Open5066 - s5066d

Sampo Kellomäki (sampo@iki.fi)

October 18, 2007

Abstract

Open5066 is an open source implementation of NATO NC3A STANAG 5066 protocol stack for HF radio communications. It aims to implement all of SIS (Annex A), DTS (Annex C), and some of the application layer (Annex F) functionality such as HMTP.

Eventually Open5066 will also implement crypto module and may implement a softmodem for some waveforms such as S-4285 or S-4529, or it may evolve a smooth interface to integrate softmodems from other open source projects.

Concrete embodiment of Open5066 is the s5066d daemon.

1 Status

Project was started in April 2006 and is in very early phase. Major feature enhancements and debugging is going on. Unless you have C debugging skills and are willing to contribute, you probably should not be trying this program yet. But watch this space, in a month or two we expect to have a system that is usable in limited ways.

> Current relase is **0.5**, dated 14.1.2007. This is the first release that is capable of full HMTP over DTS to remote roundtrip. Download from `http://open5066.org/download/open5066-0.5.tgz`

We follow the philosophy of "release early, release often". We are now in the early part.

2 Howto

2.1 Compiling and Installation

At the moment only Linux 2.6 and Solaris 8 are supported. We use *epoll(4)* so porting to other UNIX will require some effort.

Table 1: S-5066 Features

Status	Feature
80%	`I/O` engine, command line interface, and daemon code
75%	SIS Primitives over TCP, parsing and generation
60%	DTS over TCP
0%	DTS over modem
50%	S_PDUs
50%	C_PDUs
40%	D_PDUs
95%	Nonarq transfers, including repetitions
10%	ARQ transfers
0%	Rank and priority support
0%	Expedited anything
0%	Soft link establishement and tear down
0%	Hard link establishement and tear down
0%	Special broadcast mode (as opposed to simply sending nonarq)
0%	Break-in
0%	ALE or similar
0%	Routing
0%	Crypto module
0%	Soft-modem

```
tar xvzf open5066-0.5.tgz
cd open5066-0.5
make
cp s5066d /usr/sbin
```

Solaris 8 using `/dev/poll` can be compiled as follows:

```
make TARGET=sol8
```

Sparc Solaris 8 using `/dev/poll` can be compiled using a gcc cross compiler as follows:

```
PATH=/apps/gcc/sol8/bin:/apps/binutils/sol8/bin:$PATH
make TARGET=xsol8
```

This assumes your cross compiler is in the supplied path. If you change the path, you MUST also change the definition of the SYSROOT variable inside the make file. See gcc documentation for further instructions on creating cross compilers if you are interested in this approach.

2.1.1 Linux-ix86 using tcc-0.9.23

Compilation using TinyCC (see `http://fabrice.bellard.free.fr/tcc/`) is supported.

```
CC=tcc LD=tcc make -e
```

While it will compile all right, it seems it does not quite run allright :-(

2.2 Running

You can pass mail between two nodes by doing following

On node 1:

```
echo secret | s5066d -p sis::5066 -p dts::5067 -c AES256 -k 0 dts:node2:5067
s5066d -p smtp::25 sis:localhost:5066 smtp:mail.yourdom.com:25
```

On node 2:

```
echo secret | s5066d -p sis::5066 -p dts::5067 -c AES256 -k 0 dts:node1:5067
s5066d -p smtp::25 sis:localhost:5066 smtp:mail.theirdom.com:25
```

Next you need to configure the mail servers `mail.yourdom.com` and `mail.theirdom.com` to forward mail to node1 and node2, respectively. Probably the best way to do this is to pick some special subdomain and forward all mail to this subdomain to the s5066 pipe. For example mail destined to `her@hmtp.theirdom.com` could be funneled to port 25 on node 1 and emerge from node2 where the local daemon there will talk to the SMTP server and try to deliver her mail.

```
NODE 1                                          NODE 2

Mail Client A                                   Mail Client B
    |                                               ^
    V                                               |
mail server                                      mail server
    |                                               ^
    V                                               |
 hmtpgwd --s--> s5066d ---z---> s5066d --s--> hmtpgwd

s = SIS primitives (Annex A)
z = "over the air" (currently TCP) DTS traffic (Annex C)
hmtpgwd = specially configured s5066d (the second command line on each node)
```

