V1 29 January 2022

SUCCESS RATES OF UPLOADS TO WSPRNET.ORG

Gwyn Griffiths, G3ZIL gwyn@autonomousanalytics.com

Summary of findings

1. The upload success rate with WsprDaemon is 100%. Spots are uploaded in bulk using MEPT (curl to http://wsprnet.org/meptspots.php).
2. The upload success rate using KiwiSDR WSPR 1.3 extension is about 99.9%. Most of these spots arrive well after those from WSJT-X and WsprDaemon within a two-minute interval.
3. The upload success rate for WSJT-X depends on type of computer used, affected by the load on the wsprnet.org server, and varies with time of day. Success rates for a Pi4B running a single instance of WSJT-X range from 88% to 97%. Success rate for Mac Mini, Macbook pro and an HP i5 quad-core 1.8 GHz laptop with Windows10 varied from 99.5% to 99.9%.
4. Examination of TCP transactions for failed spot uploads show instances of no receipt of acknowledgements from wsprnet.org yet POSTs from the WSJT-X client and wsprnet.org resetting the connection. WSJT-X itself seems not to retry failed uploads.
5. It's not easy to draw a conclusion from an analysis with Spotnum as a proxy for time of uploads within a two-minute interval. In particular we cannot reconcile the relative 'tightness' of the Pi4 spots' Spotnums, suggesting the spot-by-spot uploads were not drawn out in time, with the fact that only 9 out of 24 were uploaded in that interval.

Preliminary

- We have examined the WSPR upload success rates for three packages: WsprDaemon, the KiwiSDR WSPR extension, and five releases of WSJT-X. Between them, these packages provide the large majority of spots to wsprnet.org; Figure 1 shows a snapshot.

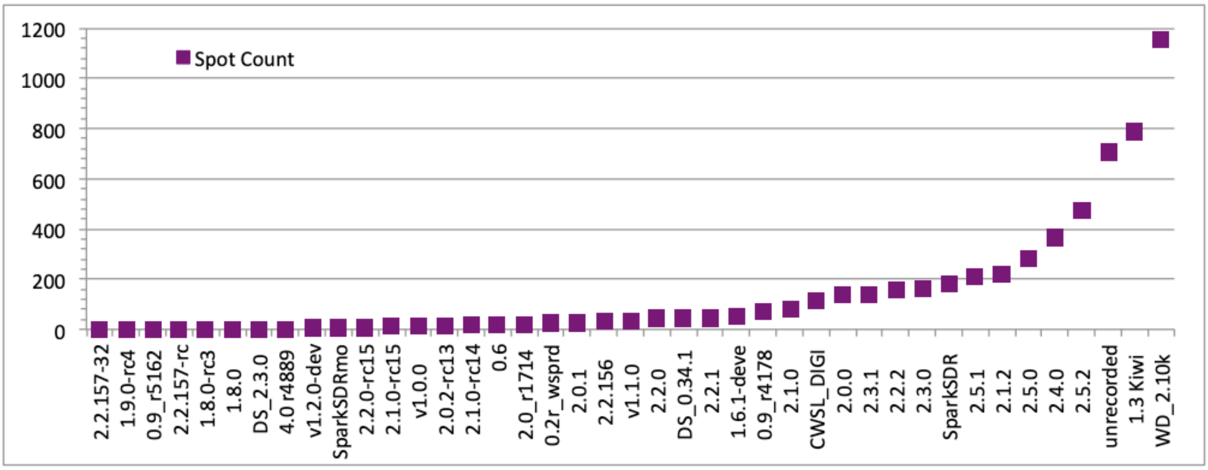


Figure 1. Number of spots reported by different software packages/versions of for 17:06 UTC on 1 December 2021 from the WsprDaemon table. Note that 'Unrecorded' includes those decoded by WsprDaemon versions prior to 2.10k and those uploaded via the old database interface.

1. WsprDaemon version 2.10k

- A test at G3ZIL on 40 m spanned part of 28–29 December 2021 with results summarised below. There were no instances of spots in the ALL_WSPR.TXT file not uploaded to wsprnet.org by the WsprDaemon client. By separate inspection, the match was there in all details of the spots not just the total number.

