Amateur kernel networking development

Introducing the HamBSD project

irl

HamBSD

6th December 2019
NorthernRST 2019, Aberdeen, Scotland
$ whoami

- irl (MM0ROR)
- Foundation licence: March 2011
irl (MM0ROR)

Foundation licence: March 2011

MSci Computing Science: August 2014
$ whoami

- irl (MM0ROR)
- Foundation licence: March 2011
- MSci Computing Science: August 2014
- Intermediate licence: December 2014
$ whoami

- irl (MM0ROR)
- Foundation licence: March 2011
- MSci Computing Science: August 2014
- Intermediate licence: December 2014
- Debian Developer: July 2015
$ whoami

- irl (MM0ROR)
- Foundation licence: March 2011
- MSci Computing Science: August 2014
- Intermediate licence: December 2014
- Debian Developer: July 2015
- Full licence: October 2016
$ whoami

- irl (MM0ROR)
- Foundation licence: March 2011
- MSci Computing Science: August 2014
- Intermediate licence: December 2014
- Debian Developer: July 2015
- Full licence: October 2016
- Scottish Consulate ARC Founding Club Official: November 2019
What is amateur radio?

- A hobby
What is amateur radio?

- A technical hobby
What is amateur radio?

- A technical hobby with strict rules
What is amateur radio?

- A technical hobby with strict rules that requires a licence
What is amateur radio?

- A fun technical hobby with strict rules that requires a licence
What is amateur radio?

- Antennas

What is amateur radio?

- Antennas
- Electronics

What is amateur radio?

- Antennas
- Electronics
- History

https://commons.wikimedia.org/wiki/File:W6om.png CC BY-SA 3.0
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation

https://commons.wikimedia.org/wiki/File:Sm0_4s7ab.jpg CC BY-SA 3.0
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport

https://en.wikipedia.org/wiki/File:Frg01_hst.jpg CC BY-SA 3.0
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space

What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space
- Technique

Stone Mountain radio club
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space
- Technique
- EmComm

https://twitter.com/G1SSR
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space
- Technique
- EmComm
- Computing

https://screenshots.debian.net/package/fldigi
What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space
- Technique
- EmComm
- Computing
- Packet Networking

What is amateur radio?

- Antennas
- Electronics
- History
- Recreation
- Sport
- Space
- Technique
- EmComm
- Computing
- Packet Networking
- Internetworking

https://commons.wikimedia.org/wiki/File:HamNet,DBOMQT.jpg CC BY-SA 4.0
Automatic Packet Reporting System

- real time digital communications
Automatic Packet Reporting System

- real time digital communications
- information of immediate value
Automatic Packet Reporting System

- real time digital communications
- information of immediate value
- regarding the local area
Automatic Packet Reporting System

- real time digital communications
- information of immediate value
- regarding the local area
Automatic Packet Reporting System
One Way Trackers

- takes GPS data

https://commons.wikimedia.org/wiki/File:Byonics_AIO.jpg
Automatic Packet Reporting System
One Way Trackers

- takes GPS data
- converts to APRS packet

https://commons.wikimedia.org/wiki/File:Byonics_AIO.jpg
Automatic Packet Reporting System
One Way Trackers

- takes GPS data
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)

https://commons.wikimedia.org/wiki/File:Byonics_AIO.jpg
Automatic Packet Reporting System
One Way Trackers

- takes GPS data
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
- other amateurs see where you are

https://commons.wikimedia.org/wiki/File:Byonics_AIO.jpg
Automatic Packet Reporting System

Two Way Trackers

- takes GPS data
Automatic Packet Reporting System

Two Way Trackers

takes GPS data

converts to APRS packet
Automatic Packet Reporting System
Two Way Trackers

- takes GPS data
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
Automatic Packet Reporting System
Two Way Trackers

- Takes GPS data
- Converts to APRS packet
- Transmit on pre-agreed frequency (144.8MHz)
- Also receives packets from others
Automatic Packet Reporting System

Two Way Trackers

- takes GPS data
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
- also receives packets from others
- shows where other amateurs are in relation to you
Automatic Packet Reporting System

Two Way Messaging

- takes a message

hello i hope you are enjoying the talk!
Automatic Packet Reporting System

Two Way Messaging

- takes a message
- converts to APRS packet
Automatic Packet Reporting System

Two Way Messaging

- takes a message
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
Automatic Packet Reporting System
Two Way Messaging

- takes a message
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
- also receives packets from others
Automatic Packet Reporting System
Two Way Messaging

