must be exactly 100.

There is a third mode which uses ICMP ping targets which can be specified by the user by creating a special file "CustPingSites.newtxt" and placing this file in the /home/pi/tests/customerpings/ directory.

[there is a sample file on the sourceforge page]. This will run for 1 day only [2AM till 1AM], but the user can create a cron file to reload this file every day if he wishes to make it permanent. These targets were originally computed as the ICMP ping targets of the initial "guinea pig" customers, and represented "deep ICMP" targets, as opposed to the "near ICMP" targets used in the first mode. "Offline" events, which are missed pings are plotted.

Two additional tests are run each hour. These are a curl of the www.google.com site and a DNS query, which is targeted to the local router - a good test of the local router response [it is checked that it can indeed perform DNS lookups -via caching/forwarding].

On "offline" is computed and plotted when there is a near ICMP miss coincident with a TCP ping miss, during a 5 minute interval. "Offline" is a state which is very difficult to define, but for our purposes it represents two single minute intervals with timeouts in a 5 minute interval that correspond with misses. Depending on the timing, "instantaneous" offlines will not be detected. But then how do you define an "offline?"

The iotsnoop function on the pi4 works as follows. Each hour "tshark [terminal version of wireshark]" is launched to collect packets for this hour. At the end of the hour, tshark is stopped and restarted. Statistics are then extracted for the previous hour of IOT activity. Two plots are performed, and there is a special sub page -referred to from the main/home index page- which allows IOT monitor and interrogation. A facility is provided to extract data for a specified IOT for its last hour.

Local Raspberry pi Web Server

A web server -lighttpd- is installed on the Raspberry pi, and it serves up an index.html to a browser pointed to on the normal port 80. This index.html [refreshed every hour] presents a summary of vital configuration information, statistics and plots such as those discussed above. The stats on the web page may be delayed 1-15 minutes behind the actual info because of the quantizing of the info by the cron collection scripts. A sample of the page is presented in the appendix.

Simple http on port 80 is used, not https on port 443, so you will have to make exceptions in your browser to access the unencrypted page. This is no problem since it is on your local network.

The preferred way to view the network statistics is via browsing directly to the Raspberry pi on the local network. This address is reported in the daily email to the customer. The raspberry pi must be on the same network as the computer accessing it, in general. This is true in most cases. However, if there is an internal router on the customer's network, and the raspberry pi is behind this, the computer cannot access the raspberry pi from the main network without opening a pinhole in the internal router allowing access for port 80 from the outside to the inside. This is not unreasonable given that the internal router is “internal” i.e. not facing the Internet.

Pihole

In addition to all the internet and local network performance monitoring, the "pihole" application is also installed on the rpi [for the imonitorg app on the pi3B and 3B+ only]. It is accessed by a link on the main rpi web page. You would change your DNS server on PCs [or your entire network] to use this rpi and you can monitor the DNS activity, or you can install the blocking files using the pihole config page. Pihole as installed here does no blocking.

Additional stats collected

There are many additional stats collected about the local and IOT APN network. These include an historical arp table of devices, a table of hosts discovered, as well as services offered by the hosts. There is an "nmap" scan of the local network performed each day, and a listing of the services. There is also a wifi scan and a listing of the available networks. Speedtests are performed twice each day, and an archival plot is provided. A recording of offline time, boot times is kept. Service and host changes are flagged day over day.

A most useful feature is an archive of all the stats collected. This is referred to on the rpi web page.

How this works:

A small processor, known as a Raspberry Pi3/3B+/4B is connected via ethernet, to your network at the exit point, either at the ADSL[cable/satellite/cellular] router, or the switch which serves it. A wired connection is not necessary, but it is preferred. The pi will do all its tasks connected via wifi if you desire it. Just connect up a KVM and enter the SSID and passwd using the icons on the top there. It expects an IP address from your private network, and will thus appear on your network much like any of your devices, including accessing the Internet. Custom scripts are executed on the raspberry pi which access the Internet, and gather stats about your local network, and save the data for plotting and presentation. The individual raspberry pis are running and collecting data independent of any external network as long as they have power. You might want to consider putting them on a UPS along with your other important electronics.

The cost and scale of the effort.

There is nothing unique about this service, except that I targeted it for a small audience and using completely local equipment. For the "guinea pig" phase I am basically developing and running this service for my own "enjoyment" and interest. I am using my own servers and my own Internet service.

Mode 1 - "imonitor" - the "guinea pig" phase is a managed service, where the rpis have keys for the server and communication is possible.

Mode 2 - "imonitorg" is a non managed service. The rpis are completely standalone. Mode 2 imonitorg rpi image is released to sourceforge.

Mode 3 ' "iotsnoop" is also nonmanaged, and includes the aforementioned imonitorg scripts. A pi4 image was released to sourceforge.

The usefulness/accuracy of the information.

Because of the statistical nature of the Internet, there is really no way to measure "instantaneous" performance. We can only gauge longer term performance, which is REALLY lacking right now. This should put some quantitative measures on the "my Internet is always down" or "my Internet wasn't working this morning." This "Internet monitoring service" attempts to address this issue, especially as I/we develop the service going forward. The service can provide a record of good/fail ping responses, every minute from your network to a stable internet router, thus "showing" that your network is on/off the Internet. The archive plots and data provide a historical record.

Security and Bandwidth Considerations.

I will have a document explaining exactly what is happening with the scripts. The raspberry pi is a standard implementation using the "pi" and "root" logina for the scripts.