Number of 2-minute intervals:		487
Total G3ZIL 40 m spots in wsprnet.org:		2395
Total G3ZIL 40 m spots in ALL_WSPR.TXT:	2408	
Of which duplicates, transmitter with spur:	3	
Of which Type 2 spot with no CallSign but <...>	10	
Number of spots with understood reason why they were in ALL_WSPR.TXT but not expected to be in wsprnet.org:	13	
Net number of spots in ALL_WSPR.TXT expected to be in wsprnet.org:		2395

2. KiwiSDR WSPR extension v1.3

- The KiwiSDR has a built-in WSPR decoding extension, with access and setup via a web browser. This application (last revised in 2015), on average, provides the second-most number of spots to wsprnet.org after WsprDaemon. Tests of spot upload success rate were made at G3ZIL (reporting as G3ZIL/K) on 20 m.
- Test 1 spanning 10:32 to 17:44 on 30 December 2021 with a Chrome browser on an Apple Macbook Pro. As no ALL_WSPR.TXT file was accessible the decoded spots were copied and pasted from the extension's window. Of 819 spots in this interval 818 were reported successfully to wsprnet.org, a success rate of 99.9%. There was no obvious issue with the single spot that was not uploaded.
- Test 2 spanned 08:30 to 15:38 on 3 January 2022 using the same setup. All 1010 spots were reported successfully.
- We conclude the KiwiSDR 1.3 extension is capable of an over 99.9% spot upload success rate.

3. WSJT-X client

- Locally recorded ALL_WSPR.TXT spot files were uploaded to database tables at WsprDaemon.org to enable these comparisons using the bash pre-processing and postgreSQL in Annex 1.

WSJT-X version	**LAN**	**Duration**	**Processor**	**Count (2) in ALL_WSPR**	**Count in wsprnet.org**	**Success rate %**
2.4	WiFi 2.4 GHz	22 h	Pi4	3578	3323	92.9
2.4	WiFi 2.4 GHz	29 h	Pi4	4240	3714	87.6
2.4	WiFi 2.4 GHz	70 h	Pi4	6590	6220	94.4
2.5.4	100 Mbs Ethernet	22 h	Pi4	2091	2028	97.0
2.5.4	100 Mbs Ethernet	20 h	Pi4	5780	5320	92.0
2.5.4	100 Mbs Ethernet	78 h	Pi4	24569	23132	94.2
2.2.1	*100Mbs Ethernet (1)*	*33 d*	*Mac Mini*	*200200*	*199958*	*99.9*
2.5.3	*WiFi 2.4 GHz*	*19.5 h*	*Macbook Pro*	*997*	*992*	*99.5*
2.5.2	*100 Mbs Ethernet*	*78 h*	*Macbook Pro*	*6132*	*6101*	*99.5*
2.5.2	*100 Mbs Ethernet*	*50 h*	*Macbook Pro*	*18340*	*18367*	*99.9*
2.5.2	*WiFi 5 GHz*	*33 d*	*i5 Win10*	*43586*	*43451*	*99.7*

Table 1. Success rate defined as spots in wsprnet.org divided by spots in ALL_WSPR.TXT after the considerations in the Notes below together with LAN connection and computer running WSJT-X. All tests were with Fiber to the Cabinet/Kerb WAN connections. Different colours are different reporters.

1. A managed circuit, delivering Ethernet to the premises from a router owned and remotely managed by the ISP. Assured latency, not a shared connection.
2. Invalid spots, e.g. tx_call of <...>, or clearly in error, e.g. G4P/000AAA in ALL_WSPR.TXT, and own tx_call are excluded as are spots received when wsprnet was down on 23 December 2021 between19:02 – 20:00 UTC.

- Initial tests were at G3ZIL with WSJT-X 2.4 on a Raspberry Pi 4B, results are in table 1 in red showing success rates of 88–94%. To check this disappointing success rate other reporters kindly contributed their ALL_WSPR.TXT files and details of their local setups (in green, purple and blue). It appears that success rates can be setup-dependent. That is, while the method and code in WSJT-X and reception and insertion code at wsprnet.org are capable of 99.9% upload success there are circumstances in which that is not the case.
- The average for Pi4 is 93.0% +4%/-5% and 99.7% +/-0.2% for Mac/PC. The two sets do not overlap, they are statistically completely different.
- As far as we know there is no mechanism in the WSJT-X client software to ensure that all valid decoded spots from ALL_WSPR.TXT are present in the wsprnet.org database.
- The following sections cover our lines of inquiry in an attempt to better understand the lower upload success rate with the Pi4:
  - Is there a correlation between spot upload success and the overall load on the wsprnet.org server?
  - What happens in the TCP transactions between wsprnet.org and the WSJT-X client when spots are uploaded normally and when they are missed?
  - Does the time into a two-minute interval affect the probability of a successful upload?

