Hamlib, Rigserve & Open Source

Martin Ewing, AA6E
Branford CT

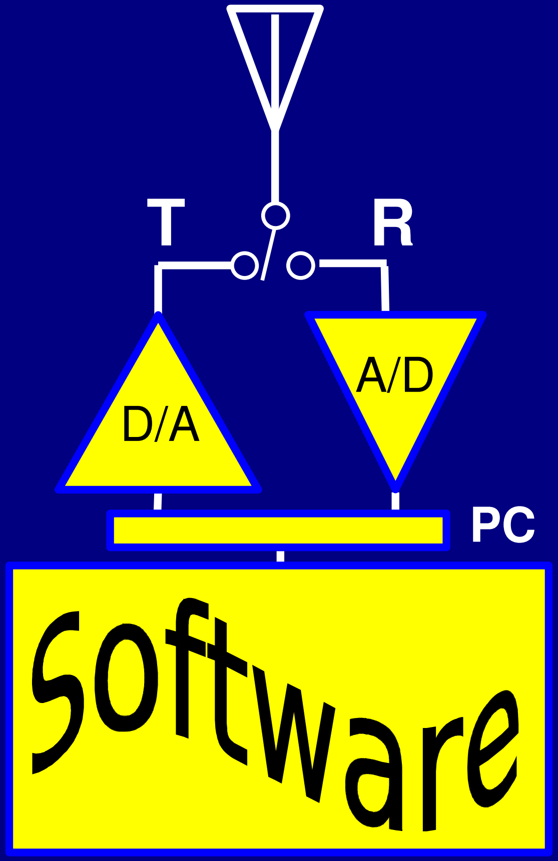
DCC 2007

Introduction

- Preliminaries
  - D.C. ⇔ Software in Ham Radio?
- Amateur Radio v. Amateur Software
- Rig Control
  - Easy?
- Hams & OSS Development

Amateur {Radio | Software}

- Hams:
  - some are pros
  - some are appliance ops
  - the rest of us are ...
- Builders and tinkerers?

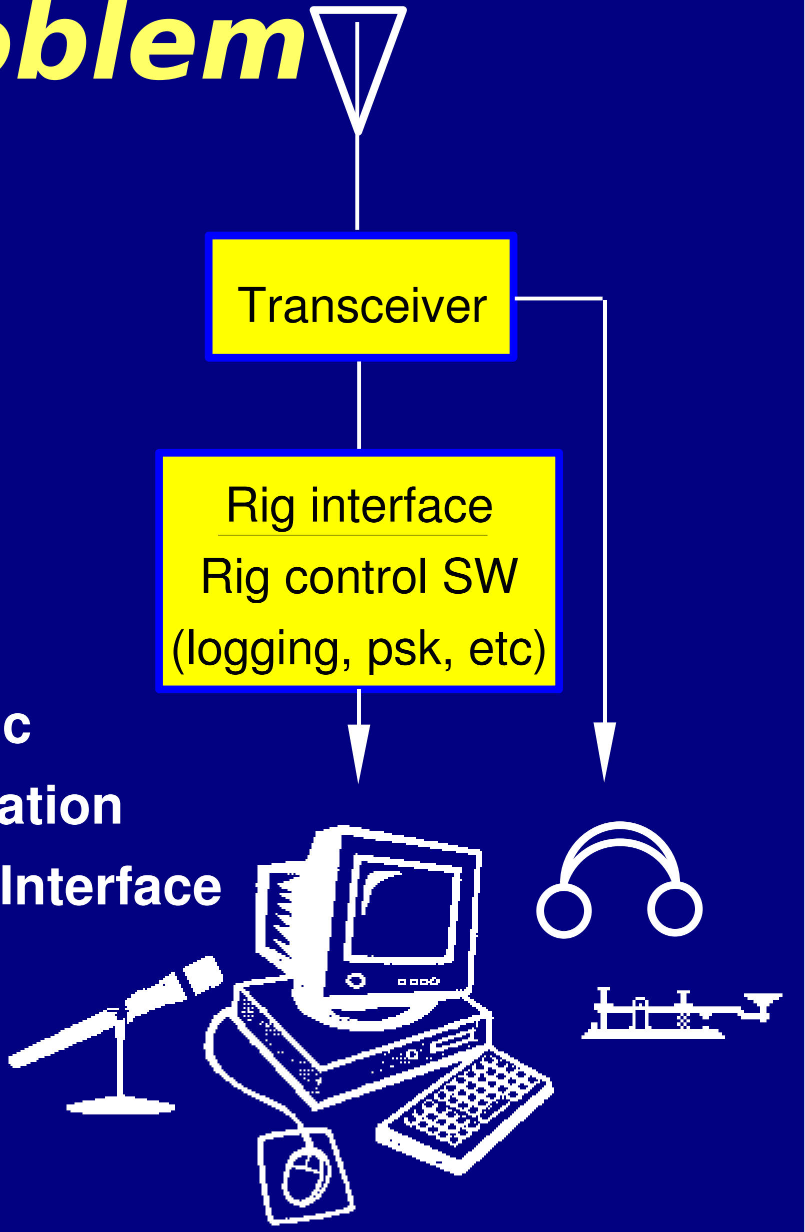


Operator optional --

An Ideal Ham Station?