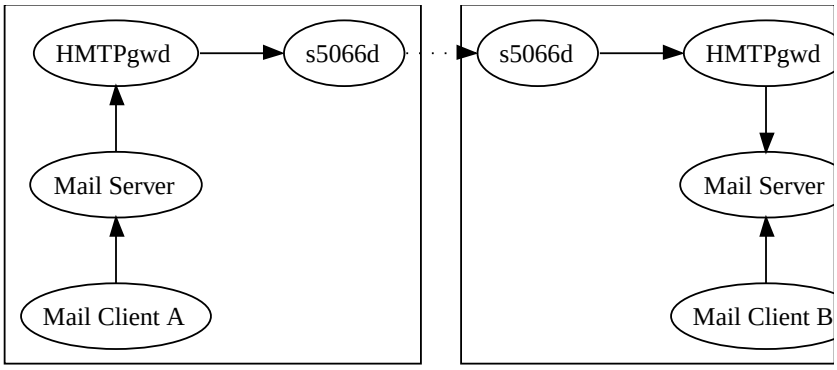

Figure 1: Connecting regular SMTP servers via HMTP gateways and S5066 network.

Needless to say, you will need to be pretty good at your mail server configuration to pull this off. Please do not ask me how to configure them: consult your mail server documentation.

3 s5066d - The STANAG 5066 Daemon Documentation

3.1 Invocation

```
Usage: s5066d [options] PROTO:REMOTEHOST:PORT
       echo secret | s5066d -p sis::5066 -c AES256 -k 0 dts:quebec.cellmail.com:5067
       s5066d -p smtp::25 sis:localhost:5066 smtp:mail.cellmail.com:25
  -p  PR:IF:PORT   Protocol, network interface and TCP port for listening
                   connections. Default sis:0.0.0.0:5066. You can omit interface.
                     sis:0.0.0.0:5066   - Listen for SIS (Annex A primitives)
                     dts:0.0.0.0:5067   - Listen for DTS (Annex B)
                     smtp:0.0.0.0:25    - Listen for SMTP (RFC 2821)
                     http:0.0.0.0:80    - Listen for HTTP/1.0 (simplified)
                     tp:0.0.0.0:5068    - Listen for test ping protocol
  -t  SECONDS      Connection timeout for both SIS and DTS. Default: 0=no timeout.
  -c  CIPHER       Enable crypto on DTS interface using specified cipher. Use '?' for list.
  -k  FDNUMBER     File descriptor for reading symmetric key. Use 0 for stdin.
  -nfd  NUMBER     Maximum number of file descriptors, i.e. simultaneous
                   connections. Default 20 (about 16 connections).
  -npdu NUMBER     Maximum number of simultaneously active PDUs. Default 60.
  -nthr NUMBER     Number of threads. Default 1. Should not exceed number of CPUs.
  -nkbuf BYTES     Size of kernel buffers. Default is not to change kernel buffer size.
  -nlisten NUMBER  Listen backlog size. Default 128.
  -egd PATH        Specify path of Entropy Gathering Daemon socket, default on
                   Solaris: /tmp/entropy. On Linux /dev/urandom is used instead
                   See http://www.lothar.com/tech/crypto/ or
                   http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/prngd.html
  -rand PATH       Location of random number seed file. On Solaris EGD is used.
                   On Linux the default is /dev/urandom. See RFC1750.
  -uid UID:GID     If run as root, drop privileges and assume specified uid and gid.
  -pid PATH        Write process id in the supplied path
  -watchdog        Enable built-in watch dog
  -kidpid PATH     Write process id of the child of watchdog in the supplied path
  -afr size_MB     Turn on Application Flight Recorder. size_MB is per thread buffer.
  -v               Verbose messages.
  -q               Be extra quiet.
  -d               Turn on debugging.
  -license         Show licensing details, including NATO C3 Agency disclaimer.
  -h               This help message
  --               End of options
N.B. Although s5066d is a 'daemon', it does not daemonize itself. You can always say s5066d\
```

