IOTsnoop Quick Guide [3-16-2025]

Installing the iotsnoop on your pi4 32bit raspberry pi will give you wide-ranging and near realtime monitoring of your local network and Internet performance, via a webpage available on the pi4 365/24/7. The "iotsnoop" function added to the original "imonitorg" functions will perform monitoring of IOT devices connected via a wifi APN on the wlan0 -wifi interface using both plots and statistics.

https://youtu.be/v-NOPoMh860 youtube intro to imonitorg and iotsnoop

Iotsnoop will "come up running" and it is only necessary to set the time zone via raspi-config, and the email config via the iotsnoop web page, if you desire a daily email. It uses the "pi" user to run most of the scripts. There are some root scripts as well. The "iotsnoop/vbox/imonitorg" scripts were originally developed on raspberry pi 3B and 3B+, and the scripts are transferred/altered to run on a 32bit raspberry pi4. Pihole is NOT included in this image, nor is the wifi monitoring, since the wifi is converted to an APN for the iotsnoop. The "iotsnoop" variant must be attached to the ethernet on the WAN side of your network [ideally the router, or a switch downstream], and the pi4's wifi is put into "APN" mode to allow clients to connect via wifi normal authentication. It is "intended" to wifi-authorize [login to iotsnoopg SSID APN] home IOT devices here [which are mostly on wifi] to shift them to this "guest" network, off of your main network. Scripts are then run to capture all the IOT packets and interrrogate them and derive plots and statistics which are displayed much as the imonitorg statistics of former implementations.

Early versions of iotsnoop running on 1GB pi4s may run into limitations for heavy IOT traffic. Running steaming IOT such as TVs is not recommended on 1GB pi4s. A 4GB pi4 iotsnoop is being evaluated and appears largely up to the task, even with HD streaming sources. 4K stream sources will saturate the capture at about 30 min of the allotted 60 min capture with the default 2GB tshark buffer. An 8GB 32bit pi4 is ideal for running iotsnoop. pi4 Images are provided for on sourceforge.net.

1. Who should use an iotsnoop

Residential or small business Internet customers

Any person NAT'd to the Internet who wants to monitor and gauge their Internet performance over long terms should be able to use this, much as the "imonitorg" images were used. The iotsnoop variant allows wifi clients to connect to the pi4 wifi SSID "iotsnoopg" to allow iOT monitoring as well.

2. Setup and Access the iotsnoop

The iotsnoop pi4 must be connected via ethernet to your router, or a switch downstream, since the wifi is set in APN mode and dedicated for "iotsnooping." Iotsnoop should not be connected directly to the Internet, as there is no firewall running. The iotsnoop expects an IP address, gateway and DNS, just like any device that you add to your home network -it needs no initialization. You should ideally assign this to be static via your DHCP server [normal dynamic is OK, but the stats will appear bogus and need "rejiggering" once the scripts work their way "thru" the new IP assignment]. Once you have the IP address, just browse to the iotsnoop pi4 - this will be http only, not https, so you may have to make exceptions in the browser to allow the http. Once you have the web page, you can setup various parameters, tho it will work out of the box. The most important thing you can setup is the TZ, and the email configuration. This will use gmail.com as a relay [you can send to the gmail address as well]. Of course you need a gmail account to do this, and you need to set up 2FA on gmail. [Instructions are referenced on the iotsnoop webpage] You don't need to add the email capability, of course.

The iotsnoop login is pi and the password is "raspberry." [this is a normal OOB raspberry pi4 in all respects] You will definitely want to set the TZ if you are not on the US East Time. You can do this in the normal way via raspi-config.

It is important to realize that the goal of the [original] imonitorg scripts is longer term performance monitoring. You will have to wait for at least a day/overnite for some imonitorg stats to accumulate. This is not an application like "Ping plotter" which gives a much nearer term performance measurement. Unlike the imonitorg stat collection, the "iotsnoop" stats are collected hourly, analyzed, and displayed in a plot and iframes -thus quantized to the hour. The main IOT plot is a rolling display of 250 calculations, once per hour, so it extends of 10+ days.

3. Setup iotsnoop interface

The "iotsnoop/imonitorg" was originally an image created for a generic raspberry pi 3B or 3B+. The "iotsnoop" is an addition for the pi4 variant. Many scripts have been added to effect the network monitoring, now including the wifi APN "iotsnoopg" SSID on the wifi - these will run automatically on boot, once it is able to see the Internet and synchronize ntp clock service. You can login to the iotsnoop using the pi login, either via a desktop KVM or via ssh to the user pi from another computer. Do not modify the pi or root scripts or cron files. Vnc should be avoided for longer term, because it steals significant resources from the pi and the intense process for packet capture. Command line access [ssh] is much better for longer term access.

BEWARE: The desktop display manager, and "pulseaudio" are disabled by default in the pi4 iotsnoop image to enhance performance overall. To turn this on, if you are KVM, as pi "systemctl start lightdm" and then "startx."

4. Configure iotsnoop options from the webpage:

- Email setup

On this generic iotsnoop, it is possible to receive a daily status email by using the relay capability of your gmail for user accounts. Your gmail account must be set for 2FA, and you must obtain a 16 byte key to use as the authorization string,

Set gmail relay [where you can receive the daily email status] on the iotsnoop "management" web page:

gmail relay enable state - YES or NO

gmail account - gmail account "username@gmail.com"

gmail authorization - gmail app password -16 byte key obtained on gmail site

ultimate email - ultimate destination email address[es]

See this link for e.g. instructions: https://johnloop.com/imonitor/gmailRelayInstructions.txt.

- Set Cloud Management Options

There is no cloud management for the iotsnoop. It is completely standalone. The original raspberry pi3B[+] "imonitors" used a cloud server for monitoring/update as the scripts were developed. 25+ users across the US with different access technologies and ISPs were employed. It covered cable/fiber/ADSL/WADSL/Satellite/StarLink and home networks from exceedingly simple to dozens of home network gadgets. A generic "imonitorg" and a VM implementation [vbox] was released later which has no connection to any server.

A user may login as "pi" and perform operations, but he must be careful. All monitoring scripts run as user pi or as user root. He can create a new user and use that login for personal projects. The easiest "monitoring" is just a browser open on your main PC pointed to the iotsnoop.

Optimally, the iotsnoop pi4 is "left alone" for its imonitorg/iotsnoop functions, and the user accesses iotsnoop via browser on a PC pointed to the pi4 iotsnoop.