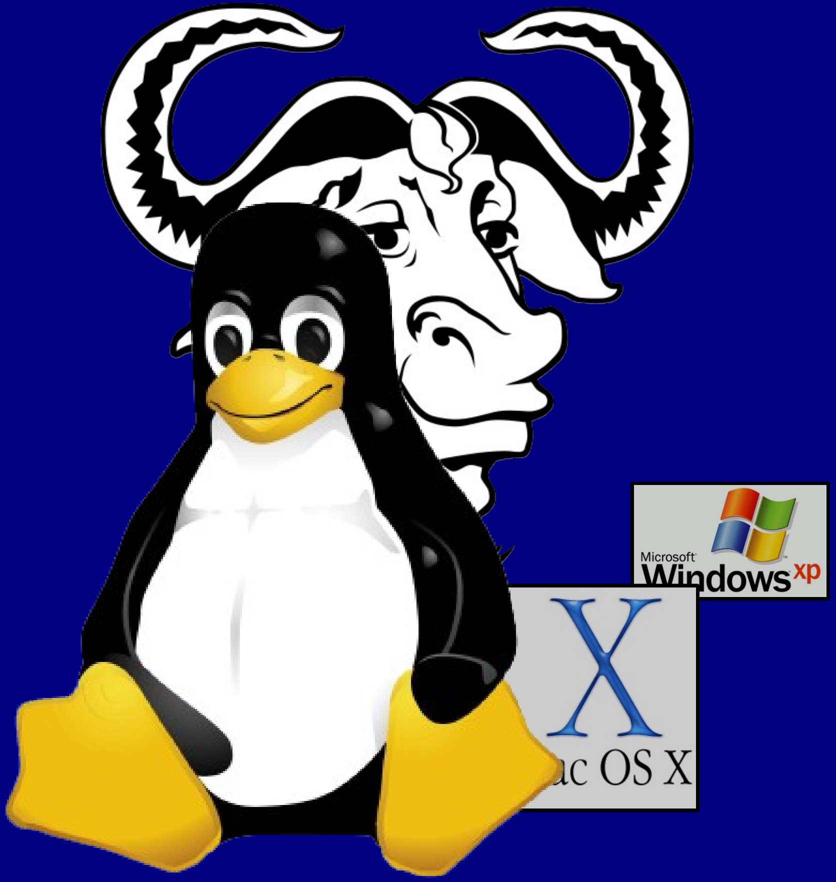
The Rig Control Problem

- A universal rig control method?
  - Standard “API” for all rigs
- **Hamlib** – a C library, since 2000
- **Rigserve** – an experimental server.



Hams & OSS Development

- Open Source SW: “ham-like”?
- Licensing?
- Profit or fame??
- What tools for “fame”?
- And they better be cheap!



Tux and the GNU, etc.

Three Short Stories...

- Hamlib
  - Since 2000
  - C-based library
- Rigserve
  - New, improved (?)
  - Python-based network server
- Considering OSS

The Rig Control Problem

- **What is Rig Control?**
  - **Running stuff from an ext. computer**
- **So many rigs, so little time!**
- **Two approaches:**
  - **Library (API) – Hamlib**
  - **Server - Rigserve**

Hamlib

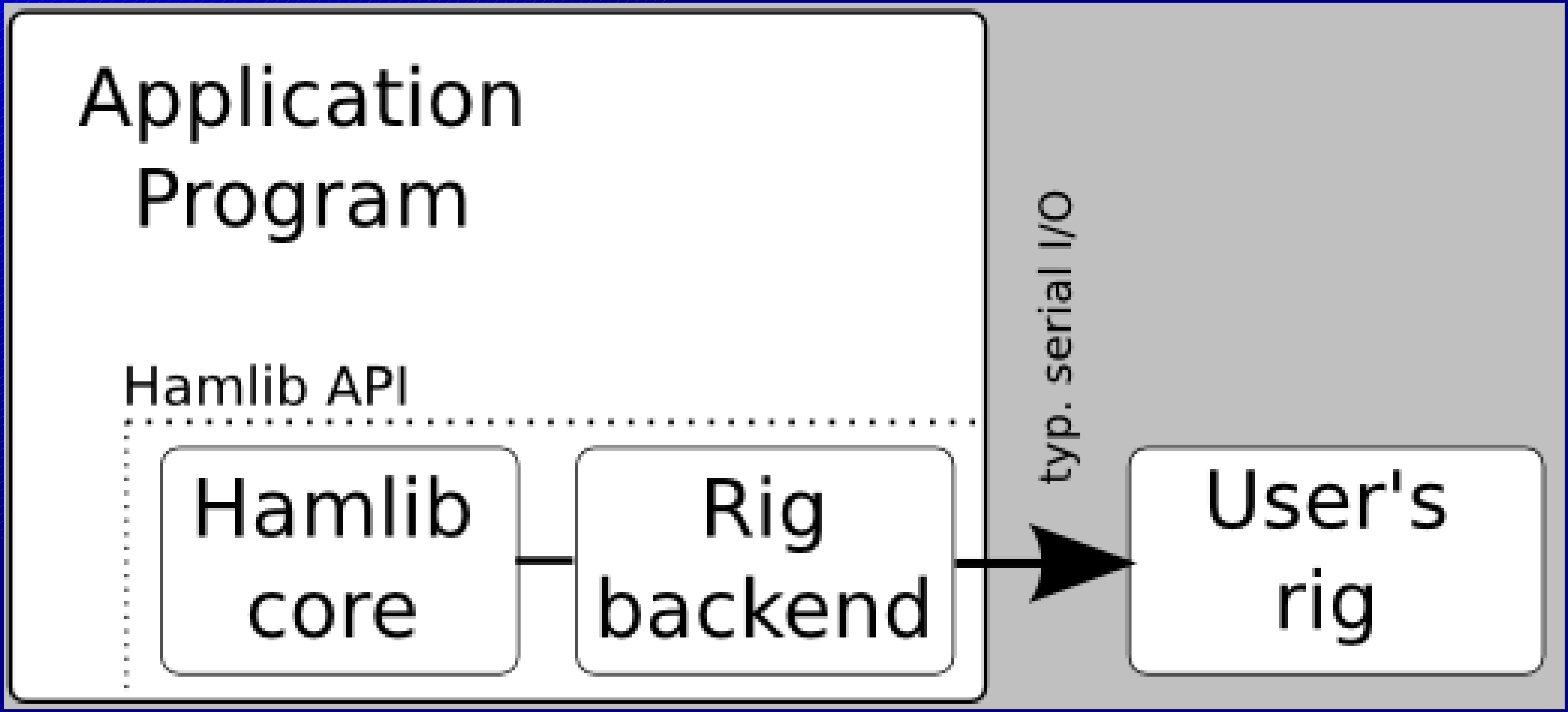
- Target: application developers
  - Not end-users!
- A C library
  - for C, C++, (Python, PERL, & TCL)
- Standardized “front-end” API
- Backends for many rigs