3.1 Upload success and the overall load on the wsprnet.org server

- Taking the number of spots from all reporters uploaded in a two-minute interval as a proxy for load on the wsprnet.org server we see a clear diurnal pattern with time of day, Figure 2.
- The time series of the count of spots present in the Pi4 ALL_WSPR.TXT file but not successfully uploaded to wsprnet.org, Figure 3, suggests that spot uploads are less successful when the load on the wsprnet.org server is greatest. For 30-31 Dec 255 out of 3578 spots were not uploaded over 619 two-minute intervals, a success rate of 93%, and for 31 Dec - 1 Jan 526 spots were not uploaded over 667 two-minute intervals, a success rate of 88%.
- The upload success rate using a Pi4 and WSJT-X depends on the load on the wsprnet.org server, and varies with time of day.

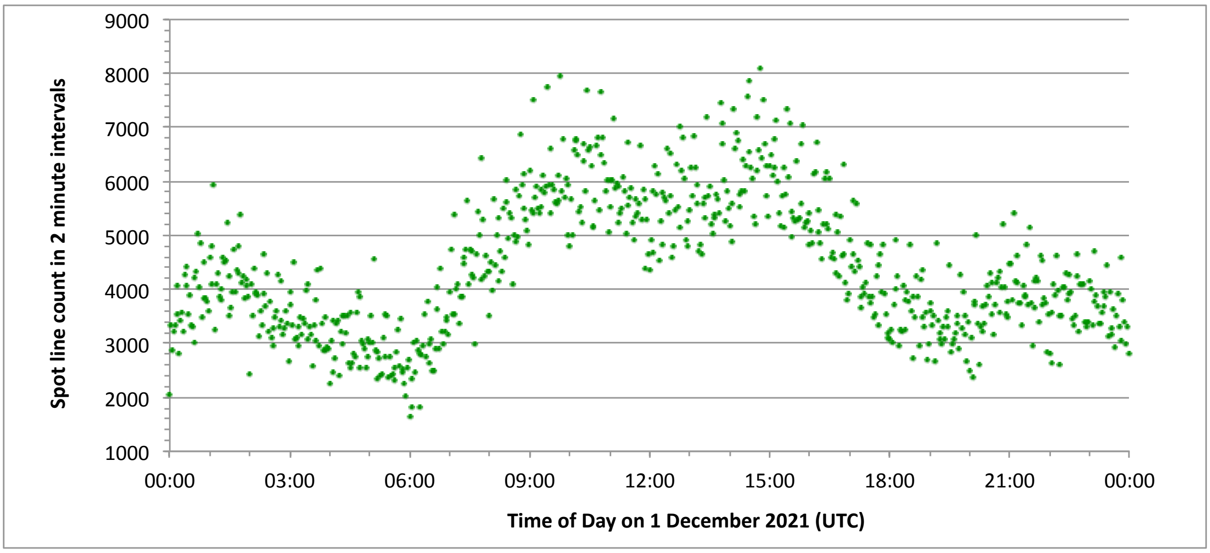


Figure 2. Spot count in two-minute intervals for 1 December 2021 for wsprnet.org.