- Legacy imonitorg functions:

--none of these need to be set, but they are customizable for certain situations.

- Configure “near” ICMP Ping target, and "LOCK" or "AUTO"

Ping target is temporarily settable, but it is intended to operate in automatic mode, and reverts to this at the next hour unless the "LOCK" option is invoked. A "LOCK" option on the ICMP ping target is available to override this, settable on the main rpi web page. It is fixed until manually returned to "AUTO. " "AUTO" is the default operation. A LOCK is occasionally necessary with ISP rearrangements/configuration changes.

The timeout value is fixed at 1000 msec. AUTO mode configures the ping target -each hour- as the closest pingable hop starting at hop [5,4,3,2 -configurable via the web page with 5 being default] and counting down. It uses **IP address only** to forego use of DNS. Thus it uses a "near" IP target to gauge local ISP performance, independent of DNS and Internet-at-large. ICMP echo request packets [UDP only] are used. The timeout is set to 1 sec default. This near ICMP ping accumulates response times and will will probe for "prolonged" outage once a ping is missed [these are referred to as "hyperpings."]. A a " burst block" of ICMP request packets are sent each minute, typically 4, and a single miss registers as a failure.

UPDATE: the latest versions actually send a settable "block" of ICMP queries, 4-12, and a single missed ping within the block is designated a "fail." This provides better testing of the high bandwidth connections such as cable and fiber.

As mentioned, it is possible to set the hop start used for the ping determination to further allow customization. This can be 5,4,3, or 2 or 1. Sometimes the AUTO algorithm settles on a ping target[s] that can become very unreliable. We need to find a reliable near ICMP ping target. This can be adjusted once the ping behavior is noted after a few days. Deviation from "reliable" is then one of the performance monitoring goals of the imonitorg. If a reliable ping target cannot be obtained from the ISP -some vary the target, and even block pings, then the LOCK can be exercised, and a reliable ping target such as "1.1.1.1" can be configured. "1.1.1.1 [cloudflare mirror]" is likely pretty close to everybody!

- “Deep” ICMP ping configuration

There is no “deep” ICMP target for the imonitorg or the iotsnoop "out of the box." For the imonitor [development] mode, these targets were the collection of the near ICMP targets of the other customers, automatically gathered and distributed to give a feel for "the Internet at large."

However, it is possible to place a file "CustPingSites.newtxt" in the /home/pi/tests/customerpings/ directory, and the pi will perform deep ICMP ping for one day starting at 2AM and ending at 1AM. This file should contain an IP address on each line. If you want the ping to be repeated, make a cron file to reposition the file each day at 1:30 AM [it is renamed in order to stop it after a day]. This will show up on the "real time" and the daily plots as a third line plot.

- Configure TCP target, and "LOCK" or "AUTO"

A TCP "connect" is performed to a "foundation" target to determine online/offline status each minute, and then a TCP "ping" is performed to a round robin of "100 top web sites" to determine response times each minute. The Internet FQDN DNS is used in order to invoke the DNS query mechanism. "www.google.com" or "www.msftncsi.com" is suggested for the "foundation" target. A "LOCK" or "AUTO" may also be applied to this ping. The "AUTO" option uses the top 100 Web sites as TCP ping targets [as of spring 2020]. The "LOCK" option uses the foundation target as the TCP ping target. The "foundation" TCP connect target remains fixed even with "AUTO" option. It uses a “TCP” target to better gauge larger Internet performance. It uses TCP half open exchange, sending a SYN, and if a SYN/ACK is received, returns a FIN/ACK. The TCP Ping is only attempted while online [Internet accessible]. Care should be exercised in interpreting the TCP ping data. The top 100 web sites are largely implemented on local CDN sites, and are often very close!

As of 5-2023 there is a variable to set to enable/disable TCP ping plotting. Often, because the tcp pings are to the CDN nearest, there is little utility in the plots. The "anomaly" TCP timeout is always plotted which gives more info. Just assume that TCP pings are "normal" unless you see the TCP timeout plotted in the anomaly section. You can, as root, "echo [YES|NO] > /root/DEFtcpplot.txt" to enable/disable this capability. It is only updated on a reboot [monthly unless forced].

As of spring 2022, the top100 web site listing uses "naked" domains. Occasionally, it is necessary to convert the "naked" domain, such as "google.com" to a valid FQDN such as "www.google.com." This occurs when a domain subscribes to a CDN, which requires a valid domain for CNAME. You can actually edit the /root/Top100WebSites.txt file to correct this problem. I have had to do this once. You can see when this problem occurs because the tcp ping timeout will appear periodic in the plot.   The web page has an iframe where the TCP ping results are displayed, and you can spot this.

The /root/Top100WebSites.txt file is for the US. You can modify this file, or reconstruct it in the same format for different countries [you can eliminate the third column]. It must contain exactly 100 lines.

"AUTO" is the default setting.

- Configure curl of URL

Twice an hour, a “curl” of a configurable URL is performed, and the response time recorded. A “curl” is just a cmd line way of “browsing” - pulling down a web page. This response time is plotted on the daily plot or the "realtime" plot with the three other measurements. This scan represents the “highest level [from a protocol level]” scan performed. Only the index page is downloaded, not the images and links referenced in the index page. As a second DNS performance measurement, the DNS query response time for www.google.com is recorded and plotted.   This is normally done to your router, as long as it caches or proxies DNS requests. The raspberry pi checks this, and reverts to 8.8.8.8 if the router does not perform caching/proxying.

ISPs will sometimes force the router to use their own DNS, such as comcast often does with 75.75.75.75. It is not possible for the router to override this -usually. The DNS server used for the DNS query is noted in the daily plot. Most often it is your own router.

The iotsnoop configuration assigns 9.9.9.9 as DNS for IOT devices. This is settable on the rpi web page, but should be left. 9.9.9.9 provides automatic denial for malicious websites. See quad9.com for information. Of course, the pi4 iotsnoop assigns itself [192.168.50.10] as the DNS to its IOT clients.

- Enable *Internal* scan of your network [default OFF, ON]

This is a scan conducted by the iotsnoop on the connected main home private network [not the iotsnoop wifi network]. It can be especially important if your host's OS is running wifi in a wifi maze of networks. It scans for vulnerabilities on your network hosts using linux "nmap." It discovers, lists all hosts and changes day/day on your internal network via IP address. There is a listing, including a historical listing of hosts/ports in “Data snapshots,” and a complete report in “counts.” These stats are available on the rpi webpage. There is an algorithm which spans a week to detect "NEW [never been on your network before]," "GONE [it was previously on your network, but disappeared. This lasts for two scans", and "persistent [this host has been on your network 10 or the last 12 scans]." This algorithm normally runs twice per day, at 6PM and 1AM [busy and not busy]. It is useful to detect transient hosts and permanent hosts. This is actually configurable by changing the "findhosts" schedules in the pi crontab.