Hamlib: Vital Statistics

- Started 2000 by
  Frank Singleton VK3FAS/KM5WS
  and Stéphane Fillod F8CFE.
- Coverage of 140 rigs
- 188 K lines of code, 767 files
- Some 30 developers / testers

Hamlib: How it Works

- **Hamlib is a library, linked to app.**



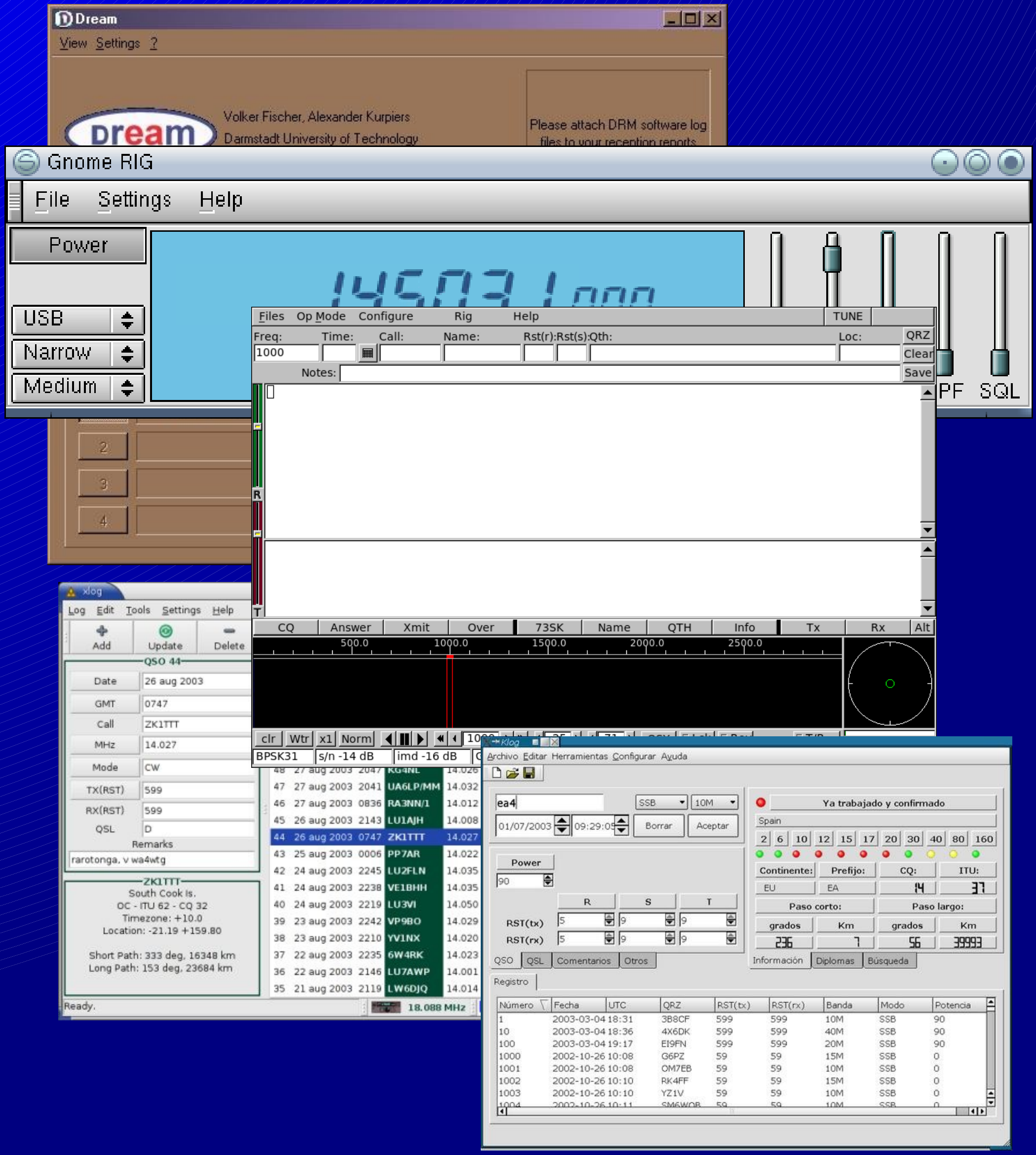
Hamlib: How it Works

- **User code links to Hamlib API**
  - **API ⇔ application programming interface ⇔ subroutine calls, data types, constants, ...**
- **“Backends” translate API to each rigs' own commands.**