- takes a message
- converts to APRS packet
- transmit on pre-agreed frequency (144.8MHz)
- also receives packets from others
- enables two-way text messaging
Automatic Packet Reporting System

Digipeating

Alice

Bob
Automatic Packet Reporting System

Digipeating

[Graph showing Venn diagram with Alice, Dave, and Bob]
Automatic Packet Reporting System
APRS-IS and Internet Gateways (IGates)
Automatic Packet Reporting System
APRS-IS and Internet Gateways (IGates)
Automatic Packet Reporting System
Software - Xastir
Automatic Packet Reporting System
Software - Dire Wolf

direwolf.service - DireWolf is a software "soundcard" modem/TNC and APRS decoder
Loaded: loaded (/lib/systemd/system/direwolf.service; disabled; vendor preset: enabled)
Active: active (running) since Fri 2019-11-22 13:55:33 GMT; 1 weeks 4 days ago
   Docs: man:direwolf
Main PID: 3075 (direwolf)
    Tasks: 12 (limit: 2200)
   Memory: 1.3M
CGroup: /system.slice/direwolf.service
        3075 /usr/bin/direwolf -c /etc/direwolf.conf

Dec 03 18:30:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3GN *111111z5701.06N/0022
Dec 03 18:30:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3NG *111111z5736.13N/0020
Dec 03 18:40:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:5709.89NI00209.67W#Northfield
Dec 03 18:40:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3GN *111111z5701.06N/0022
Dec 03 18:40:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3NG *111111z5736.13N/0020
Dec 03 18:40:35 raspberrypi direwolf[3075]: [0L] MB7UAR>APDW14,wide1-1,wide2-1;;GB3GN *1111
Dec 03 18:40:35 raspberrypi direwolf[3075]: [0L] MB7UAR>APDW14,wide1-1,wide2-1;;GB3NG *1111
Dec 03 18:50:34 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:5709.89NI00209.67W#Northfield
Dec 03 18:50:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3GN *111111z5701.06N/0022
Dec 03 18:50:35 raspberrypi direwolf[3075]: [ig] MB7UAR>APDW14;:GB3NG *111111z5736.13N/0020

lines 1-20/20 (END)
Automatic Packet Reporting System

Versions of Xastir seen in about 30 minutes one evening in December
Automatic Packet Reporting System
Versions of Dire Wolf seen in about 30 minutes one evening in December
Automatic Packet Reporting System
Not Internet Safe

• reminds me of SS7 or BGP
Automatic Packet Reporting System
Not Internet Safe

- reminds me of SS7 or BGP
- originally no authentication built-in
Automatic Packet Reporting System
Not Internet Safe

- reminds me of SS7 or BGP
- originally no authentication built-in
- generally passes on anything received
Automatic Packet Reporting System
Not Internet Safe

- reminds me of SS7 or BGP
- originally no authentication built-in
- generally passes on anything received
- minimal filtering
Automatic Packet Reporting System
Not Internet Safe

- reminds me of SS7 or BGP
- originally no authentication built-in
- generally passes on anything received
- minimal filtering
- plaintext transport (though TLS is supported on some servers)
HamBSD
Can I do better?
HamBSD
Built on OpenBSD

- forked from NetBSD in 2005
HamBSD
Built on OpenBSD

- forked from NetBSD in 2005
- secure by default
HamBSD
Built on OpenBSD

• forked from NetBSD in 2005
• secure by default
• emphasis on code quality
HamBSD
Built on OpenBSD

- forked from NetBSD in 2005
- secure by default
- emphasis on code quality
- known for its high-quality documentation
HamBSD

Goals

- KISS TNC support
HamBSD
Goals

- KISS TNC support
- AX.25 networking support
HamBSD

Goals

- KISS TNC support
- AX.25 networking support
- APRS application support
HamBSD

Goals

- KISS TNC support
- AX.25 networking support
- APRS application support
- APRS-IS compatibility
simple framing protocol
four special characters:
  • FEND (frame end)
  • FESC (frame escape)
  • TFEND (transposed frame end)
  • TFESC (transposed frame escape)
frames are sent separated by FEND

if FEND or FESC appear in the frame, instead send a two byte sequence:
- FEND is replaced with FESC TFEND
- FESC is replaced with FESC TFESC
- FEND always means the end of the frame
- FESC always means the start of an escape sequence

---

HamBSD

kiss(4): KISS TNC Support

- the start of every frame is a command
- we will only look at the "data" command, which means the frame contains packet data