Mostly the daemon acts as an I/O engine or proxy. You will need to specify listening port with -p option if you want to contract the daemon or if you want anyone else to contact the daemon. If you want the daemon to contact anyone else, you will need to

specify the servers to contact on the command line. Each -p or commandline specification follows format

```
PROTOCOL:HOST:PORT
```

The PROTOCOL is one of

- sis - S5066 Annex A SIS primitives are passed over TCP stream
- dts - S5066 Annex C DTS D_PDUs are passed over TCP stream
- smtp - Simple Mail Transfer Protocol (SMTP, RFC2821) is spoken over TCP connection
- http - `HTTP/1.0` is spoken over TCP. Note that this support is very limited and only used for debugging and benchmarking the `I/O` engine. *This is NOT a real web server.*
- tp - Test Ping Protocol. Used for debugging the `I/O` engine.

For listening sockets the HOST specifies which network interface the listener will bind to. This is only relevant for multihomed hosts and generally you will know what this means if you need it. For everybody else, specify `0.0.0.0` or just omit the host and the port will be bound to all interfaces.

For remote servers the HOST specifies the domain name or IP address of the remote server. You can also specify `localhost` or `127.0.0.1` if you want.

The PORT specifies either the TCP port to listen to or the TCP port to contact. Typical ports to use

- 5066 - SIS primitives over TCP (Annex A)
- 5067 - DTS D_PDUs over TCP (Annex C)
- 5068 - Test ping
- 25 - SMTP (you will need to run s5066d as root, but see -uid UID:GID flag)
- 80 - HTTP (you will need to run s5066d as root, but see -uid UID:GID flag)
- 8080 - HTTP if you do not want to run as root

3.2 SIS Server

Main function of s5066d is to relay SIS primitives from a TCP socket to DTS connection (which also may be TCP, but will eventually be HF Modem connection). To accomplish this you have to make sure s5066d listens for SIS primitives. You also need to specify the remote DTS node (usually another s5066d) to contact. It is also useful for the s5066d to listen for new DTS traffic from other nodes. All of this can be accomplished with the following command line:

```
s5066d -p sis::5066 -p dts::5067 dts:remote.theirdom.com:5067
```

3.3 HMTP Gateway Node

High Frequecy Mail Transfer Protocol (Annex F) gateway can be created by configuring s5006d to listen to SMTP (Simple Mail Transfer Protocol, RFC2821) and then contact the SIS

```
s5066d -p smtp::25 sis:localhost:5066 smtp:mail.mydomain.com:25
```

3.4 Testing

Effectively create back-to-back gateway that talks to itself:

```
./s5066d -p smtp:0.0.0.0:2525 -p sis:0.0.0.0:5066 sis:localhost:5066 smtp:ilpo:25
perl hitest.pl 1
```

- This actually CAN NOT work because we send out UNIDATA_REQ, but expect to receive UNIDATA_IND and the two are not sufficiently compatible.

Full test setup

```
./s5066d -di Bsis -p sis:0.0.0.0:5065 -p dts:0.0.0.0:5067
./s5066d -di Asis -p sis:0.0.0.0:5066 dts:localhost:5067
./s5066d -di Asmtp -p smtp:0.0.0.0:2525 sis:localhost:5066   # SMTP server
./s5066d -di Bsmtp sis:localhost:5065 smtp:ilpo:25           # SMTP client
perl hitest.pl 1
```

See: `./setup1.sh` for a way to lauch all necessary servers.