Remember, scanning is not performed on the wifi APN iotsnoop network. We do packet capture and analysis there.....

- IOTsnoop functions:

- Iotsnoop configuration and use

As of 10-2024, there is no iotsnoop configuration other than the specifics which are detailed on the iotsnoop sub sections. Six main statistics are collected and plotted/listed. These are 1) total unique DNS queries, 2) total packet count, 3) total unique src-dst IP pairs, 4) UDP packet count [less NTP, DNS and ICMP], 5) TCP session count, and 6) a derived bit/s count averaged over the last hour. These are plotted on a rolling graph in the main web page - currently at 250 entries. The individual listings of the DNS queries and URL visits are in a separate page. This collection is done once per hour by stopping tshark [wireshark cmd line] and restarting it while taking the pcap file and interrogating it.

There are several archives maintained, including a history of DNS queries, a history of the hourly stats going back about 10 days, and a "script performance" file showing the time to perform the packet analysis.

Interrogation of the iotsnoop host stats and packet counts and monitoring the plot can give an indication of IOT activity. Several "alerts" are generated if tshark crashes, or IOT activity is 5X the last hour, or the tshark 2GB buffer is exceeded. These are reported in the alerts panel on the iotsnoop web page, and email, via the "/home/pi/tests/mail1" file, reported each day via email [if enabled].

The interrogation of these packets for meaningful information is an ongoing task! Recently added was the ability to specify the IOT host and list its DNS and URL visits last hour and archive queries. Convenient links are displayed to copy and paste questionable IOT DNS/URL visit queries into web identifying sites. This includes virustotal.com, especially useful for malicious sites.

- Addional configuration options

Route target for hop delineation. This is a far end location such as 222.1.1.1 [Japan], or 90.1.1.1 [France] to control direction of the packets many hops from your router. This may not have an effect if the same 5 routers are encountered in your path for any destination. Other customers are but a few hops away from the backbone where it will make a difference.

- IOT plots

The IOT plot is below the initial -original- imonitorg performance monitoring plot. The IOT plot consists of two rolling displays, each lasting 5 seconds.

These two plots display IOT performance and statistics which are rotated on the base -index- page. The first plot show a rolling display for 250 hours (10+ days), which shows bandwidth used in Mb/s, number of packets, host pairs, UDP sessions, TCP sessions and DNS queries. This plot is ***updated every hour*** once the packet capture is interrogated for IP statistics. Only IP packets are captured and analyzed, not the layer 1/2 ethernet frames. This plot is meant as a long term indication of the IOT APN performance.

The second plot is a rolling display, ***updated every minute*** for 250 minutes (1day+) which shows the rcv and tmt bandwidth, in MB/sec. The upper boundary is the approximate bandwidth limit of the

802.11g 2.4 GHz channel -200MB/min or about 35 Mb/s. Exceeding this will likely overrun the packet capture buffer (tho it should not crash the actual router function). This traffic will include any layer 2 or 3 traffic, not just IP packets.

- IOT analysis section

In this section, you are able to select an IP address which is used to interrogate the packet capture of the last hour. Normally you would reference the DHCP lease map and use one of these addresses, but there are occasional rogue -non DHCP assigned- addresses used by the IOT. All this last hour data is shown in the panel where the hosts, packets, name are shown for the last hour. The IOT contact map can be selected for this analysis, tho it defaults to OFF. You can enable this by using the submit window to change it to YES.

- IOT URL visit map

By selecting the IOT IP from the dhcp leases [or a "rogue" IP!] text [using the form box and the submit], you can select the "Last hour" and "archive" of the IOT target DNS queries and URL visits. The map function is also triggered and the map displays the last hour locations of the URL visits of the IOT. Beware that many of these are CDNs, Amazon servers, Cloudfront servers, etc, so the utility may not be that useful. The map can be very useful to reveal rogue servers in different countries. This URL visit list should be used with an AI such as chatgpt or grok -simply submit the list and ask for an analysis!

Note that the LAST map "submitted" is always displayed - it is not recalculated until the "submit" is selected for another/same IP address.   The map is defaultly OFF.

5. Email advantages/summary

There is a daily status email available by using your gmail account as a relay [see above]. It includes links to the daily plots [accessible on your local network via the pi4 http server], and a list of numerous network statistics. The original development imonitorg provided the server as a mail relay so additional tests could be performed such as a port probe test. "Alerts" were also configured reporting on on/offline/DNS fails/power cycles. These alerts are not possible with the iotsnoop generic, but they do show up on the iotsnoop web page and in the email.

6. Some advantages of using the iotsnoop webpage

The iotsnoop webpage is implemented on your local network only, and is completely safe. There is no cloud server with all your data that you have to access! It gives all plots/stats reported in the email [comparing it to the “imonitorg”], and is the way to configure your iotsnoop.

Perhaps the neatest "imonitorg thing" is that you can display a “real time” plot – it starts at 2AM and ends when you request the plot. It is linked on a separate page called by this script. You must wait a minute for it [as of the iotsnoop implementation, this plot shows up on the main page, rotating with the speedtest archive plot]. You can also select a "ping blast" plot, where 60 pings are delivered to the ICMP ping target, and then plotted. You can also trigger plots at any time, and specify the start of the "real time" plot so that you can narrow in on problems. In addition there is a "hyperping" capability which triggers a more intense ping of 6/sec when the 1/min ICMP pings time out. If these "hyperpings" timeout, they will designated by "ticks" on the plot ascending vertically above the minute they were launched. This gives a much greater feel about the network availability around the 1/min ICMP [block] ping failure.

To "preserve" the plot, simply click on the link indicated to steer it to a new tab. It will remain there. These "imonitorg" features were carried over en masse from "imonitorg."

There is no upload to a webserver in the Internet cloud to collect info [many similar services use a cloud account and require you to have an account/login to see the info for your own local network]

The webpage is accessible even if the Internet is down, as long as your local network is OK.

