MOOS on embedded devices

Ian Baldwin, Paul Newman

Devices & Bindings

- Gumstix

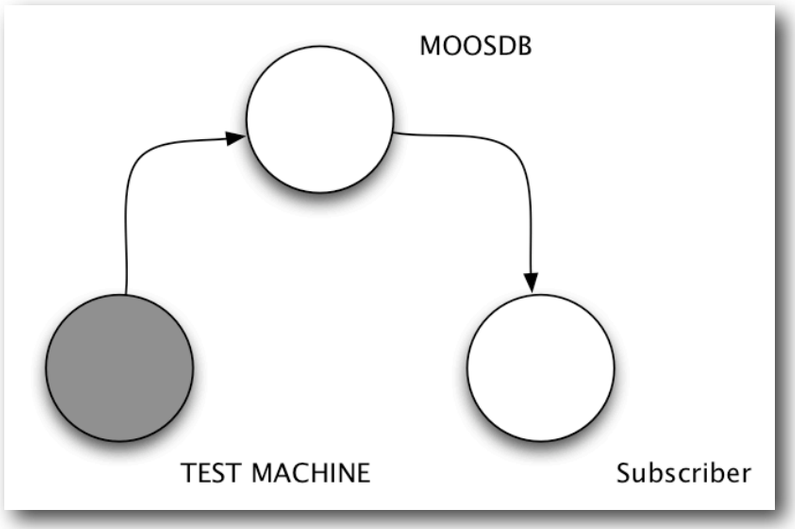
- iPhone/iPod touch

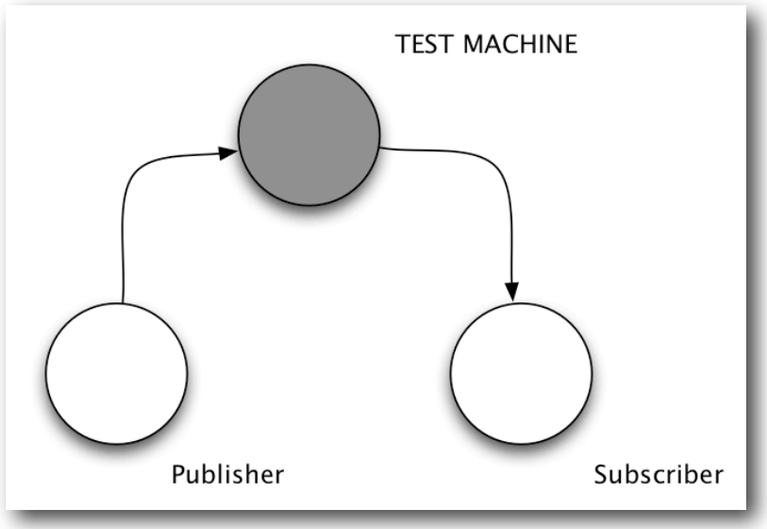
- HTC device

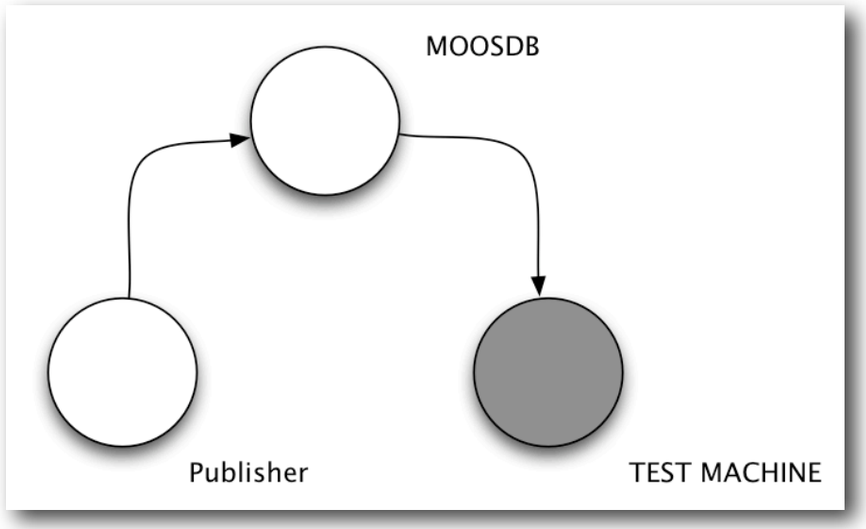
- Python

- Java

Topologies







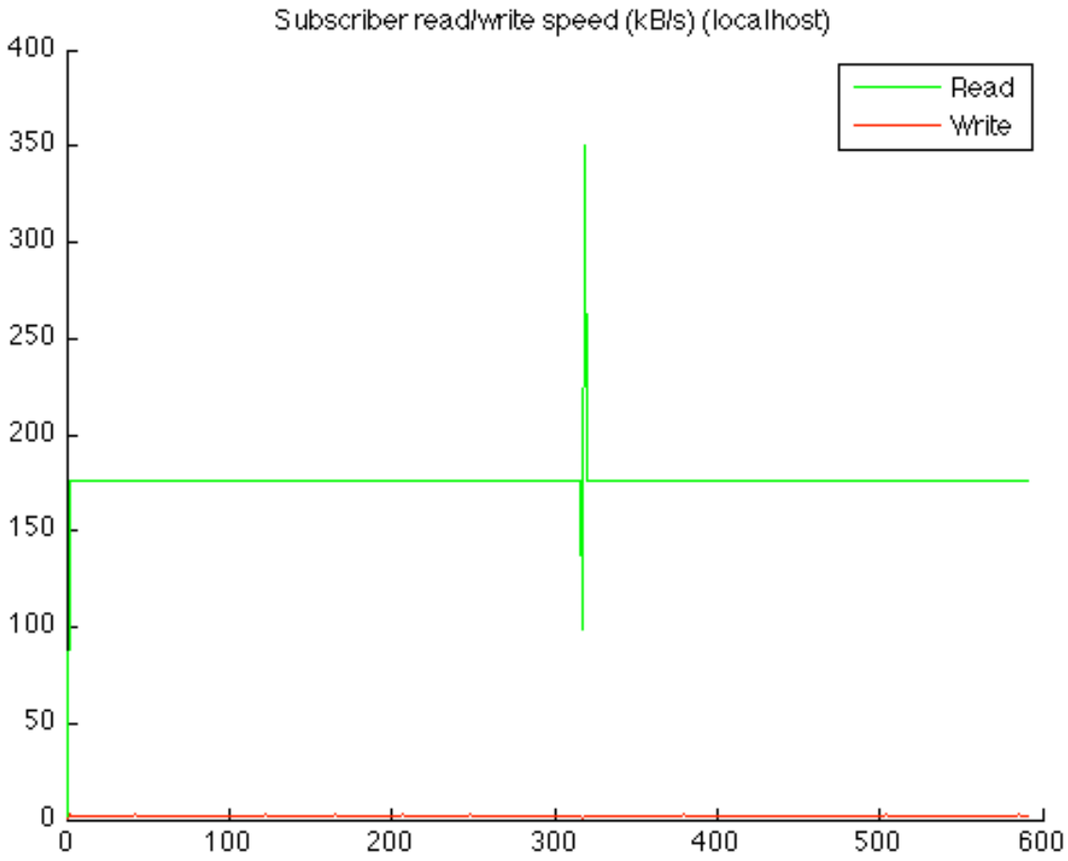