Hamlib Successes

- Rig control API
- Modular framework
- Multi-platform (Linux, Win, etc)
- Multi-lingual:
  - C, C++, Python, PERL, TCL
- Transparent & enthusiastic project
- See www.hamlib.org, v 1.2.6

Hamlib Adoptions


Dream
View Settings ?
Volker Fischer, Alexander Kurpiers
Darmstadt University of Technology
Please attach DRM software log files to your reception reports
Gnome RIG
File Settings Help
Power
USB
Narrow
Medium
PF SQL
Files Op Mode Configure Rig Help TUNE
Freq: Time: Call: Name: Rst(r):Rst(s):Qth: Loc: QRZ Clear Save
Notes:
CQ Answer Xmit Over 73SK Name QTH Info Tx Rx Alt
clr Wtr x1 Norm
xlog
Log Edit Tools Settings Help
Add Update Delete
Ready.
18.088 MHz
Klog
Archivo Editar Herramientas Configurar Ayuda
Borrar Aceptar
Power
QSO QSL Comentarios Otros
Información Diplomas Búsqueda
Registro

Freq: 1000

BPSK31 | s/n -14 dB | imd -16 dB

QSO 44

Date	26 aug 2003
GMT	0747
Call	ZK1TTT
MHz	14.027
Mode	CW
TX(RST)	599
RX(RST)	599
QSL	D

Remarks: rarotonga, v wa4wtg

ZK1TTT
South Cook Is.
OC - ITU 62 - CQ 32
Timezone: +10.0
Location: -21.19 +159.80

Short Path: 333 deg, 16348 km
Long Path: 153 deg, 23684 km

48	27 aug 2003	2047	KG4NL	14.026
47	27 aug 2003	2041	UA6LP/MM	14.032
46	27 aug 2003	0836	RA3NN/1	14.012
45	26 aug 2003	2143	LU1AJH	14.008
44	26 aug 2003	0747	ZK1TTT	14.027
43	25 aug 2003	0006	PP7AR	14.022
42	24 aug 2003	2245	LU2FLN	14.035
41	24 aug 2003	2238	VE1BHH	14.035
40	24 aug 2003	2219	LU3VI	14.050
39	23 aug 2003	2242	VP9BO	14.029
38	23 aug 2003	2210	YV1NX	14.020
37	22 aug 2003	2235	6W4RK	14.023
36	22 aug 2003	2146	LU7AWP	14.001
35	21 aug 2003	2119	LW6DJQ	14.014

ea4 | SSB | 10M
01/07/2003 | 09:29:05

Power: 90

	R	S	T
RST(tx)	5	9	9
RST(rx)	5	9	9

Ya trabajado y confirmado
Spain

2	6	10	12	15	17	20	30	40	80	160
🟢	🟢	🔴	🔴	🔴	🔴	🔴	🟢	🟡	🟡	🟢

Continente:	Prefijo:	CQ:	ITU:
EU	EA	14	37

Paso corto:		Paso largo:	
grados	Km	grados	Km
236	7	56	39993

Número	Fecha	UTC	QRZ	RST(tx)	RST(rx)	Banda	Modo	Potencia
1	2003-03-04	18:31	3B8CF	599	599	10M	SSB	90
10	2003-03-04	18:36	4X6DK	599	599	40M	SSB	90
100	2003-03-04	19:17	EI9FN	599	599	20M	SSB	90
1000	2002-10-26	10:08	G6PZ	59	59	15M	SSB	0
1001	2002-10-26	10:08	OM7EB	59	59	10M	SSB	0
1002	2002-10-26	10:10	RK4FF	59	59	15M	SSB	0
1003	2002-10-26	10:10	YZ1V	59	59	10M	SSB	0
1004	2002-10-26	10:11	SM6WOB	59	59	10M	SSB	0

Dream	**Grig**
Xlog	**Fldigi**
SGControl	**GMFSK**
TLF	**Xdx**
ql	**Klog**
PSKmail	**Ktrack**

Hamlib: Challenges

- Libraries need to be linked
  - Language & platform specific
  - N rigs/ 1 app and N apps/ 1 rig?
- C is widely known, but
  - Low-level, simple typing, low “SNR”, little error checking
- Internal issues

Hamlib: Challenges, contd.

- **Priority: backend developers**
  - **Learning curve**
  - **Clear philosophy**
    - **least common denominator?**
  - **Documentation of Internals**
- **SWIG (swig.org): blessing or curse?**
- **Defining v2…**