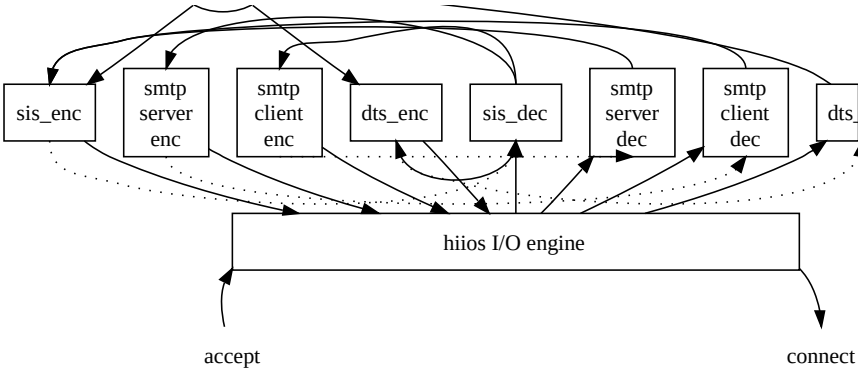

Figure 2: Module structure of s5066d

4 Architecture

4.1 Connection Management

The hiios engine is used. Every connection (io object) MUST be associated with a file descriptor. The memory for the connection is obtained from an array indexed by the file descriptor. Since the file descriptor is used as a mutual exclusion mechanism for the connection memory array, you must not close a file descriptor until you are fully done with all the memory associated with it.

4.2 PDU Management

Allocate from preallocated pools.

Reference counted? Garbage collected?

4.3 todo_queue

1. polling inserts the io objects that are eligble for `I/O`
2. also PDUs can be inserted if they require further processing
3. worker threads eat from the todo_queue and dispatch

4.4 Detecting that `I/O` is possible

- *epoll(4)* on Linux 2.6

  - edge triggered events
  - requires every `I/O` to be "exhausted" until it returns EAGAIN

- `/dev/poll` on Solaris 8 and newer
  - To Be Investigated

4.5 Some Open Issues

- Remote management
  - remotes are given on command line, but should not really be opened immediately
  - depends on protocol: DTS immediate, SMTP only upon SIS HMTP traffic
  - closing a remote, e.g. SMTP after message
  - reconnect after failure?
  - multiple remotes to choose from
- Handling sidebars

5 Protocol Processing

5.1 Protocol TODO

- Segmentation of HMTP at SIS interface (currently entire HMTP message has to fit in one S_PDU wihout segmenting)
- more SIS primitives
- soft link establishment
- arq mode
- hard link establishment
- expedited modes

5.2 DTS

5.3 SIS

5.4 SMTP Client Processing

When a HMTP connection arrives from SIS layer, a connection is established to destination SMTP server which must have been specified on the command line. The SMTP server will send responses which are processed by *smtp_resp_data()* as they come. This processing updates the `smtp_state` variable on the SMTP connection object.

The problem is "feeding" the HMTP payload piecemeal to the SMTP server. We can not just blast the whole blob to the server because that would be a violation of the SMTP protocol, pipelining or not. The `smtp_state` effectively acts as a synchronization mechanism between the feeder and the response processor. This mechanism only supports pipelined SMTP [RFC2197]

Table 2: Sequence of pipelined SMTP states for client

Value	Symbol	Who	Action
0	INIT	resp	Wait to receive 220 greeting. Can not send HMTP yet.
1	EHLO	feeder	Send SMTP EHLO. Feeder will do this as soon as HMTP message arrives from SIS layer.
2	RDY	resp	Wait to receive 250 from EHLO. Can not send HMTP yet.
3	MAIL	feeder	Send HMTP payload, except for message and QUIT. As mail can be big, several SIS primitives may be consumed.
4	RCPT	feeder	Continue sending SMTP payload except QUIT. Internal state of feeder.
5	DATA	resp	Process 250 responses from MAIL FROM and RCPT to commands until 354 enter mail is seen
6	SEND	feeder	Send message. As mail can be big, several SIS primitives may be consumed.
7	SENT	resp	Expect to see 250 from message input. Reply to SIS layer by sending HMTP response (synthesizing 221 goodbye response). Reply to SMTP server with QUIT.
8	QUIT	resp	Expect to see 221 goodbye from Quit. Close SMTP server connection.

The tricky part is that the HMTP pdu will originally trigger EHLO, but can not be sent immediately. What we do is schedule the HMTP pdu to todo_queue only once we have seen the 250 response. Eventually the processing will resume.

A slight problem remains in how to prevent reading further PDUs from the SIS connection? This is solved by NOT scheduling the SIS connection to the todo until we are ready to receive something from it. We also set the need more data field to zero to block it from reading further input in case there is a spurious `I/O` event reported.