- Instrumentation:

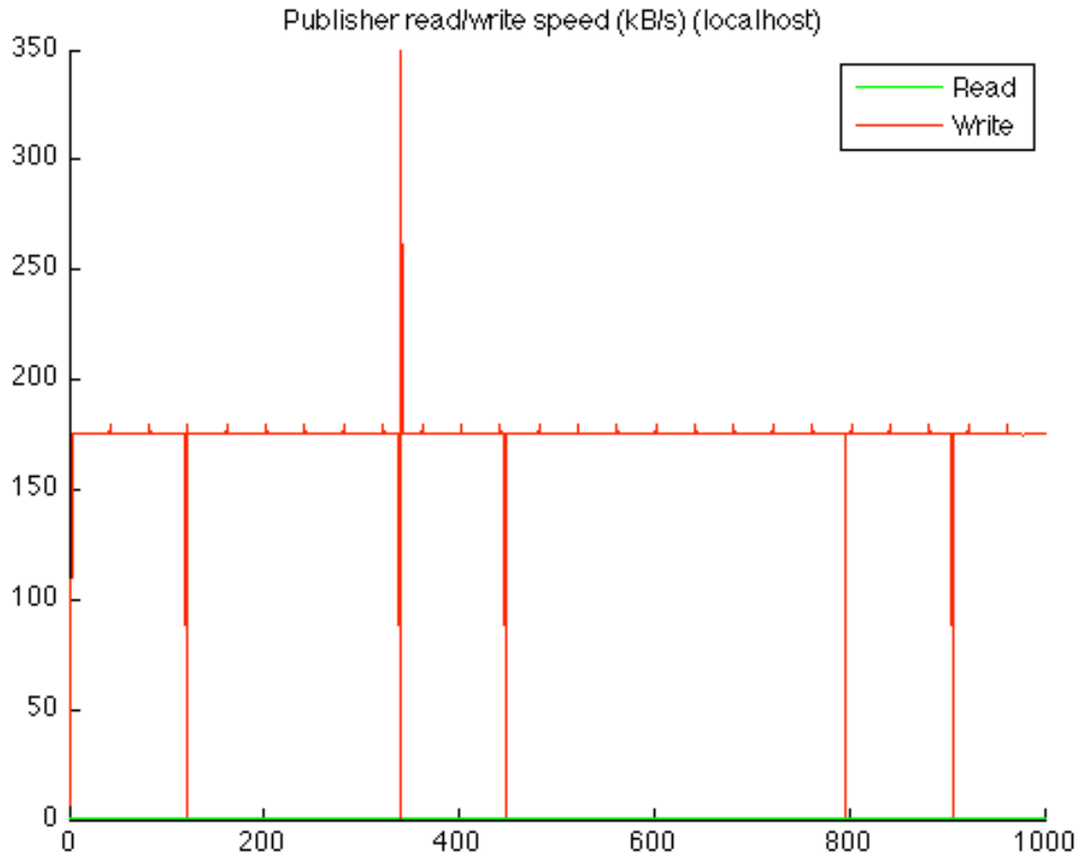
```
bool CMOOSCommClient::DoClientWork()
{

    //Instrument this Loop
    try
    {...}
    catch(CMOOSException e)
    {
        MOOSTrace("Exception in ClientLoop() : %s\n",e.m_sReason);
        OnCloseConnection();
        return false;
    }

    return true;
}
```