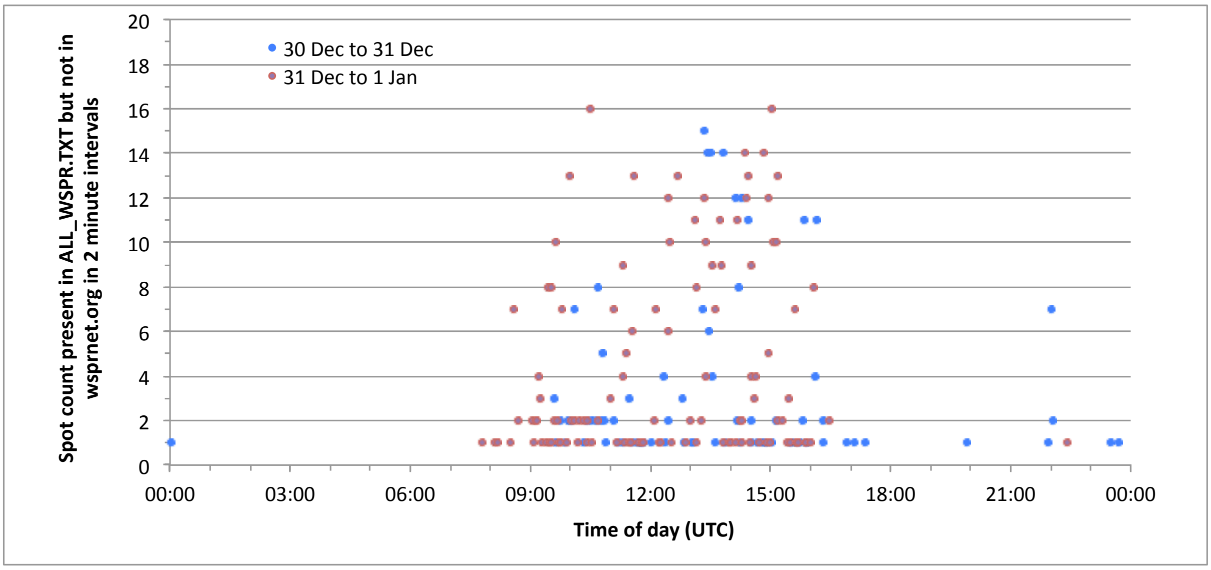
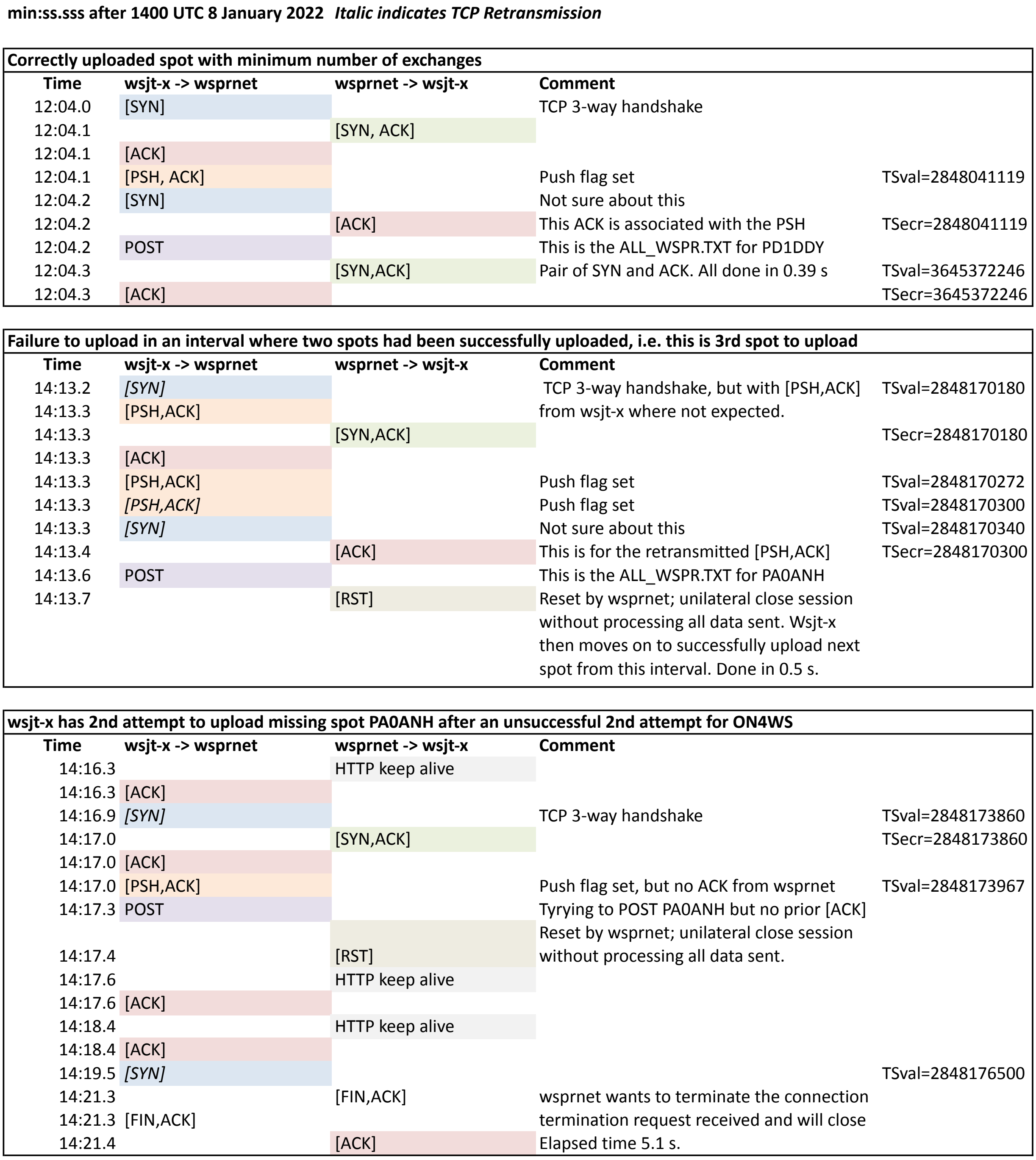