Current implementation only supports one SIS source and one SMTP destination, both of which are supplied on command line. Supplying more than one of each will produce unpredicatble results.

5.5 SMTP Server Processing

When a SMTP connection is accepted from another SMTP server, we need to play the pleasantries (EHLO, etc.) until we get to the point of the payload that starts with MAIL FROM command.

Until and including when DATA is received, we keep receiving all the protocol data into cur_pdu, playing a state machine that will send the right responses.

Once the DATA has been received we formulate HMTP PDU and send it using SIS layer. As an important optimization, we look beyoud DATA and add as much as makes sense to the SIS primitive. This could mean we see the dot on its own line that terminates the message. In that case we add our own QUIT and the HMTP PDU is complete.

It could also happen that the mail message is long and all we can see is in middle of some data, or worse something like "˚.̊" (but not the final ""). If so, we record the state of end of visible data in `smtp_state` variable and just send the data with SIS primitive. For long mail there can be any number of SIS primitives that just contain data and no protocol information.

Finally we will see "˚.˚" or deduce its existence from the `smtp_state`. We add our own QUIT and ship the last fragment with SIS (What if QUIT does not fit? No problem, we can use writev).

The SIS receiving process eventually gets HMTP response PDU. We scan the HMTP message until we see 354 after which the next response code is expected to contain the outcome of the transaction.

We send the outcome back to the SMTP server (usually "250 message sent") and move the protocol to state where either QUIT or new MAIL FROM can be received. In case of former we send "221 Bye" response and close the connection. In case of the latter, we start the processing from the beginning, sans the EHLO.

6 Simple Routing

s5066d implements[1] routing by decoding DTS packets and using destination node address to look up a line from a routing table. The line can say:

[1] Release 0.1 does not implement routing yet.

Table 3: Sequence of pipelined SMTP states for server

Value	Symbol	Who	Action
0	INIT	write	Send 220 greeting
11	START	read	Wait for EHLO. Send 250 response.
12	MAIN	read	Wait for MAIL FROM and rest of payload (we are using pipelining) and ship it as SIS primitive.
13	TO	read	Wait for RCPT TO or DATA. Substate of main.
13	MORE0	read	For long mails, continue shipping the mail message as SIS primitives. Work this way until end of message is detected (sole dot on a line). Asseme no precursor of "°.°" has been seen.
14	MORE1	read	Assume "°" has been seen and ".°" needs to be seen.
15	MORE2	read	Assume "°." has been seen and "°" needs to be seen.
16	WAIT	sisread	Expect to see 354 response in HMTP response.
17	STATUS	sisread	Expect to see 250 indicating the success of message. Send the status to SMTP, but do not send 221 bye even if one was part of HMTP message.
18	END	read	Wait for QUIT (which probably was already in the buffer) and send 221 bye. If we get MAIL FROM instead, continue processing at state MAIN (12).

- ignore
- my local address, process locally
- my group address, process locally
- route to other DTS interface

The last form makes it possible to route traffic from one subnet to another provided that the node is able to participate on both subnets and knows which subnet is closer to the ultimate destination. Following hierarchy of the network is is easy to establish a default route.

The routing capability can also be used to implement a repeater within the same subnetwork.

In the above diagram note that L2 can only reach the rest of the network via L1, which is configured as a repeater for L2 traffic. While MAD may be hearable within all of the CS Liberia territory, most of the actual Outposts are not hearable by MAD, thus two way communication MUST pass through CS Liberia, or via L1 in case L2 wants to communicate.