The iotsnoop [like imonitorg] also provides identical near-realtime plotting via a simple link. This is a link at the top of the main iotsnoop webpage. Clicking this link will open a new tab on your browser which will be updated every 1 minute with the current plot.   You can still do a custom plot from the main iotsnoop webpage, which is linked from this page [click it and refresh to preserve the plot, otherwise it will be overwritten with the next auto update].

The iotsnoop plot is displayed directly below the imonitorg plot image, and is updated every hour. It is thus a rolling image of [currently 250 hours]. It cycles between a normal and a log plot on the y axis for better visibility as to activity.

UPDATE 3-20-2024: The imonitorg "near realtime" and iotsnoop 10 day plots are now displayed in real time on the initial page........   The previous section 6 paragraphs still apply however.

The webpage is not loaded with widgets and gadgets, with background loading of dozens [hundreds?] of websites and cookie URLs. It is a very simple implementation showing stats and enabling configuration. There is only one figure included [to be added] and no scripting other than for alerts, and to allow you to set iotsnoop parameters. The "alerts" were originally configured for the imonitor [managed version] and generated a file to be discovered by the management server and emailed to the user. There is no such facility on the iotsnoop, but the alerts "accumulate" in the "mail1" file which is visible on the iotsnoop or via the daily mail if setup, and iotsnoop alerts also appear on the iotsnoop web page.

The iotsnoop webpage will show dated ping times, plus giving a link to the plots. Plus:

1 year archive of speedtests, services and port changes on your network, plus the daily emails.

1 year archive [plus plot] of Internet offline dates and times.

Many more statistics about your network and performance will be noticed. .

7. Advantages of mgmt server iotsnoop.com[imonitorg.com]

There is absolutely no contact with this server for the iotsnoop function, tho it serves as a knowledge base and reference for the project.   The server was used in the development phase of imonitorg, where 25+ users/friends across the US allowed me to test these scripts on varying situations/homes/networks/ISPs using a dedicated raspberry pi 3B or 3B+.   Many ISPs were used, notably comcast, spectrum, ATT, Starlink, windstream, municipal ISPs, fiber, cable, sat, WADSL.

This implementation was developed independently using 3 separate instances [pi3B, 3B+ and ova]. The base scripts are identical to those implemented on the trial imonitor/imonitorg.

8. Some advantages of using the imonitorg/iotsnoop on your network.

The imonitorg ping plots are particularly useful for quick comparisons of your network longer term performance. You can quickly scroll thru days of performance in the archive window on the iotsnoop webpage. The same goes for many other plots and stats.

The iotsnoop plots and stats will give great insight into the activity of your IOT network.

Archiving of all data, emails and plots, boot times, IP address records, for at least for 1 year.

Daily scan of your home network for open ports and services [if scan enabled]

The ping tests will probe into Internet.

Listing of your local network hosts and svcs, plus notification of changes [detected at 7PM].

DNS tests - DNS change alerts.

Browser tab showing near-realtime performance plot.

Ipv6 connectivity is indicated. The existence of an ipv6 default route is first checked, and even if the pi is unable to do an ipv6 ping of www.google.com, ipv6 is declaed OK. A "roofline" is drawn on the plot if ipv6 fails, meaning there is no ipv6 default route. Please note that ***IPv6 is disabled for the iotsnoopg APN!*** The ipv6 indication on the plot is for the WAN side of the iotsnoop, into your Internet access.

The use of the router as the ping target is also noted with a "roofline" on the plot, since this tends to limit the characterization and performance metric for the internet access. Sometimes the near ICMP ping target is AUTO selected as the router, and this is indicated thus. You can also LOCK the near ICMP target to your router and this is flagged by the "roofline."

Twice-daily speedtest measurements using the ookla speedtest CLI. Archive plots will record this performance over long term. As usual, caution is advised in interpreting the plots. They may vary tremendously, but consistency is important to achieve.

IOT monitoring is added with the iotsnoop. You can see total IOT packet count, host pair count, and DNS query count, along with a list of the DNS queries and the IOT hosts. The plots give overall indication of long term activity.

The iotsnoop wifi SSID APN "iotsnoopg" defaults to 2.4GHz. The 2.4GHz wifi is better for further distances, and the lower speeds are amenable to IOT use. --WHY would you want your IOT stealing your bandwidth! In addition, the wifi transmitter on the pi4 is much less powerful than on a router/wifi access point, thus 2.4GHz makes more sense. For a large, scattered IOT network, you should probably use a centrally located wifi extender, such as the Netgear https://netgear.com/extenderhelp

UPDATE 10-10-2024: it is possible to switch the wifi SSID to 5Gz by "sudo su -" and going to /etc/hostapd and copying the 5GHz conf file to hostapd.conf and rebooting.

9. Definition variables [settable via sudo to root]

It is rarely necessary to change these. This is for the record only.

These variables are located in /root and are set by sudo su – from the pi login. These can be changed by the imonitorg/iotsnoop customer as indicated below, either via directly setting them via an ssh session, or via the rpi web page. They are not normally accessed except in special circumstances. Most configurable parameters are accessible on the imonitorg/iotsnoop web page.

1. DEFpi.txt -this is set to "iotsnoop" for iotsnoop

2. DEFpingtarget.txt -this defines the initial near ICMP ping target, re-auto determined every hour unless it is locked via DEFpinglock.txt. CommToServer picks it up from localDEFpingtarget.txt every hour. Settable on the web page.

3. DEFconnecttarget.txt - this defines the connection target, currently “www.google.com”; “www.msftncsi.com” is another good one - used by Microsoft to determine network connectivity.

4. DEFcgnat.txt YES for CGNAT ISP, NO for regular ISP [Internet IP is non distinctive]. This must be specified, otherwise Internet IP may wobble and alert continuously.

5. DEFmgmtserver.txt – set to imonitorg.com so the iotsnoop can display the latest iotsnoop version once/month. --not configured yet, 10-2024

6. DEFinterval.txt - not used

7. DEFmgmt.txt - this is not used by iotsnoop/imonitorg.

8. DEFdnspull.txt – this defines where to get my IP address from. Currently “myip.opendns.com” This is hardcoded in the script. "o-o.myaddr.l.google.com" is also used. **Should NOT be changed**

9. DEFcustemail.txt -this is not used by the iotsnoop/imonitorg. [different from gmail address]

10. DEFmgmtemail.txt -this is not used by the iotsnoop/imonitorg.

11. DEFtimeoutinterval.txt - This defines the ping timeout interval, in msec. Currently “1000" is default for ping and is not changeable.