Other "Universal" Rig Interfaces

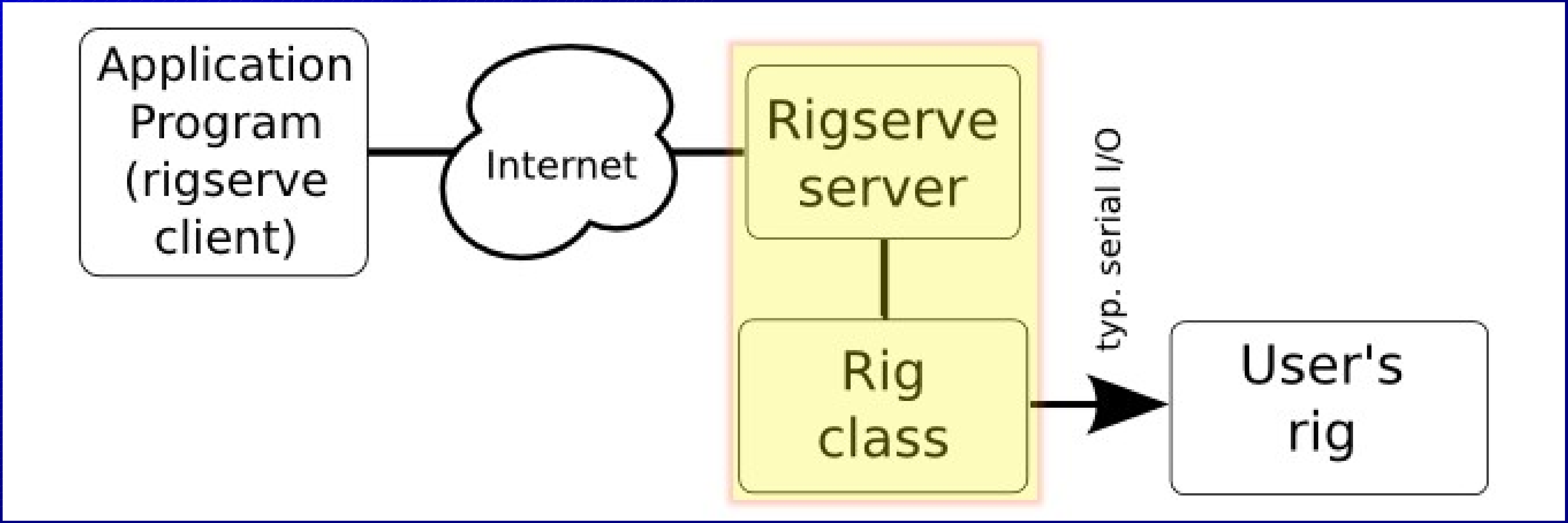
- **Ham Radio Deluxe**
  - HB9DRV, hrd.ham-radio.ch
  - Free noncommercial, but proprietary
  - Windows only, GUI+TCP/IP interface
- **rigCAT**
  - W1HKJ, w1hkj.com/xmlarchives.html
  - Rig control as a "data problem"
  - XML is verbose, ±human readable
- **And** ...

Something completely different?

- **Rigserve** (rigserve.sourceforge.net)
  - a response to Hamlib challenges
  - Completely incompatible...
  - Status: Small project, lots of potential...

Rigserve Overview

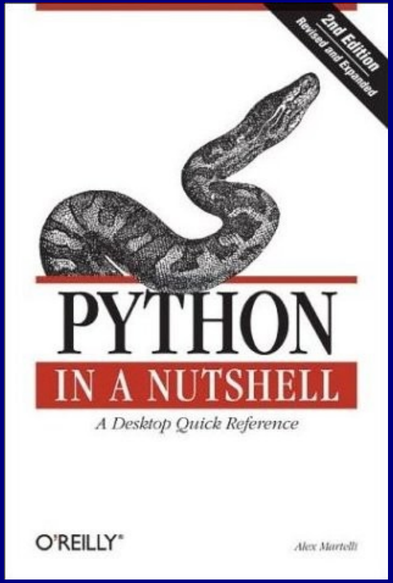
- **Client-Server design**



Rigserve Philosophy

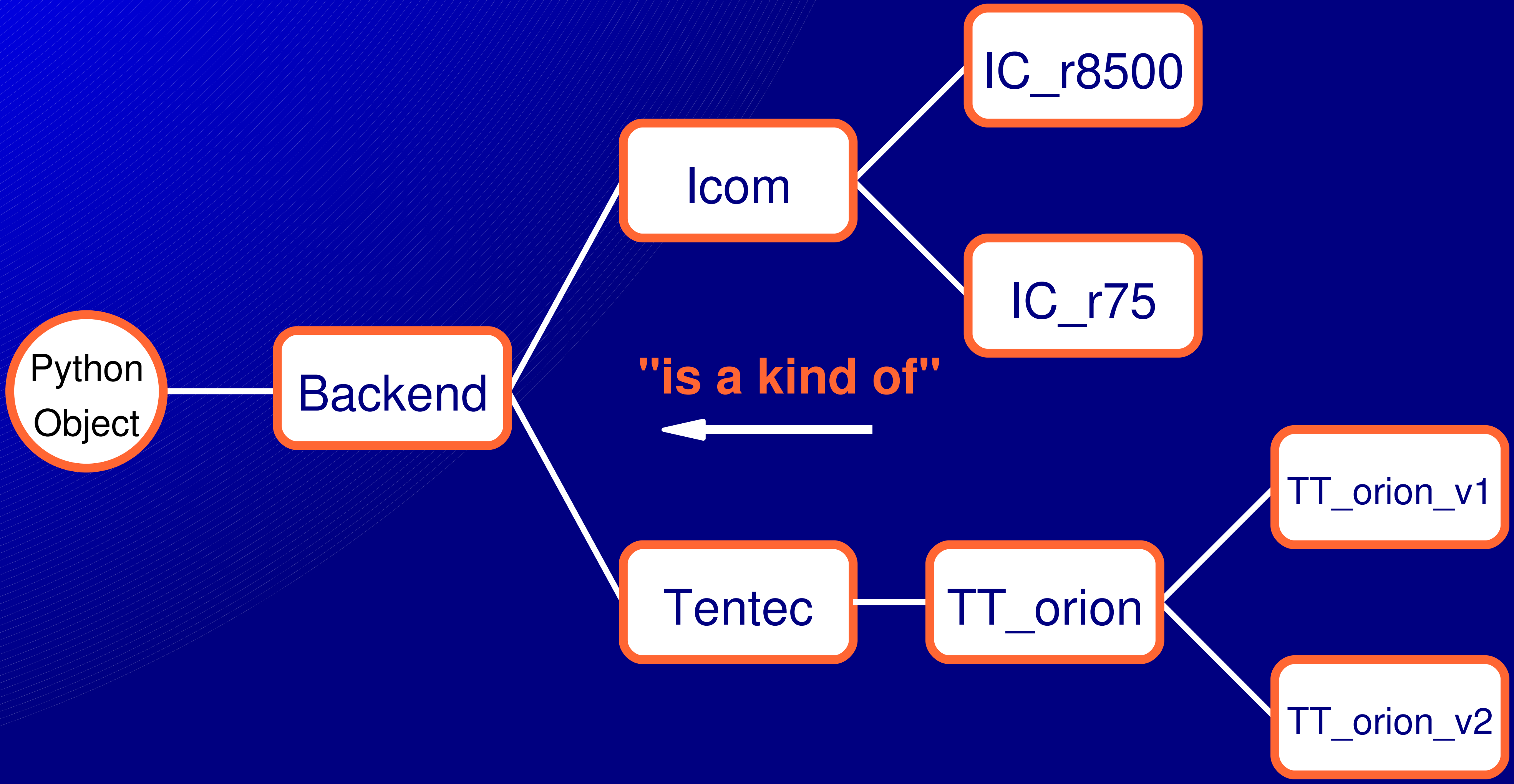
- **Server via TCP/IP (local or remote)**
  - **Telnet-compatible sessions**
  - **Human-readable transactions**
  - **Language- and platform-independent for clients**
- **OO implementation, HL language**
  - **Easy to learn, document (relatively!)**