If J1 wants to communicate with P1, it sends a message that is detected by CS Java as message requiring routing. CS Java uses the long haul network and SIN notices it and also determines it as routable. Effectively SIN will act as a repeater and send the

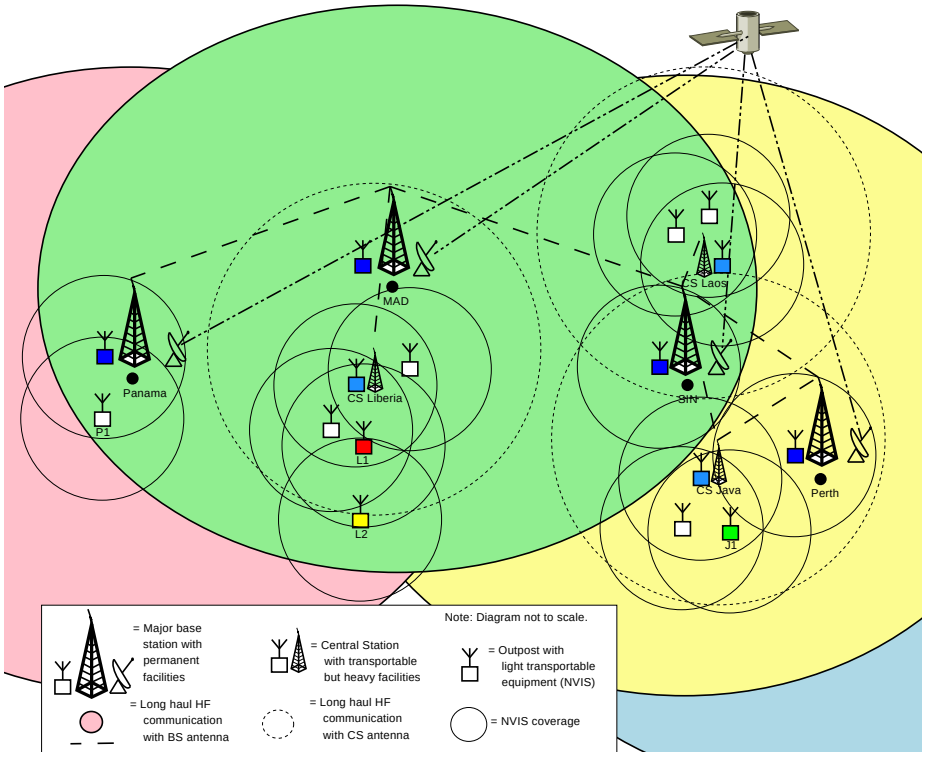

Figure 3: Global HF network connects to regional networks that use NVIS propagation. Some global bandwidth is provided by HF.

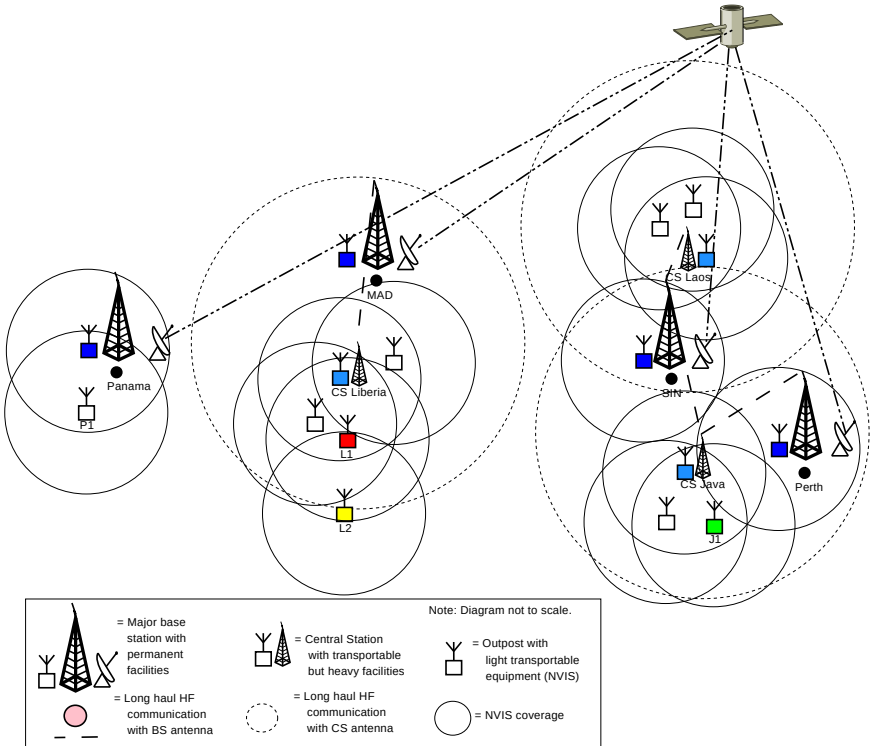

Figure 4: Global HF network connects to regional networks that use NVIS propagation. All global bandwidth is via satellite.