Figure 3. Time series of spot count present in G3ZIL ALL_WSPR.TXT file in two-minute intervals but not in wsprnet.org on two days.

3.2 TCP transactions between wsprnet.org and WSJT-X client

- The upper panel in Table 2 shows the TCP transactions, captured using Wireshark, between the wsprnet.org server and a Pi4 running WSJT-X 2.4 when a spot is successfully uploaded. Each spot is uploaded separately using a POST. The transactions for this single spot upload spanned 0.39 s.
- The centre panel shows a failed upload, immediately after two spots had been uploaded successfully. wsprnet.org has unilaterally closed the session using a [RST].
- The lower panel, starting 2.6 s later is an automatic retry. There is no [ACK] from wsprnet.org after the [PSH,ACK] from the Pi4, nevertheless the Pi4 issues a POST; there is no [SYN,ACK] from wsprnet.org, rather there is a [RST] and a subsequent [FIN,ACK] that closes the session. There is no attempt by WSJT-X to retry uploading the failed spots. This failed attempt to upload a single spot has spanned 5.1 s.
- We are not sufficiently familiar to provide a further interpretation of what may be the cause of this pattern of TCP transactions.



min:ss.sss after 1400 UTC 8 January 2022 ***Italic indicates TCP Retransmission***

Correctly uploaded spot with minimum number of exchanges

Time	wsjt-x -> wsprnet	wsprnet -> wsjt-x	Comment	
12:04.0	[SYN]		TCP 3-way handshake	
12:04.1		[SYN, ACK]		
12:04.1	[ACK]			
12:04.1	[PSH, ACK]		Push flag set	TSval=2848041119
12:04.2	[SYN]		Not sure about this	
12:04.2		[ACK]	This ACK is associated with the PSH	TSecr=2848041119
12:04.2	POST		This is the ALL_WSPR.TXT for PD1DDY	
12:04.3		[SYN,ACK]	Pair of SYN and ACK. All done in 0.39 s	TSval=3645372246
12:04.3	[ACK]			TSecr=3645372246

Failure to upload in an interval where two spots had been successfully uploaded, i.e. this is 3rd spot to upload

Time	wsjt-x -> wsprnet	wsprnet -> wsjt-x	Comment	
14:13.2	*[SYN]*		TCP 3-way handshake, but with [PSH,ACK] from wsjt-x where not expected.	TSval=2848170180
14:13.3	[PSH,ACK]			
14:13.3		[SYN,ACK]		TSecr=2848170180
14:13.3	[ACK]			
14:13.3	[PSH,ACK]		Push flag set	TSval=2848170272
14:13.3	*[PSH,ACK]*		Push flag set	TSval=2848170300
14:13.3	*[SYN]*		Not sure about this	TSval=2848170340
14:13.4		[ACK]	This is for the retransmitted [PSH,ACK]	TSecr=2848170300
14:13.6	POST		This is the ALL_WSPR.TXT for PA0ANH	
14:13.7		[RST]	Reset by wsprnet; unilateral close session without processing all data sent. Wsjt-x then moves on to successfully upload next spot from this interval. Done in 0.5 s.	

wsjt-x has 2nd attempt to upload missing spot PA0ANH after an unsuccessful 2nd attempt for ON4WS

Time	wsjt-x -> wsprnet	wsprnet -> wsjt-x	Comment	
14:16.3		HTTP keep alive		
14:16.3	[ACK]			
14:16.9	*[SYN]*		TCP 3-way handshake	TSval=2848173860
14:17.0		[SYN,ACK]		TSecr=2848173860
14:17.0	[ACK]			
14:17.0	[PSH,ACK]		Push flag set, but no ACK from wsprnet	TSval=2848173967
14:17.3	POST		Tyrying to POST PA0ANH but no prior [ACK]	
14:17.4		[RST]	Reset by wsprnet; unilateral close session without processing all data sent.	
14:17.6		HTTP keep alive		
14:17.6	[ACK]			
14:18.4		HTTP keep alive		
14:18.4	[ACK]			
14:19.5	*[SYN]*			TSval=2848176500
14:21.3		[FIN,ACK]	wsprnet wants to terminate the connection	
14:21.3	[FIN,ACK]		termination request received and will close	
14:21.4		[ACK]	Elapsed time 5.1 s.	