Rigserve Implementation

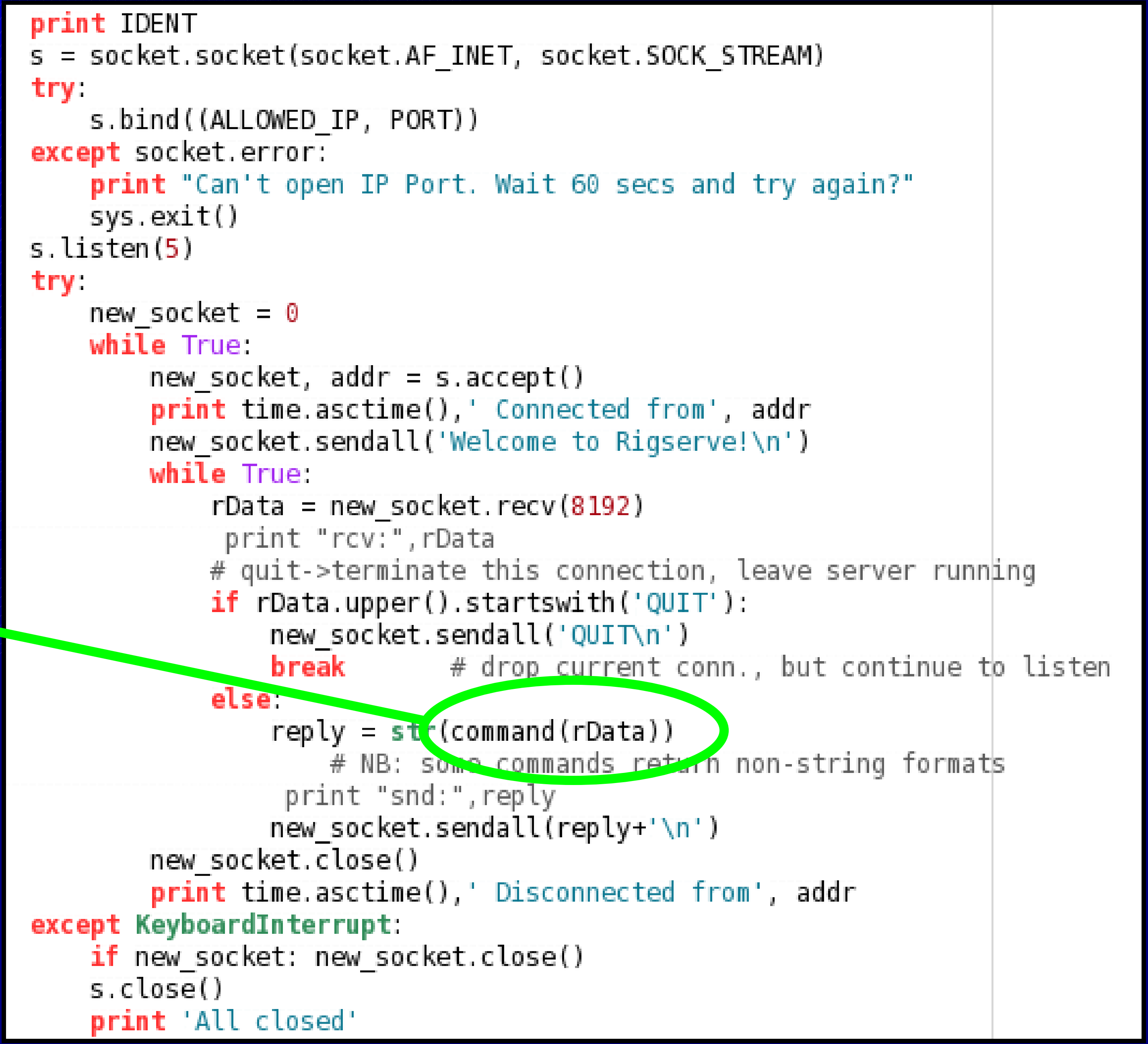


- Using Python (python.org)
  - High-level, strongly typed
  - Object oriented
  - Linux, Windows, MacOS, …
  - Big module library
  - Fast compile: 0.5 sec. vs 540 sec
- Rigs as objects
  - Both data and behavior.
  - Class hierarchy – families of rigs
  - Clarity…