12. DEFrealrouter.txt -this file exists only if the iotsnoop/imonitorg host OS is behind an internal router, and is the internal IP address of the ***Internet*** router. Can test for existence of this file to test if behind internal router. **The customer must set this variable. NOT used by iotsnoop scripts. The pi4 APN is always NAT'd behind the router connected to the ISP. It is a "double NAT."**

13. DEFixc.txt -traceroute destination 222.1.1.1 points to Japan, 90.1.1.1 points to France. It is really intended as a route eventual destination for "traceroute" even tho only the first hops are used.

14. DEFtemp1sensor.txt -not used by iotsnoop/imonitorg

15. DEFtemp1downlimit.txt -not used by iotsnoop/imonitorg

16. DEFtemp1uplimit.txt -not used by iotsnoop/imonitorg

The DEFtemp variables are not used for iotsnoop/imonitorg. The original raspberry pi could add temp thermometers and samples were taken and plotted.

17. DEFemailforIPchange.txt – this is not used for the iotsnoop/imonitorg.

18. DEFsa.txt – this is not used by the iotsnoop/imonitorg. It is set to ON, “standalone.”

19. DEFconfig.txt -not used by iotsnoop/imonitorg

20. DEFgmailENABLE.txt - enable email relay via your gmail account. Default NO

21. DEFgmailUSER.txt - gmail user account to authenticate to for relay

22. DEFgmailAUTH.txt - gmail app password used with DEFgmailUSER.txt

23. DEFgmailULTIMATE.txt - ultimate email destination via gmail relay

The DEFgmail*.txt variables can be set from the web page.

24. DEFplotmap.txt Default NO. Set this manually [no GUI way to set] YES|NO -tells the plotting script to gather map coordinates for the IOT URL visits, and plot them. This takes a LONG time for many URL entries, which can amount to hundreds. The method of acquiring the map coordinates is to query ip-api.com which only allows maybe 10 queries per query. So to hasten the info, we turn off the map generation.

25. DEFpinglock.txt AUTO or LOCK; "LOCK" locks value, "AUTO" lets script determine it automatically each hour for the next hour. This can be set on the rpi web page.

26. DEFhop.txt 5,4,3,2,1 This represents the hop to start at to check for near ICMP ping target. it is decremented until "1" to find a pingable near target. Otherwise it is set to 1.1.1.1

27. DEFscan.txt OFF or ON --internal nmap scan of customer network; default OFF. This can be set on the imonitorg/iotsnoop web page. Performed once each day.

28. DEFdailyemail.txt YES or NO or OFF – this is not used by iotsnoop/imonitorg. It is set to OFF. It was a separate mechanism to deliver mail in the development phase. Separate from the gmail relay capability. This is different from the gmail enable setting.

29. DEFtcppinglock.txt LOCK or AUTO. LOCK means use the value entered in the form. AUTO means use the top 100 web site list [as of spring 2020]. The value entered in the form is always used for the connect foundation "test."   If you choose AUTO, it is only for the TCP foundation ping testing. Note that choosing “AUTO for the top 100 web, these will normally be sites on the nearest Content Distribution network [CDN].

30. DEFcreateRTplot.txt – set by entering the start time, and pushing the “PLOTIT” button on the web page to create current plot. This is an imonitorg function. Cleared by script that does the plotting. This variable is set via the web page and checked every minute by CommToServer. The plot will show the plot from the designated time to the time it is invoked.

31 DEFcreateMINplot.txt - set by pushing the ping blast "PLOTIT" button on the web page to create ping blast plot. 60 pings in a minute to ICMP ping target.

32. DEFcustomerping.txt –   this is set to NO for the iotsnoop/imonitorg.   Customer may still implement customer pings as noted elsewhere.

33. DEFversion.txt - set at image version creation. Simple date. Each execution of "/root/clean_piG_files_generic" is recorded here to signify "reset to factory default:."

34. DEFbootcurl.txt - not used by iotsnoop/imonitorg

36. DEFpiversion.txt -not used by iotsnoop/imonitorg

37. DEFimonitorversion.txt date, such as 07/22/2020

38. DEFcurltarget.txt -enter a URL for a favorite web site. www,google.com is default. This is used for the twice-hourly curl test.

39. DEFpasswd.txt must be set to the pi user passwd. Ping is then able to run sudo and send back-to-back pings for better stress testing.

40. DEFtcpplot.txt enter YES or NO to enable tcp ping plotting. Default is YES for generic imonitorg.

41

42.

41 and 42 should not be changed in the initial firmware because they mayinterfere with plotting.

43. DEFhostholdback.txt Set to 10. Allows hosts to disappear for 2/13 scans to still be considered "permanent" hosts. Settable via root.

44. DEFiothostholdback.txt Set to 3. Allows iOT hosts to disappear for 3 of 12 scans since they are liable to be phones, tablets and may move. Settable via root.

45 DEFtsharkBuffer.txt Set to 2GB [2,000,000,000 bytes]. Setting of this variable is under study. It is used directly as the "-b" option in tshark.   A 4GB pi4 is recommended as a minimum for an IOT network of maybe 20 gadgets. 8GB pi4 should definitely be considered for larger IOT networks. You should probably not modify this value! As of 4-1-2025, this variable has no effect.

46. DEFpingburstinterval.txt Set to 0 to allow pi to send back to back pings [well, almost]. The DEFpasswd.txt is necessary because the ping cmd needs sudo to do this burst in later pis.

47. DEFpingburstcount.txt The ping burst count at the start of every minute. Set to 4. Can be set up to 12 without messing timing up. A single missed ping within the burst counts as a failure for the ping script.

48. DEFfindiothosts.txt Global iotsnoop setting. This is YES to enable IOT monitoring. Setting this to NO essentially turns the pi4 into an "imonitorg" only function, much as the 3B[+] pis are doing. The IOT stats and plots are NOT collected, tho the plots and stats skeletons are still visible.

49. DEFsnoopsource.txt -used by iotsnoop DNS listings

50. DEFiotdns.txt -settable DNS for IOT devices for dnsmasq, set to 9.9.9.9 initially

51. DEFtshark.txt -future use, MAX [current], MIN, MED

52. DEFhomessid.txt -APN SSID default iotsnoopg

53. DEFhomekey.txt -APN KEY default iotsnoop

10. Special Considerations

On a boot/reboot, for a raspberry pi, the system clock may be wildly off until the Internet is acquired and ntp can sync the clock. If possible, the rpi4 and the modem/router should use a UPC for reliable stats. The ping target is recalculated each hour [in AUTO mode], so restarting the iotsnoop/imonitorg at any other time will use the ping target of the previous hour.