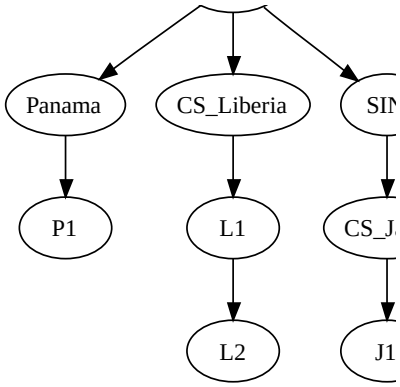

Figure 5: Routing Scenario

message with enough power for MAD to hear it (incidentally, this power is enough to reach J1 as well, but that does not help because it is one way communication). MAD determines that the message needs to be repeated so that Panama can hear it. Finally Panama figures that P1 is within local NVIS coverage and sends it using that channel.

> The s5066d routing capability, despite being very simple, is very powerful and allows any node on the globe to reach any other node.

7 Resources

S5066 site `https://elayne.nc3a.nato.int/` (needs password for most interesting stuff)

Project home `http://open5066.org/`

Download `http://open5066.org/open5066-0.4.tgz`

Freshmeat page `http://freshmeat.net/projects/open5066`

Mailing list none yet, though announcements will happen on `freshmeat.net`

Reporting bugs no bug tracking yet, mail author

CVS access None. If you contribute a lot, I can create you nonanonymous access.

This document as PDF `http://open5066.org/open5066.pdf`

This document as HTML `http://open5066.org/open5066-frame.html`

8 S-5066 Doubts

If someone could answer following questions, or improve the S-5066 spec, I would be grateful.

1. Meaning of confirmation modes "node" and "client"? Local node or remote node? Remote client?
2. Redundancy of ARQ mechanism wrt "confirmations"? Especially node confirm would seem to me to be the same as ARQ.
3. How is HMTP traffic segmented to U_PDUs given that a mail can be much bigger than the U_PDU maximum limit?
4. TTD Julian date? Exactly how is this formatted? In seconds? Since what epoch? Any modular arithmetic invoved?

5. Would destination sap id and source sap id ever differ?

6. The SMTP - HMTP comparison table, Table F-2, on p. F-10 would seem to violate RFC2920 section 3.1 "Client use of pipelining" third paragraph. The actual message can not be pipelined with DATA. However it can be pipelined with QUIT.

9 License and Disclaimers

You may use this code under the terms of Gnu General Public License, version 2 (GPL 2).

Given early phase of the project, I am open to justified suggestions about the licensing model.

> IMPORTANT: If you use this code for actual radio emissions, beware that this has neither been debugged yet nor validated, let alone certified to be conformant with anything. If you perform radio emissions YOU, and not me or my contributors, are responsible for any bad signal or protocol emitted, interference caused, horrible consequences triggered, or for starting a nuclear war. I expressly do NOT warrant that this code, in its present state, is correct. Of course this disclaimer is redundant as GPL disclaims all warranties anyway and requires you to waive all responsibility on my behalf or on behalf of my contributors.
>
> That said, if you are willing to validate, certify, or sponsor a certification, this type of help would be very welcome. Even just informal interoperability testing would be very valuable.

Another point to remember, I am working under assumption that STANAG 5066 is patent free and I have not done, neither do I plan to do, patent search. Obviously I will not indemnify for any IPR breach, your sole remedy is to cease using this software. If this worries you and you do a patent review, or you are aware of patent claims, please let me know (with specific patent numbers so I can look them up myself).

This product includes software developed by the NATO C3 Agency The Hague the Netherlands and its contributors. Such software is subject to following copyright and terms

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the NATO C3 Agency The Hague the Netherlands and its contributors.

4. Neither the name of the NATO C3 Agency nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior permission.

References

[RFC2821] SMTP

[RFC2197] SMTP pipelining extension