Rig Class Hierarchy



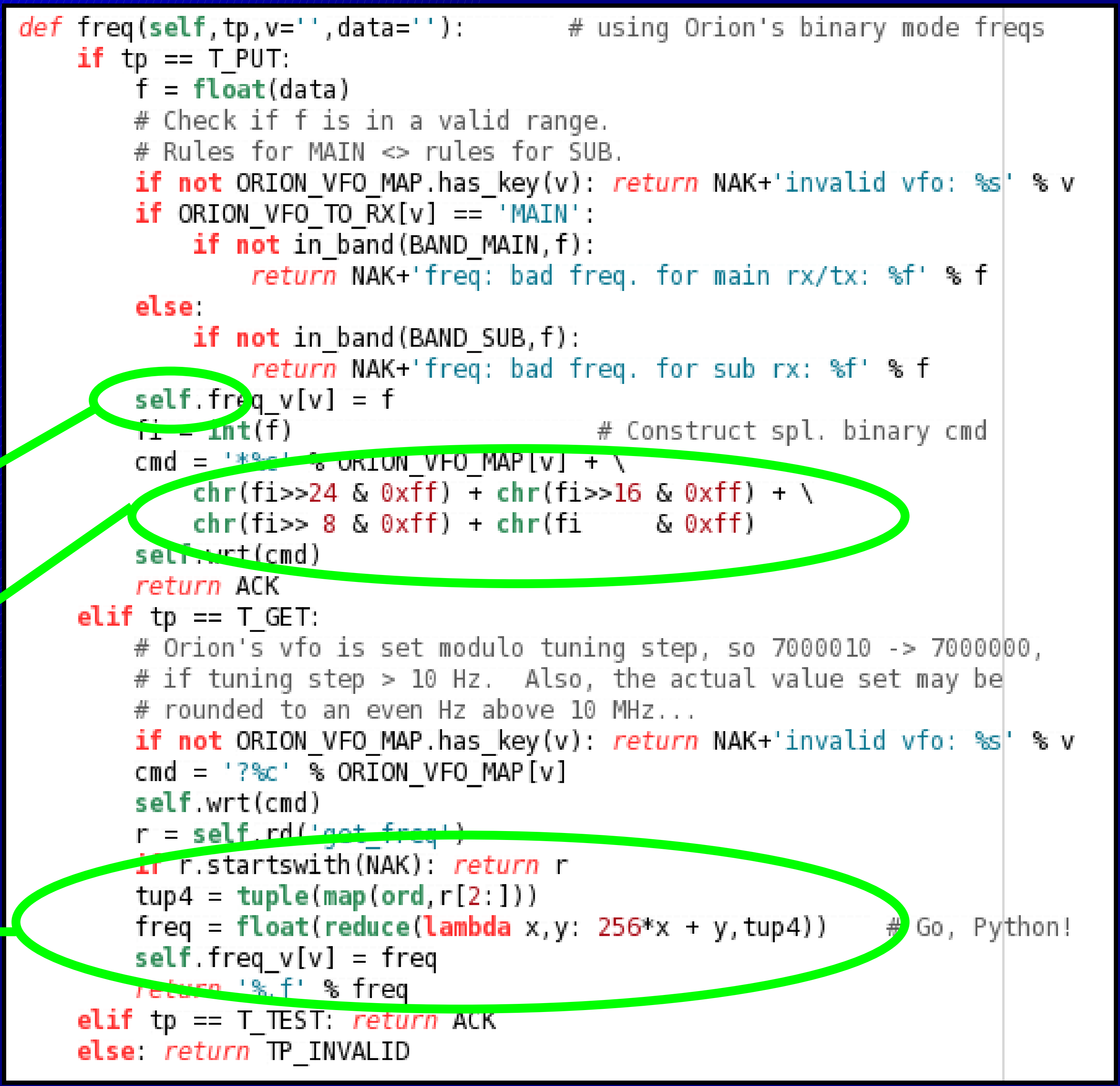
Rigserve TCP server



```
print IDENT
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    s.bind((ALLOWED_IP, PORT))
except socket.error:
    print "Can't open IP Port. Wait 60 secs and try again?"
    sys.exit()
s.listen(5)
try:
    new_socket = 0
    while True:
        new_socket, addr = s.accept()
        print time.asctime(),' Connected from', addr
        new_socket.sendall('Welcome to Rigserve!\n')
        while True:
            rData = new_socket.recv(8192)
             print "rcv:",rData
            # quit->terminate this connection, leave server running
            if rData.upper().startswith('QUIT'):
                new_socket.sendall('QUIT\n')
                break       # drop current conn., but continue to listen
            else:
                reply = str(command(rData))
                    # NB: some commands return non-string formats
                 print "snd:",reply
                new_socket.sendall(reply+'\n')
        new_socket.close()
        print time.asctime(),' Disconnected from', addr
except KeyboardInterrupt:
    if new_socket: new_socket.close()
    s.close()
    print 'All closed'
```