When the pi is powered off for a length of time, and is rebooted and does not have access to the Internet to acquire time via ntp, the clock typically resumes from where it left off. Thus, without Internet, the time reported by the pi will be slower by the amount of the power off. So without Internet and ntp, the mapping, statistics will be very far off.

There is a special script "clean_iotsnoop_files" in /root which you can use to clean the iotsnoop for re initiation. Beware, it wipes all iotsnoop/imonitorg history.   Plots and stats are reset to zero, like OOB.

The iotsnoop/imonitorg will operate with any IP address it is given via DHCP, but the scripts may need an overnite to update all the imonitorg webpage links when the local IP changes. An entire day may be needed in some instances. Assign a static IP to your iotsnoop to minimize these effects. To determine hosts, a week will be needed to work the algorithm. IOT hosts are given a 1 hour lease on the IP via DHCP.   **Beware that Leases may have expired/be unused but still reserved and displayed on stats.**

For the iotsnoop function, the intensity of the tshark packet collection may sometimes drive the pi into "out of memory" situations. Tshark will be terminated by the OS using "oom-reaper." It should restart automatically at the next hour. "alerts" are placed in the mail file [delivered daily] to signify these events.   ***This is an ongoing area of development, tho the latest iteration using the ring buffer "-b" tshark option seems to be more stable.***

UPPDATE 4-20-2024 tshark MAX with 4GB RAM seems to work reliably for bandwidth up to about 6-8 Mb/s average over the hour, even doing HD streaming [using a few Mb/s bandwidth]. a single 4K streaming device [using maybe 10Mb/s bandwidth] will fill the 2GB buffer in about 30 min. 4K streaming and streaming devices in general should not be placed on the iotsnoop APN. Investigations are ongoing to MIN and MED tshark capture capabilities to allow more "shallow" but complete traffic capture and characterization.

UPDATE 10-10-2024 pi4 with 1GB RAM has been working reliably on an IOT network without streaming with about 29 devices. The bandwidth rarely exceeds 1Mb/sec average over an hour. This still works even using the default 2GB buffer, tho I should probably change this to 500MB or so. Ongoing....

Please remember that the overriding guide was for longer term measurements of stats and performance, not minute to minute or even hour to hour performance. You will encounter web page instances where the data is not "real time." In addition, the IOT APN is purposely "throttled down" to 50Mbs [802.11agn] and ipv6 is not implemented.

CAUTION: Occasionally, the complete shutdown of the wifi APN has been experienced, tho all "systemctl" cmds indicate correct operation. It is an ongoing project to monitor for this situation, and at the very least alert -please reboot- or do an auto reboot. This is ongoing investigation. The failure mechanism is not known.

11. For Information: https://iotsnoop.com

John D. Loop pccitizen@gmail.com or jdloop@johnloop.com

Below are sample combined plots. Individual ICMP and TCP ping plots are also available via the iotsnoop/imonitorg web page. It shows the values collected during the day, and the iotsnoop/imonitorg going offline at 6:00PM till 1AM the following day. The scripts and presentation are the sole property of John Loop.

12. Converting to a "managed" iotsnoop/imonitorg

It is possible to convert the iotsnoop into a managed iotsnoop via generating a key pair and sending me the public key, plus setting some of the DEF* variables above. ***It is not planned to do this for the iotsnoop/imonitorg at present***, tho the underlying scripts are all there. Please contact me for info.