Table 2. TCP transactions captured using Wireshark on a Pi4 running WSJT-X 2.4 interacting with wsprnet.org for a successful spot upload and two failures.

3.3 Spot upload timing from WSJT-X, WsprDaemon and KiwiSDR WSPR extension

- We cannot directly answer the question, "Does the time into a two-minute interval affect the probability of a successful upload?" as we have no method of knowing when a whole mass of spots arrive at the wsprdaemon.org server and are accepted. Other, that is, than the upload times from one machine monitored using Wireshark as above.
- However, the auto-index Spotnum field added to WSPR spots by wsprnet.org can be used as a proxy for time. Spotnums increase monotonically with time spots are received but may have gaps, and will undoubtedly be non-linear. Nevertheless, as shown for one two-minute interval in Figure 4, there is a broad pattern in the arrival of spots from WSJT-X, WsprDaemon and the KiwiSDR WSPR extension in 'Spotnum space'.

- The overall picture is that WSJT-X spots (geen), uploaded one-at-a-time using POST, arrive first. Next come spots from WsprDaemon users, uploaded in bulk using MEPT. The later arrival is not a surprise given the noise level and other processing within WsprDaemon. Finally, the majority of KiwiSDR WSPR extension spots arrive.
- We have no insight into the early arrival of small numbers of KiwiSDR WSPR extension spots, which can occur from a single reporter, Figure 5.
- We suspect that WsprDaemon early arrivals may come from some users using Intel i5 or i7-based machines rather than the more widespread use of Pi3s and Pi4, or from users spotting on only a few bands simultaneously. The early WsprDaemon arrivals from WA2TP in Figure 5 fall into this category.

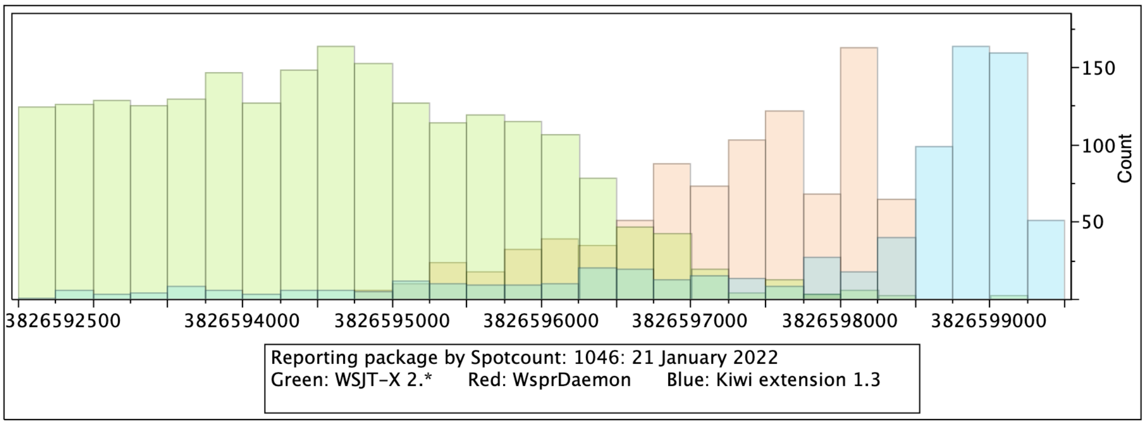


Figure 4. Arrivals of spots at wsprnet.org from three packages in "Spotnum" space - a proxy for time.