Where it all happens

Rigserve Orion Put/Get Freq.



```
def freq(self,tp,v='',data=''):         # using Orion's binary mode freqs
    if tp == T_PUT:
        f = float(data)
        # Check if f is in a valid range.
        # Rules for MAIN <> rules for SUB.
        if not ORION_VFO_MAP.has_key(v): return NAK+'invalid vfo: %s' % v
        if ORION_VFO_TO_RX[v] == 'MAIN':
            if not in_band(BAND_MAIN,f):
                return NAK+'freq: bad freq. for main rx/tx: %f' % f
        else:
            if not in_band(BAND_SUB,f):
                return NAK+'freq: bad freq. for sub rx: %f' % f
        self.freq_v[v] = f
        fi = int(f)                         # Construct spl. binary cmd
        cmd = '*%c' % ORION_VFO_MAP[v] + \
            chr(fi>>24 & 0xff) + chr(fi>>16 & 0xff) + \
            chr(fi>> 8 & 0xff) + chr(fi     & 0xff)
        self.wrt(cmd)
        return ACK
    elif tp == T_GET:
        # Orion's vfo is set modulo tuning step, so 7000010 -> 7000000,
        # if tuning step > 10 Hz.  Also, the actual value set may be
        # rounded to an even Hz above 10 MHz...
        if not ORION_VFO_MAP.has_key(v): return NAK+'invalid vfo: %s' % v
        cmd = '?%c' % ORION_VFO_MAP[v]
        self.wrt(cmd)
        r = self.rd('get_freq')
        if r.startswith(NAK): return r
        tup4 = tuple(map(ord,r[2:]))
        freq = float(reduce(lambda x,y: 256*x + y,tup4))     # Go, Python!
        self.freq_v[v] = freq
        return '%.f' % freq
    elif tp == T_TEST: return ACK
    else: return TP_INVALID
```

Object-Oriented

C-like

Pythonic

A Rigserve Session

```
$ ./rigserve.py
rigserve 0.30 08/2007 AA6E
Wed Sep  5 20:47:28 2007  Connected
from ('127.0.0.1', 51549)
(Opening rig_type = IC_r8500)
Wed Sep  5 20:51:32 2007
Disconnected from ('127.0.0.1',
51549)
Wed Sep  5 20:52:10 2007  Connected
from ('127.0.0.1', 53463)
(Opening rig_type = IC_r8500)
Wed Sep  5 20:53:48 2007
Disconnected from ('127.0.0.1',
53463)
```

```
$ ./rigclient.py
rigclient.py v. 0.22
Using Port 14652
Connected to server 127.0.0.1
Welcome to Rigserve!
$open RIG IC_r8500
....resp: Icom R8500 communications receiver
$put RIG.CONTROL.init /dev/ham.8500 19200
....resp: OK
$get RIG.VFOA.freq
....resp: 89900000
$get RIG.MAIN.rx_mode
....resp: WFM
$put RIG.VFOA.freq 91.1e6
....resp: OK
$get RIG.VFOA.freq
....resp: 91100000
$quit
....resp: QUIT
client socket closed
```

Rigserve: Vital Statistics

- Started Feb., 2007
- ~ 3,500 lines of code
- Crew: Martin AA6E with Jim MØDNS
- Rigs: TT Orion, Icom R75 and R8500. More to come!

Users for your software?

-- OR --

- **Business plan**
  - **1: Code**
  - **2: ???**
  - **3: Profit!**

- **It can work**
  - **Even free (beer)**

- **Open Source Software (OSS)**
- **Find like-minded people**
  - **Discuss, learn**
- **Give it away!**
  - **Glory, not $$**
  - **Under a license**
- **World will help you.**

Notable OSS Projects

PROJECT	SLOC*
Linux Kernel 2.6	9.8 M**
Gcc	3.2 M**
Mozilla Firefox	1.7 M**
GIMP	1.1 M**
Apache	203 K**
PowerSDR	283 K
Hamlib	137 K
Rigserve	4 K

** Source Lines of Code*

*** From www.ohloh.net*

Support for OSS work

- SourceForge.net ⇒ (n/c)
  - Community
  - Version control & releases
    - CVS, Subversion, etc.
  - Archives, mail/forums, web, etc.

- Vital Info:
  - 1,000,000 registered users
  - 100,000 projects
  - 181 “ham radio” projects

Some Ham Projects @ sf.net

Project	Downloads
TrustedQSL (LOTW)	83K
XASTIR (APRS)	37K
CqiNET (VOIP)	37K
Grig, Gpredict (Rig, Sat)	36K
AI9NL (Knoppix)	31K
Hamlib (Library)	28K
Rigserve (Server)	0.2K

Quick look: Open Source SW

- **What is “free”?**
  - **“free beer” – Ham culture** √
  - **“free speech” – Source is public**
    - **For debugging, improvement, tinkering**
  - **“Free” (both) can be very valuable!**
- **Does it need a license?**

Not that License…

- Shrink-wrap EULA…
- General Public License – GPL
  - Use GPL SW, but your work will be GPL
  - Modifications back to community
- Lesser GPL – LGPL
  - Commercial SW can use a library without becoming LGPL itself
  - Useful for libraries – like Hamlib
- www.gnu.org/licenses

Wrapping Up

- Hamlib wants you!
  - Developers and testers
  - Many rigs need testing & tweaking
  - Learn, have fun, and serve...
- Rigserve wants you, too.
  - Python programmers
- Consider the OSS model
  - Natural for ham projects...

End