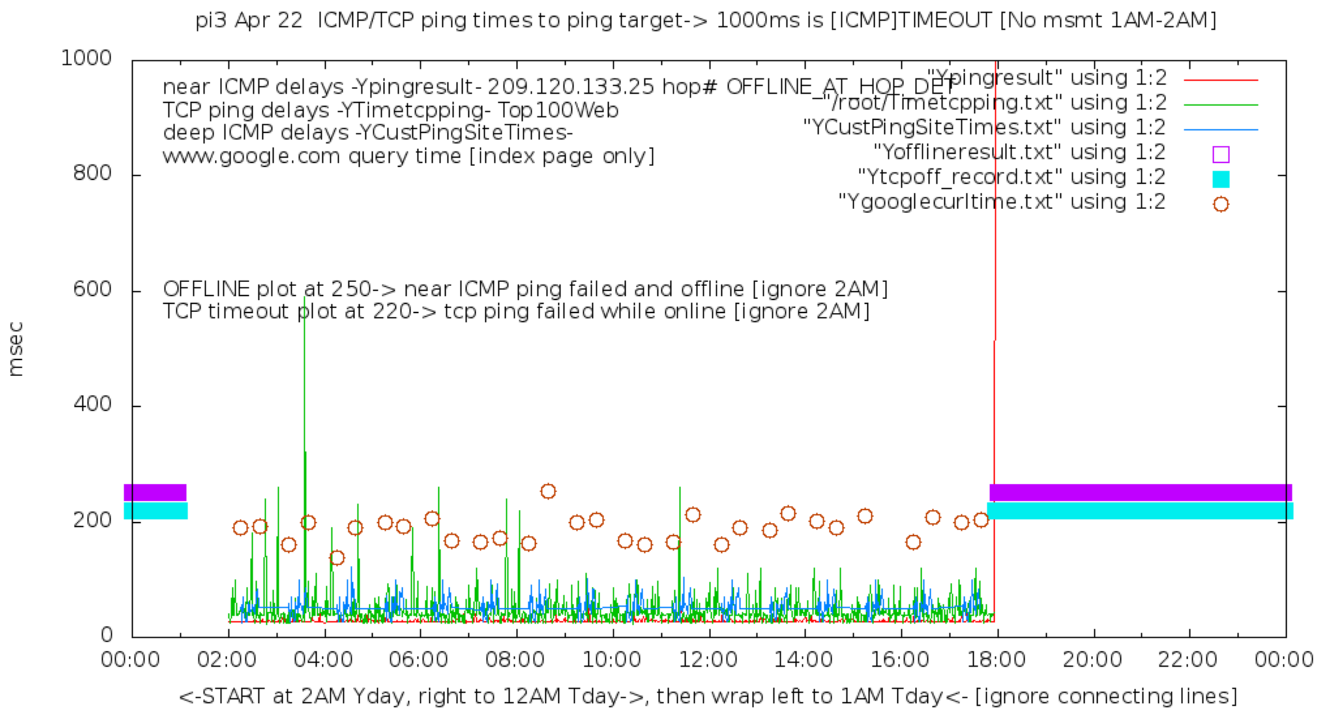
13. Plots

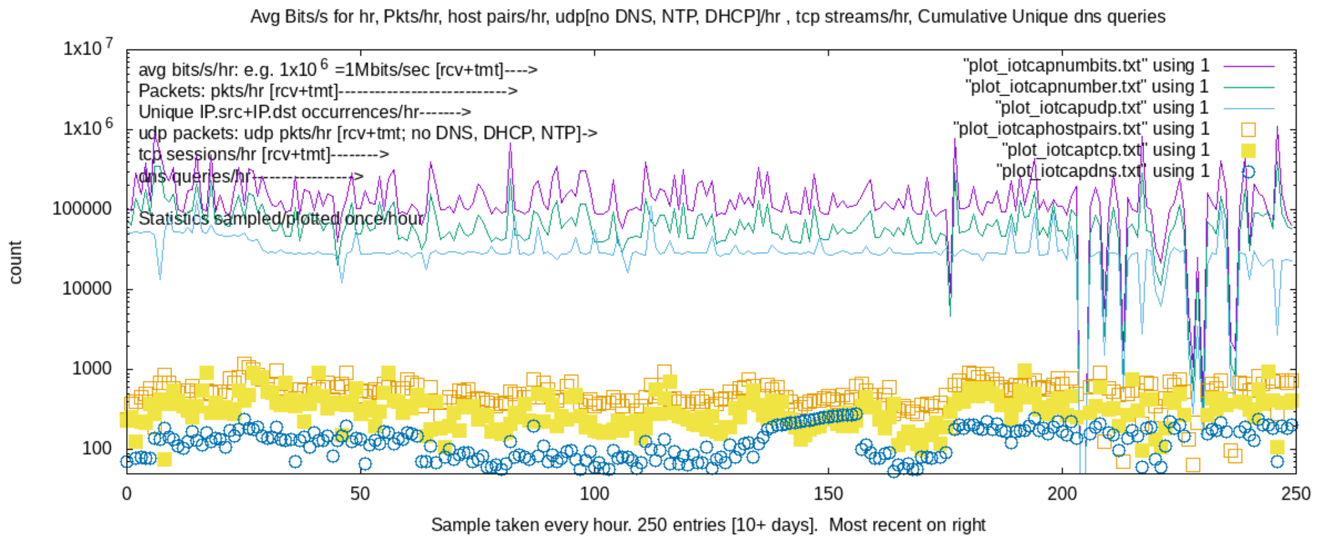
One of the neatest things you have access to is the daily [delivered via email if it is setup, otherwise available on the web page], and on-call real time plots of performance. Here are a few sample plots received in the daily email.

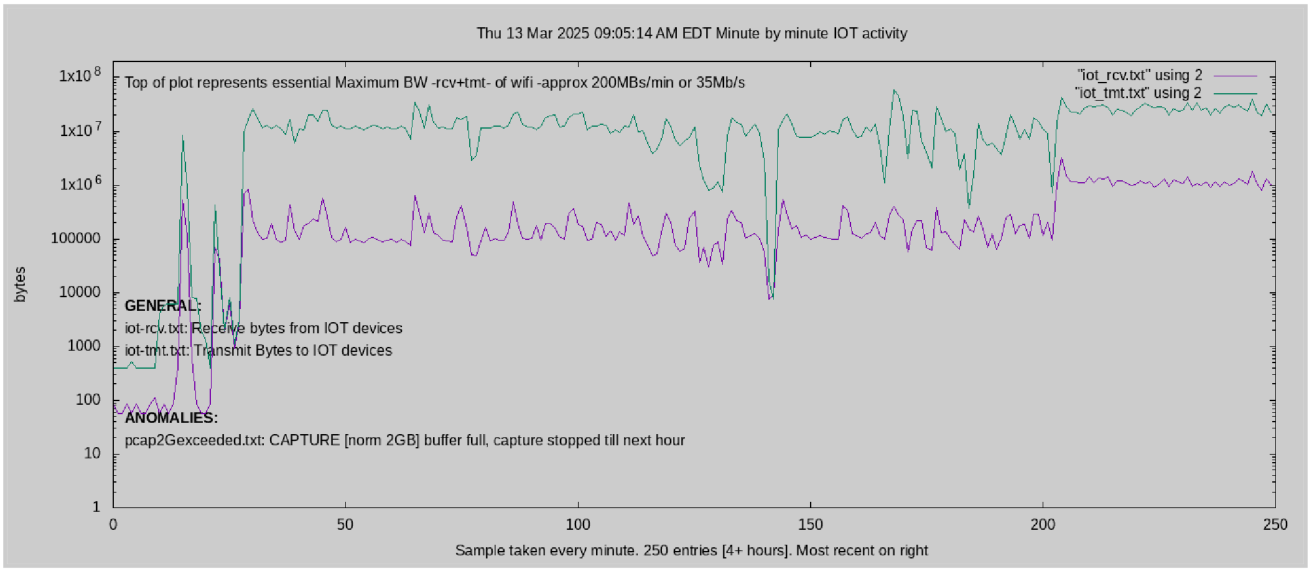
There are three plots shown on the raspberry pi home web page:

1. The combined plot, shown here, is the same received in the daily email, except it represents the plot from 2AM until the time it is requested.

2. The iotsnoop plot [second and third plots] cycles between a longer term characterization -about 10 days- and a short term characterization -about 4 hours.







14. -imonitorg- Plot explanation

There are 2 main LINE plots: The ping delays of a close router [pingresult], and the TCP delays of top 100 web sites [Timetcpping]. These are the two colored lines. The near ICMP ping delays and the TCP delays are every minute for 23 hours [2AM till next 1AM]. Typically, the near ping delays should be less than the TCP delays and be fairly constant; the TCP delays will vary widely, normally between 20-100msec. The TCP delays will typically be to the nearest Content Distribution Network (CDN) which host most of the major web sites.

If "customerpings" are enabled, there will be a third line plot.

The third and fourth plot is the points that represent the twice hourly fetch of www.google.com - the curl time and the dns query time. This will typically have the longest delay. DNS name resolution will have to be performed for this, and for the TCP delays. The ICMP ping delays do do a DNS name resolution.

The fifth and sixth "plot" is actually an overlay on the plots which will represent offline times experienced, and TCP timeouts experienced, but still on-line. These will appear as bars when there are extended periods of offline state, or multiple TCP ping failures. These are plotted at the 250 and 220 msec delay times.