---

function sendpacket(packetdata):
    put(FEND)
    put(CMDDATA)
    for byte in packetdata:
        switch (byte):
            case FEND:
                put(FESC)
                put(TFEND)
            case FESC:
                put(FESC)
                put(TFESC)
            default:
                put(byte)
    putc(FEND)
putc(KISSFEND, &tp->t_outq);
putc(KISSCMD_DATA, &tp->t_outq); /* implicitly port 0 */

while (m) {
    register u_char *ep;

    cp = mtod(m, u_char *); ep = cp + m->m_len;
    while (cp < ep) {
        /* Find out how many bytes in the string we can
         * handle without doing something special. */
        register u_char *bp = cp;

        while (cp < ep) {
            switch (*cp++) {
                case KISSFESC: break;
                case KISSFEND: --cp; goto out;
            }
        }
    }
}
```c
out:
if (cp > bp) {
    /*
     * Put n characters at once
     * into the tty output queue.
     */
    if (b_to_q((char *)bp, cp - bp,
                  &tp->t_outq))
        break;
    sc->sc_if.if_obytes += cp - bp;
}
```
/* 
  * If there are characters left in the mbuf,
  * the first one must be special...
  * Put it out in a different form.
  */

if (cp < ep) {
  if (putc(KISSFESC, &tp->t_outq))
    break;
  if (putc(*cp++ == KISSFESC ? KISSTFESC : KISSTFEND, &tp->t_outq)) {
    (void) unputc(&tp->t_outq);
    break;
  }
  sc->sc_if.ifObsolete += 2;
}
m = m_free(m);
}

sending frames
sending frames

receiving frames (slightly trickier, but not much)
HamBSD
kiss(4): KISS TNC Support

- sending frames
- receiving frames (slightly trickier, but not much)
- how to handle frames?
use the kernel networking subsystem
use the kernel networking subsystem
need a network interface: axkiss(4)
use the kernel networking subsystem

need a network interface: axkiss(4)

```
# ifconfig axkiss0
flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 610
     lladdr MM0ROR-6
     index 2 priority 0 llprio 3
     media: KISS TNC (1200 baud)
     status: active
```
### HamBSD

#### ax25(4): AX.25 Networking Support

![Frame Capture]

#### AX.25, Src: MMOROR-7, Dst: UWQWV, Ver: V2.0+

- Current Mic-E Data (not used in TM-D700)
  - Data Type Indicator:
  - Current Mic-E: Lat: 5710.76N Long: 00206.65W, Cse: 730, Spd: 0, SSID: 0, Msg A 1, Msg B 1, Msg C 1, N/S N, Long (degrees): 0x78
  - Longitude minutes: 0x5e
  - Longitude hundredths of minutes: 0x5d
  - Speed (hundreds & tens): 0x6c
  - Speed (tens), Course (hundreds): 0x20