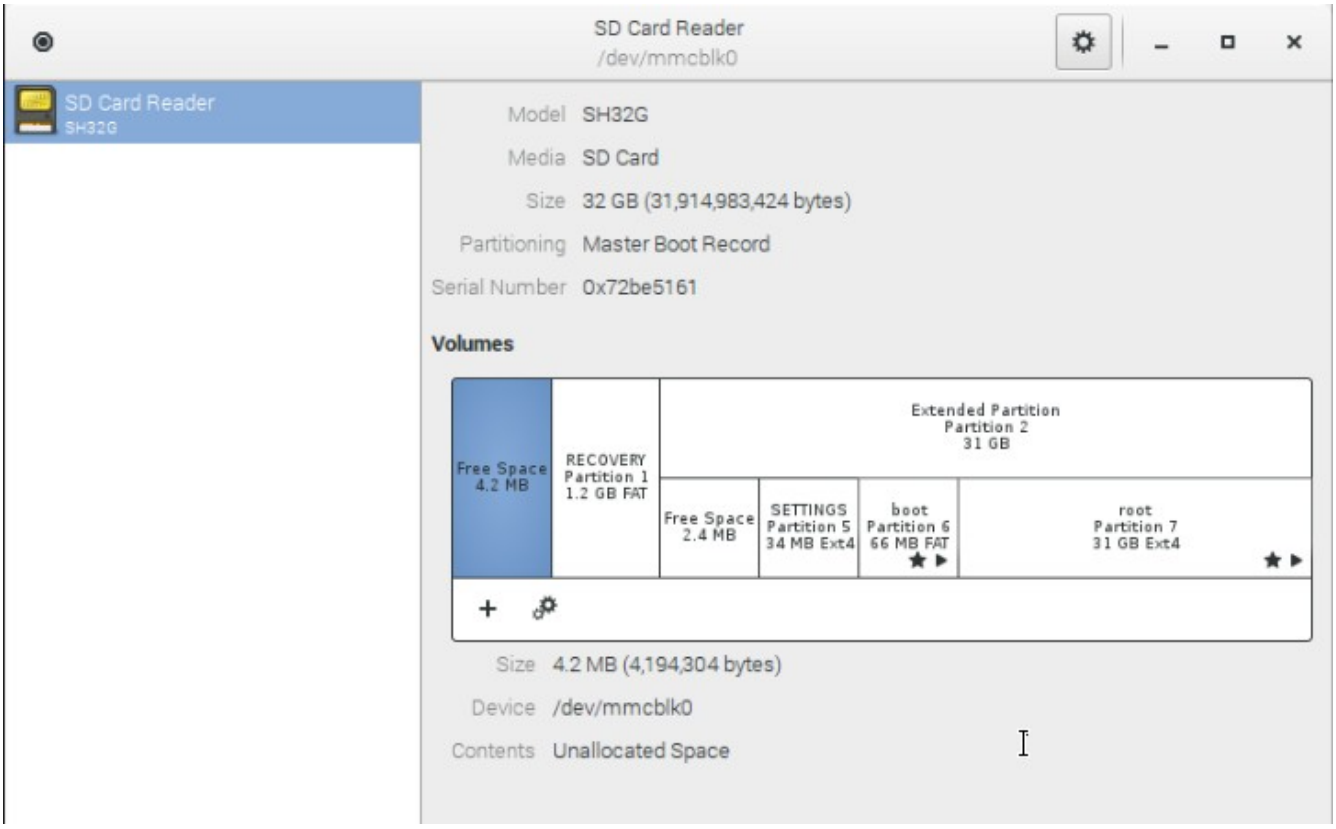
"Mode 2" -imonitorg- is not one of the infamous devices [these days] that "phones home" -- like your web connected camera that connects to a server, most likely in China, to enable it to be ID'd by an app so you can browse to your home camera by your iphone/android device. These are causing ALL sorts of problem on the Internet these days ... viz "IoT - Internet of things" - see this link just as an FYI

If you have one of these cameras on your web, and it is using UPNP especially, PLEASE give it a secure passwd. If it still has the default passwd, and it is using UPNP, you are most likely being used by Internet lowlife to perform DDOS attacks, such as the one experienced by Brian as noted here. Another Link explaining access to the cameras/IoTs.

Installation

Downloads the .gz or the .bz2 or the .xz file and uncompress it using gunzip, or bunzip2 or unxz on linux. This will result in an .img file of 32/64GB. Use a tool such as balenaetcher on linux, or win32 disk imager on windows**.** [There is undoubtedly a way to do it on MAC, but I haven't researched that yet].

If you are using linux strictly [or MAC?], just use "dd." "dd if=rpi_image of=/dev/YourDisk bs=4MB status=progress" will write the image to your SD card held in a USB caddy. You must find the disk ID using fdisk e.g., or the linux disk tools. Typically it may be /dev/sdc if you already have sda and sdb disks. The image supplied is a noobs encapsulated which looks like this: There is thus a recovery partition containing the original raspian. [This image shows a 16GB image expanded to 32GB. You can do this using gparted if you want].



Place the microSD in a 3B or 3B+ [or 4B for iotsnoop] raspberry pi. It should boot as a normal raspberry pi installation. You should find the address of the raspberrry pi in your router, and then browse to it on port 80. This is your main interface to the imonitorg/iotsnoop. Use the "QuickManual" linked at the top for additional instructions for setup. You can setup the google email. You will need 2FA authentication for your google email, and obtain a key from your google account [instructions are referenced on the rpi web page]. There is usually nothing else you need to setup, as it will come up running out of the box.

IMPORTANT: You need to wait at least overnight for the raspberry pi to "figure out where it is/what it is" and setup some parameters. Local values such as the rpi IP address may not be correct for a day, and there may be some leftover bogus data from the imaged OS.

All scripts run from pi or root user, mostly using a cron file for them. Do not disturb the scripts or the cron file. Otherwise you can run as normal user pi [default user/password]. Best to setup your own user or folder if you want to do additional work on the pi.