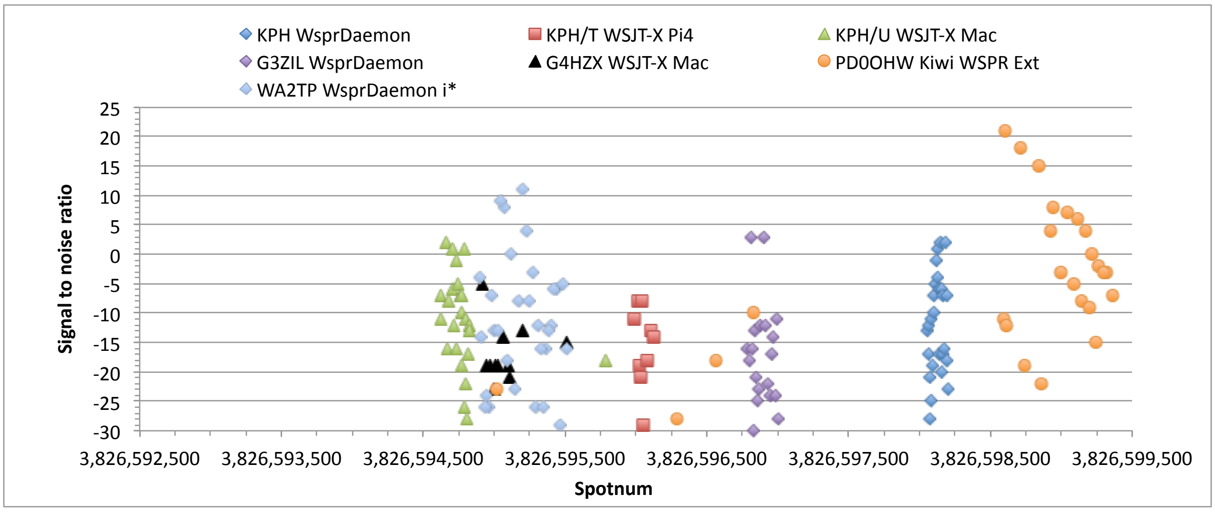


Figure 5. Arrivals of spots at wsprnet.org seven reporters using different packages in "Spotnum" space.

- The Spotnum scale in Figure 5 covers the same span as Figure 4, but shows spots from individual reporters, in order of increasing Spotnum:
  - Green triangles: A single instance of WSJT-X decoding audio on a Macbook Pro reporting as KPH/U. There is little spread in Spotnums for these 24 spots.
  - Black triangles: A single instance of WSJT-X decoding audio at G4HZX on a Mac Mini, showing greater spread in the 12 Spotnums.
  - Blue-Grey diamonds: The first WsprDaemon spots this interval. These were from WA2TP using an i9 10 core 3.6 GHz custom-built machine, showing as much spread in Spotnums for these 30 transferred using MEPT as for G4HZX with WSJT-X uploading spot-by-spot.

  - Red squares: A single instance of WSJT-X decoding exactly the same audio as for KPH/U above, but running on a Pi4 and reporting as KPH/T. Tight range of Spotnums, but only 9 spots uploaded rather than the 24 of the Macbook.
  - Purple diamonds: G3ZIL WsprDaemon on a Pi4 with 19 spots.
  - Blue diamonds: KPH WsprDaemon on a Pi4 with 26 spots. These are toward the end of the WsprDaemon uploads from the histogram of Figure 4.
  - Orange circles: KiwiSDR WSPR extension 1.3: A few isolated spots have been uploaded earlier. It is possible that there are multiple instances of the extension running. The three sloping sets of spots here are consistent with decoders on three bands, with the spots with highest SNR on each band, separately, decoded first. The spread in Spotnums is consistent with the slow decoding with this extension that uses a 2015 version of wsprd and is running on the same BeagleBone green as the SDR. These uploads are spot-by-spot but have little contention from spots from other packages, they have already been uploaded.
- It's not easy to draw a conclusion from this analysis beyond the descriptions above. In particular we cannot reconcile the relative 'tightness' of the KPH/T Pi4 spots in red, suggesting the spot-by-spot uploads were not drawn out in time, with the fact that only 9 out of 24 were uploaded.

Annex 1. Bash script and postgresql for success rates for WSJT-X uploads to wsprnet