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.76N Lo</td>
</tr>
<tr>
<td>2</td>
<td>180.809960</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.75N Lo</td>
</tr>
<tr>
<td>3</td>
<td>361.557527</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.75N Lo</td>
</tr>
<tr>
<td>4</td>
<td>1084.155021</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.76N Lo</td>
</tr>
<tr>
<td>5</td>
<td>1264.738934</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.76N Lo</td>
</tr>
<tr>
<td>6</td>
<td>1445.436443</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.76N Lo</td>
</tr>
<tr>
<td>7</td>
<td>1626.122482</td>
<td>MMOROR-7</td>
<td>UWQWV</td>
<td>APRS</td>
<td>47</td>
<td>MIC-E Lat: 5710.76N Lo</td>
</tr>
</tbody>
</table>
```
NAME

bpf — Berkeley Packet Filter

SYNOPSIS

pseudo-device bpf

DESCRIPTION

The Berkeley Packet Filter provides a raw interface to data link layers in a protocol-independent fashion. All packets on the network, even those destined for other hosts, are accessible through this mechanism.

The packet filter appears as a character special device, /dev/bpf. After opening the device, the file descriptor must be bound to a specific network interface with the BIOSSET iocll(2). A given interface can be shared between multiple listeners, and the filter underlying each descriptor will see an identical packet stream.

Associated with each open instance of a bpf file is a user-settable packet filter. Whenever a packet is received by an interface, all file descriptors listening on that interface apply their filter. Each descriptor that accepts the packet receives its own copy.

Reads from these files return the next group of packets that have matched the filter. To improve performance, the buffer passed to read must be the same size as the buffers used internally by bpf. This size is returned by the BIOSGET iocll(2) and can be set with BIOSSETLEN. Note that an individual packet larger than this size is necessarily truncated.

A packet can be sent out on the network by writing to a bpf file descriptor. Each descriptor can also have a user-settable filter for controlling the writes. Only packets matching the filter are sent out of the interface. The writes are unbuffered, meaning only one packet can be processed per write.

Once a descriptor is configured, further changes to the configuration can be prevented using the BIOSLOCK iocll(2).
NAME

bpf — Berkeley Packet Filter

SYNOPSIS

pseudo-device bpf nhật

DESCRIPTION

The Berkeley Packet Filter provides a raw interface to data link layers in a protocol-independent fashion. All packets on the network, even those destined for other hosts, are accessible through this mechanism.

The packet filter appears as a character special device, /dev/bpf. After opening the device, the file descriptor must be bound to a specific network interface with the ioctl(2). A given interface can be shared between multiple listeners, and the filter underlying each descriptor will see an identical packet stream.

Associated with each open instance of a bpf file is a user-settable packet filter. Whenever a packet is received by an interface, all file descriptors listening on that interface apply their filter. Each descriptor that accepts the packet receives its own copy.

Reads from these files return the next group of packets that have matched the filter. To improve performance, the buffer passed to read must be the same size as the buffers used internally by bpf. This size is returned by the ioctl(2) and can be set with BIOCSBLEN. Note that an individual packet larger than this size is necessarily truncated.

A packet can be sent out on the network by writing to a bpf file descriptor. Each descriptor can also have a user-settable filter for controlling the writes. Only packets matching the filter are sent out of the interface. The writes are unbuffered, meaning only one packet can be processed per write.

Once a descriptor is configured, further changes to the configuration can be prevented using the ioctl(2).
NAME

bpf — Berkeley Packet Filter

SYNOPSIS

pseudo-device bpf

DESCRIPTION

The Berkeley Packet Filter provides a raw interface to data link layers in a protocol-independent fashion. All packets on the network, even those destined for other hosts, are accessible through this mechanism.

The packet filter appears as a character special device, `/dev/bpf`. After opening the device, the file descriptor must be bound to a specific network interface with the BIOCSETIF `ioctl(2)`. A given interface can be shared between multiple listeners, and the filter underlying each descriptor will see an identical packet stream.

Associated with each open instance of a `bpf` file is a user-settable packet filter. Whenever a packet is received by an interface, all file descriptors listening on that interface apply their filter. Each descriptor that accepts the packet receives its own copy.

Reads from these files return the next group of packets that have matched the filter. To improve performance, the buffer passed to read must be the same size as the buffers used internally by `bpf`. This size is returned by the BIOCGBLEN `ioctl(2)` and can be set with BIOCBLLEN. Note that an individual packet larger than this size is necessarily truncated.

A packet can be sent out on the network by writing to a `bpf` file descriptor. Each descriptor can also have a user-settable filter for controlling the writes. Only packets matching the filter are sent out of the interface. The writes are unbuffered, meaning only one packet can be processed per write.

Once a descriptor is configured, further changes to the configuration can be prevented using the BIOCLOCK `ioctl(2)`.
send position reports
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon

- send position reports
  - OpenBSD already has GPS support
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon

- send position reports
  - OpenBSD already has GPS support
- listen for reports
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon

- send position reports
  - OpenBSD already has GPS support
- listen for reports
- digipeat reports
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon

![Image of APRS tracking interface](image.png)

- MM0ROR-6: center, zoom, info
  - HamBSD aprsd
    - [APBSDD via TCPI*;qAC,T2CSNGRAD]
    - being tracked, stop tracking, track in Street View

---

irl (HamBSD) Amateur kernel networking development 6th December 2019 32 / 44
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon

/etc/aprsd.conf:

gps="nmea0"
beacon position sensor $gps comment "HamBSD aprsd"

# rcctl set aprsd flags -i axkiss0
# rcctl start aprsd
HamBSD
aprsd(8): Automatic Packet Reporting System Daemon
HamBSD

aprsd(8): Automatic Packet Reporting System Daemon
/etc/aprsd.conf:

