NAME

aprx−stat − statistics utility for **aprx**(8)

SYNOPSIS

aprx-stat [−t] [−f /tmp/aprx−erlang.dat] {−S|−x|−X}

DESCRIPTION

aprx−stat is a statistics utility for **aprx**(8) program.

OPTIONS

The **aprx−stat** has following runtime options:

−t Use UNIX *time_t* for timestamps, instead of human readable text format.

−f /tmp/aprx−erlang.dat
Turn on verbose debugging, outputs data to STDOUT.

−S SNMP data mode, current counter and gauge values.

−x Lattest of extended historical gauge values. This gives for each input interface

- SNMP data
- last 90 of 1 minute values,
- 10 of 10 minute values,
- 3 of 60 minute values.

−X Full extended historical gauge values. This gives all the contents of historical value data ring-buffers.

- SNMP data
- 1 minute resolution: 24 hours
- 10 minute resolution: 7 days
- 60 minute resolution: 3 months

SNMP DATA OUTPUT

For each interface feeding AX.25 packets and/or KISS frames to this program, there are following kind of

```
SNMP  /dev/ttyUSB1   798282 11088   0  0     3
SNMP  ax0   798282 11088   0  0     7
SNMP  ax1   798282 11088   0  0     94
```

where columns are:

- "SNMP" - keyword
- Interface (AX.25 IF name, or serial port device name)
- Received byte counter
- Received KISS frame (packet) counter
- Transmitted byte counter (will stay zero)
- Transmitted KISS frame counter (will stay zero)
- Age in seconds of last update of this statistics.

EXTENDED DATA OUTPUT

Extended data output gives formatted historical periodic accumulates of interface traffic counters, and Erlang value estimates based on that.

```
SNMP  /dev/ttyUSB1   816675 11332   0  0     15

1min data
2007-12-24 14:10  /dev/ttyUSB1  1m    374    6      0    0   0.047  0.000
2007-12-24 14:09  /dev/ttyUSB1  1m    377    5      0    0   0.047  0.000
2007-12-24 14:08  /dev/ttyUSB1  1m    347    5      0    0   0.043  0.000
2007-12-24 14:07  /dev/ttyUSB1  1m    140    2      0    0   0.018  0.000
•••

10min data
2007-12-24 14:10  /dev/ttyUSB1 10m   3829   55      0    0   0.048  0.000
2007-12-24 14:00  /dev/ttyUSB1 10m   2182   28      0    0   0.027  0.000
2007-12-24 13:50  /dev/ttyUSB1 10m   3205   44      0    0   0.040  0.000
2007-12-24 13:40  /dev/ttyUSB1 10m   3811   50      0    0   0.048  0.000
•••

60min data
2007-12-24 14:00  /dev/ttyUSB1 60m  22510  295      0    0   0.047  0.000
2007-12-24 13:00  /dev/ttyUSB1 60m  24886  347      0    0   0.052  0.000
•••
```

The output repeats for all interfaces.

The SNMP dataset is given in the beginning, and described above. Then each extended output line has following fields:

- Timestamp is two fields, date and time (in minute resolution) is in UTC.
- Alternate timestamp format is UNIX *time_t* as an integer, counting seconds from epoch, and as single field.
- Interface name is same as in SNMP case.
- Data qualifier tells what integration period the data is valid for: *1m*, *10m*, *60m*.
- Counter of received bytes on interface (including KISS flags etc.)
- Counter of received frames.
- Counter of transmitted bytes on interface
- Counter of transmitted frames.
- Reception *Erlang* value estimate.
- Transmission *Erlang* value estimate.

TODO

BUGS

SEE ALSO

aprx(8)

CONFIGURATION FILE

There is no configuration file.

NOTES: ERLANG

The *Erlang* is telecom measurement of channel occupancy, and in this application sense it does tell how much traffic there is on the radio channel.

Most radio transmitters are not aware of all transmitters on channel, and thus there can happen a collision causing loss of both messages. The higher the channel activity, the more likely that collision is. For further details, refer to statistical mathematics books, or perhaps on Wikipedia.

In order to measure channel activity, the **aprx** program suite has these built-in statistics counter and summary estimators.

The *Erlag* value that the estimators present are likely somewhat *underestimating* the true channel occupancy simply because it calculates estimate of channel bit transmit rate, and thus a per-minute character capacity. It does not know true frequency of bit-stuffing events of the HDLC framing, nor each transmitter pre- and port frame PTT times. The transmitters need to stabilize their transmit oscillators in many cases, which may take up to around 500 ms! The counters are not aware of this preamble-, nor postamble-times.

The HDLC bit stuffing ratio is guessed to be 8.2 bits for each 8 bits of payload.

NOTES: SUID ROOT

This program needs probably to be run as *suid−root* ! It is fairly safe to do so, as this checks that the **−f** parameter file is of correct time, and will not try to create create it if it does not exist.

AUTHOR

This little piece was written by *Matti Aarnio, OH2MQK* during a dark and rainy fall and winter of 2007−2008 after a number of discussions grumbling about current breed of available software for APRS iGate use in Linux (or of any UNIX) platforms.

Principal contributors and test users include: *Pentti Gronlund, OH3BK*, *Reijo Hakala, OH1GWK*.