Bash script allwspr.sh to prepare the ALL_WSPR.TXT file to be suitable for import into postgresql:

```
#!/bin/bash

### This bash script prepares an ALL_WSPR.TXT file in the standard format for upload to a postgresql database table
### Gwyn Griffiths January 2022
### Run in same directory as the ALL_WSPR.TXT file
# Step 1 get rid of < and > that bracket entries from type 2 spots, either with tx_call between them, which we will keep
# or with ... which we will drop, as these are not in wsprnet.org
echo "Reading ALL_WSPR.TXT file in this folder and converting for postgresql input as all_wspr.csv"
tr -d '<>' <ALL_WSPR.TXT >temp.txt
# awk line that does multiple operations:
# 1. If the sixth field, the tx_call is ... from being a location only type 2 then we do not print any field
#    The ... has to be set as a variable and the if operates on that variable
# 2. Adds 20 to the start of the year number e.g. 22 to get 2022
# 3. The substr extract the year, month and day and we insert - between them
# 4. Separate the hour and minutes using substr with a : between. We now have a valid postgresql timestamp
# 5. only print the minimal number of, comma separated, fields for us to run a comparison with the spots table via API from wsprnet,org
awk -v var="..." '{ if ($6!=var) print "20" substr($1,1,2) "-" substr($1,3,2) "-" substr($1,5,2) " " substr($2,1,2) ":" substr($2,3,2) "," \
 $3 ","$5 "," $6 "," $7 }' <temp.txt >all_wspr.csv
echo "all_wspr.csv created"
```

Create table for minimal data from ALL_WSPR.TXT after it has been processed as above. Only needs to be done once, can use same table for other instances of same data format.

```
create table all_wspr (time timestamp without time zone not null, snr integer, frequency real,tx_call varchar,tx_grid varchar);
```

Copy a file that has been scp from the Mac into the all_wspr table

```
copy all_wspr(time,snr,frequency,tx_call,tx_grid) from '/home/scraper/all_wspr.csv'
delimiter ',' CSV;
```

List those spots in the all_wspr table that are not in the spots table sourced by API from wsprnet.org:

```
\copy (SELECT all_wspr.time, all_wspr.snr, all_wspr.frequency, all_wspr.tx_call,
all_wspr.tx_grid FROM all_wspr left join spots on all_wspr.time = spots.wd_time and
spots."CallSign"=all_wspr.tx_call and spots."Reporter"='G3ZIL/A' and spots.wd_time >=
'2021-12-30 09:20:00' and spots.wd_time <= '2021-12-31 06:52:00' where spots."CallSign"
is null order by all_wspr.time asc) to '/tmp/joined.csv' with csv;
```

Get time and count of spots in each two minute interval that are in the all_wspr table but not in the spots table:

```
\copy (SELECT all_wspr.time,count(*) FROM all_wspr left join spots on all_wspr.time =
spots.wd_time and spots."CallSign"=all_wspr.tx_call and spots."Reporter"='G3ZIL/A' where
all_wspr.time >= '2021-12-31 18:16:00' and all_wspr.time <= '2022-01-01 23:00:00' and
spots."CallSign" is null group by all_wspr.time order by all_wspr.time asc) to
'/tmp/joined.csv' with csv;
```

Get time and count from all intervals in the ALL_WSPR.TXT file, not just those with missing spots in wsprnet.org:

```
\copy (SELECT all_wspr.time,count(*) FROM all_wspr where all_wspr.time >= '2021-12-31
18:16:00' and all_wspr.time <= '2022-01-01 23:00:00' group by all_wspr.time order by
all_wspr.time asc) to '/tmp/all_wspr_count.csv' with csv;
```

Then use Excel to extract times when there were missing spots (see spreadsheet).

To get spot count of all of wsprnet.org two-minute intervals over this span

```
\copy (SELECT wd_time,count(*) FROM spots where wd_time >= '2021-12-31 18:16:00' and
wd_time <= '2022-01-01 23:00:00' group by wd_time order by wd_time asc) to
'/tmp/spots_count.csv' with csv;
```

In the comparison experiment between spots decoded and uploaded by a Pi4B and a Macbook pro the ALL_WSPR.TXT file, especially on the Pi, had many spurs from W6LVP, up to 12 entries in one 2 minute interval. Because of different frequencies, they were not removed by wsprnet.org and so are in the spots table, but I can't be sure, so W6LVP spots deleted from the table from ALL_WSPR.TXT and then not considered in the join, for the Pi:

```
\copy (SELECT ai6vn_all_wspr.time,count(*) FROM ai6vn_all_wspr left join spots on
ai6vn_all_wspr.time = spots.wd_time and spots."CallSign"=ai6vn_all_wspr.tx_call and
abs(spots."MHz"-ai6vn_all_wspr.frequency)<0.000002 and spots."Reporter"='KPH/T' and
spots."CallSign"!='W6LVP' where ai6vn_all_wspr.time >= '2022-01-20 22:16:00' and
ai6vn_all_wspr.time <= '2022-01-21 18:44:00' and spots."CallSign" is null group by
ai6vn_all_wspr.time order by ai6vn_all_wspr.time asc) to '/tmp/joined.csv' with csv;
```