```
gps="nmea0"
beacon position sensor $gps comment "HamBSD aprsd"

digipeat {
    pass in on axkiss0
    pass out on axkiss0
}
```
using a new axtap(4) network interface
HamBSD

aprsisd(8): APRS-IS Daemon

- using a new axtap(4) network interface
  - complete rip-off of tun(4)/tap(4) driver using the ax25(4) functions instead of Ethernet
HamBSD

aprsisd(8): APRS-IS Daemon

- using a new axtap(4) network interface
  - complete rip-off of tun(4)/tap(4) driver using the ax25(4) functions instead of Ethernet
- make a TCP connection to APRS-IS
HamBSD

aprsisd(8): APRS-IS Daemon

- using a new axtap(4) network interface
  - complete rip-off of tun(4)/tap(4) driver using the ax25(4) functions instead of Ethernet
- make a TCP connection to APRS-IS
  - later this will only support TLS
HamBSD

aprsisd(8): APRS-IS Daemon

- using a new axtap(4) network interface
  - complete rip-off of tun(4)/tap(4) driver using the ax25(4) functions instead of Ethernet
- make a TCP connection to APRS-IS
  - later this will only support TLS
- feeds packets through the axtap(4) interface, like a VPN would
HamBSD

aprsisd(8): APRS-IS Daemon
HamBSD

What next?

- rock-solid digipeating
HamBSD

What next?

- rock-solid digipeating
- recording heard station reports
HamBSD

What next?

- rock-solid digipeating
- recording heard station reports
- privilege separation for parsing
HamBSD
What next?

- rock-solid digipeating
- recording heard station reports
- privilege separation for parsing
- room for innovation
APRS Looking Glass

mb7uar.hambsd.org

show heard

Submit Query

<table>
<thead>
<tr>
<th>Station</th>
<th>Timestamp</th>
<th>Position</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM0ROR-6</td>
<td>2019-11-13</td>
<td>57°10.74N 2°06.63W</td>
<td>Bridge of Don rx-only IGate</td>
</tr>
<tr>
<td>GM4EMX-9</td>
<td>2019-11-13</td>
<td>57°07.45N 2°08.25W</td>
<td>Sysop Kit /m Aberdeen area</td>
</tr>
<tr>
<td>MM0ROR-7</td>
<td>2019-11-12</td>
<td>57°10.74N 2°06.62W</td>
<td>---</td>
</tr>
</tbody>
</table>

aprslg is an APRS looking glass for [HamBSD](https://www.hambsd.org).

Copyright (C) 2019 Iain R. Learmonth.
HamBSD
Beyond APRS

- IPv4 over AX.25
- IPv4 over AX.25
- 6LoWPAN over AX.25
IPv4 over AX.25
6LoWPAN over AX.25
TCP over AX.25
IPv4 over AX.25
6LoWPAN over AX.25
TCP over AX.25
Amateur Wi-Fi
HamBSD
Beyond APRS

- IPv4 over AX.25
- 6LoWPAN over AX.25
- TCP over AX.25
- Amateur Wi-Fi
- Amateur LoRaWAN
IPv4 over AX.25
6LoWPAN over AX.25
TCP over AX.25
Amateur Wi-Fi
Amateur LoRaWAN
Amateur Pagers
IPv4 over AX.25
6LoWPAN over AX.25
TCP over AX.25
Amateur Wi-Fi
Amateur LoRaWAN
Amateur Pagers
Amateur Satellite Comms
HamBSD
How Can I Help?

https://www.patreon.com/fossirl
HamBSD Hardware Wanted

The following list outlines hardware that the HamBSD project could use. There are a number of reasons why developers need hardware, and it is important to first of all realize these reasons:

- To support a specific device, normally one which is more rare or new.
- To be part of their development environment.
- Some of these devices may have quirks that require some investigation and workaround in order to function correctly.
- That said, space or power may be at a premium for some people.

If you do not own these devices, but want to help us, we recommend you search on eBay for the devices. If you do the bidding and then get the device shipped to us, it really helps us. It is better if developers develop, and skip the bidding process, since any time saved can be spent on improving HamBSD instead.

Contact Iain Learmonth who will be able to tell you if something is useful, and if so, where to send it.

Please be very clear as to where you are located!! Unless an item is particularly rare, it is probably uneconomical to ship from anywhere outside Europe. (All development currently happens in Scotland.)

General Requirements

- Terminal Node Controllers (RS232 or USB)
- USB sound card interfaces
- All-mode transceivers (e.g. FT-897, FT-857 and Icom 706)
- RS232 protocol analyser

https://hambsd.org/want.html
Come hang out on IRC
ircs://chat.freenode.net/hambsd
GB1RST
Special Event Station: NorthernRST Hackathon

- 7th December - 8th December
- we’ll be operating a special event station from the NorthernRST hackathon
- we’ll be around on 2 meters and also (hopefully) on HF
- check APRS for up-to-date information on which band(s) we’re on
8th December afternoon

we’re a new club

not sure what we’re doing yet
Thanks

https://hambsd.org/