Added in Aug 2021 is the "hyperping" plot points. This will be successive **vertical** ticks on the ICMP ping plot timeout line, which designate ping timeouts at 6/min rate, invoked when the 1/min ping rate fails.

Implemented on the last few releases [but not shown on these example plots] are two additional plot lines:

-ipv6 connectivity. There will be a "roofline" if there is no ipv6 default gateway

-router is used as near ICMP target, thus compromising Internet performance metrics.

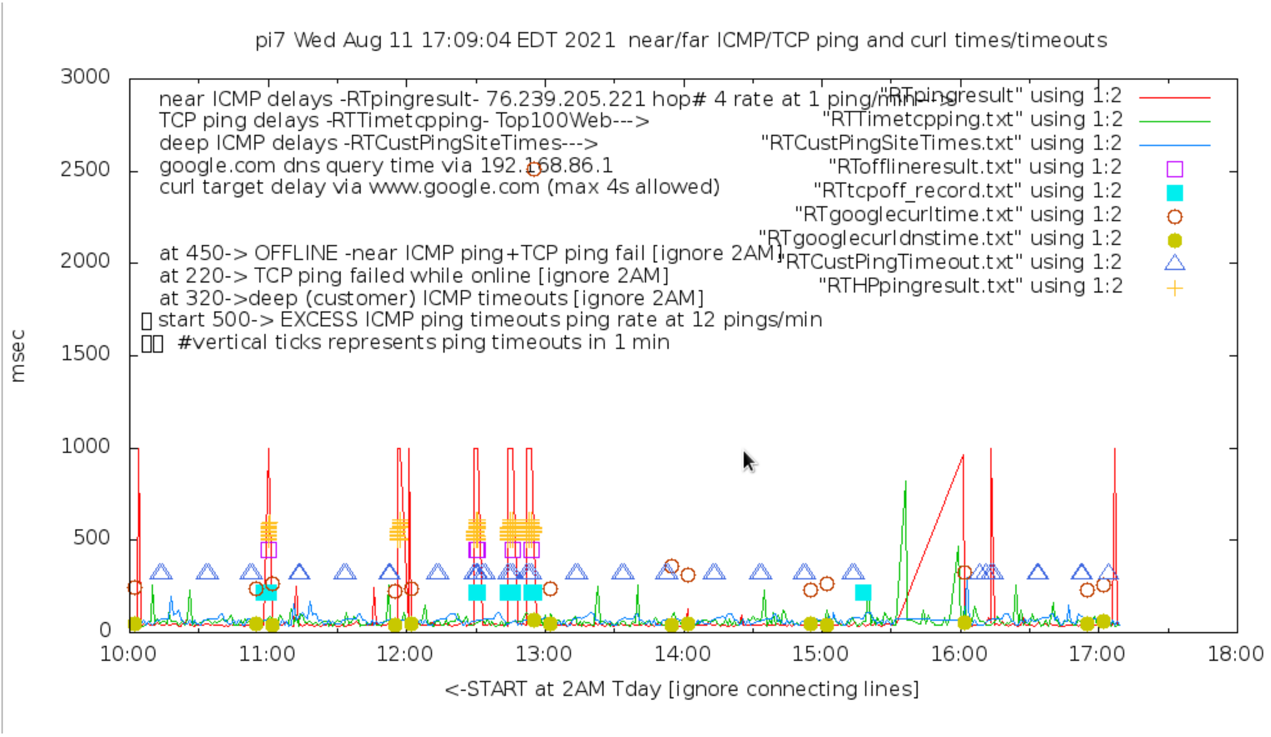
-near ICMP ping target changes are flagged on the plot. These are recalculated each hour if not LOCKed.

As an option on the iotsnoop/imonitorg, the customer can place a file "CustPingSites.newtxt" in the directory /home/pi/tests/customerpings/ and the scripts will automatically take up this file at [the next] 2AM and perform "deep ICMP" ping plotting. Originally these sites represented the ping targets of imonitor [as opposed to imonitorg] customers, so they represented ICMP pingable sites 5- hops from each of the customers on the trial project across the US. This file "CustPingSites.txt" was collected each night and copied to each pi for the use of the "deep ICMP" plots the next day. For the imonitorg of course there is no connection to the main imonitorg server, so this file has to be manually constructed. There is an example in the "files" of the sourceforge site https://imonitorg.sourceforge.io - the user can copy this file and use it, or modify it and place it in the customerpings directory. This will only perform the pings for a 23 hour period starting at 2AM. If the customer wants to make this plotting "permanent," he needs to make a pi crontab entry to replace his file in that directory each day.

E.g. save the file as CustPingSites.mytxt in /home/pi. Make a pi crontab entry to cp the file to "CustPingSites.newtxt" each day anytime before 12AM.

```
25 23 * * * /bin/cp CustPingSites.mytxt tests/customerpings/CustPingSites.newtxt
```

This will copy the customer file to the directory each day at 11:25PM. The CustPingsiteScript will use this script for the next day starting at 2AM. Here is a [partial day using "create RTplot" on web page] sample plot showing the "deep ICMP" plotting:


pi7 Wed Aug 11 17:09:04 EDT 2021 near/far ICMP/TCP ping and curl times/timeouts
near ICMP delays -RTpingresult- 76.239.205.221 hop# 4 rate at 1 ping/min--->
TCP ping delays -RTTimetcpping- Top100Web--->
deep ICMP delays -RTCustPingSiteTimes--->
google.com dns query time via 192.168.86.1
curl target delay via www.google.com (max 4s allowed)
at 450-> OFFLINE -near ICMP ping+TCP ping fail [ignore 2AM]
at 220-> TCP ping failed while online [ignore 2AM]
at 320->deep (customer) ICMP timeouts [ignore 2AM]
start 500-> EXCESS ICMP ping timeouts ping rate at 12 pings/min
#vertical ticks represents ping timeouts in 1 min
"RTpingresult" using 1:2
"RTTimetcpping.txt" using 1:2
"RTCustPingSiteTimes.txt" using 1:2
"RTofflineresult.txt" using 1:2
"RTtcpoff_record.txt" using 1:2
"RTgooglecurltime.txt" using 1:2
"RTgooglecurldnstime.txt" using 1:2
"RTCustPingTimeout.txt" using 1:2
"RTHPpingresult.txt" using 1:2
msec
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
<-START at 2AM Tday [ignore connecting lines]