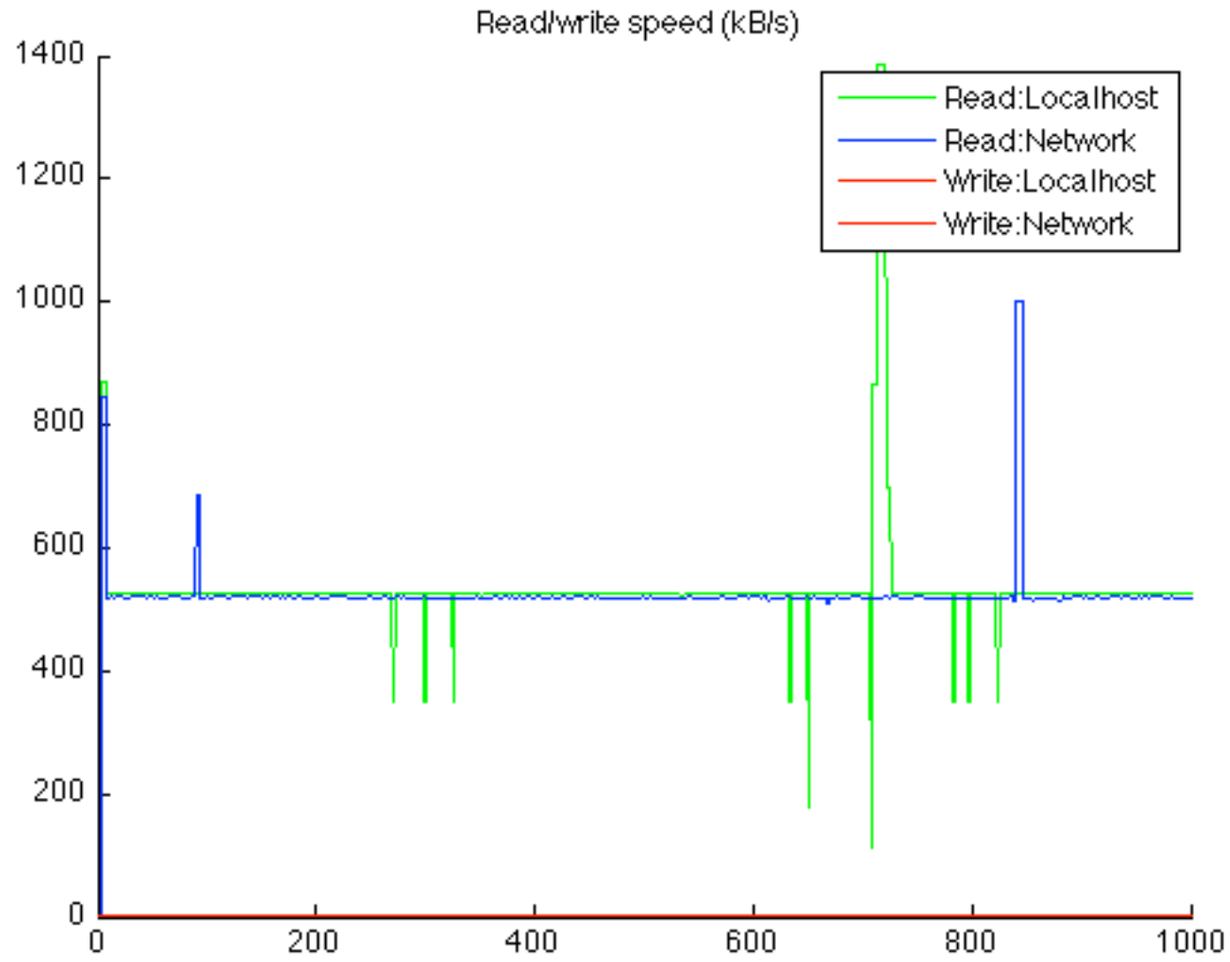
Results





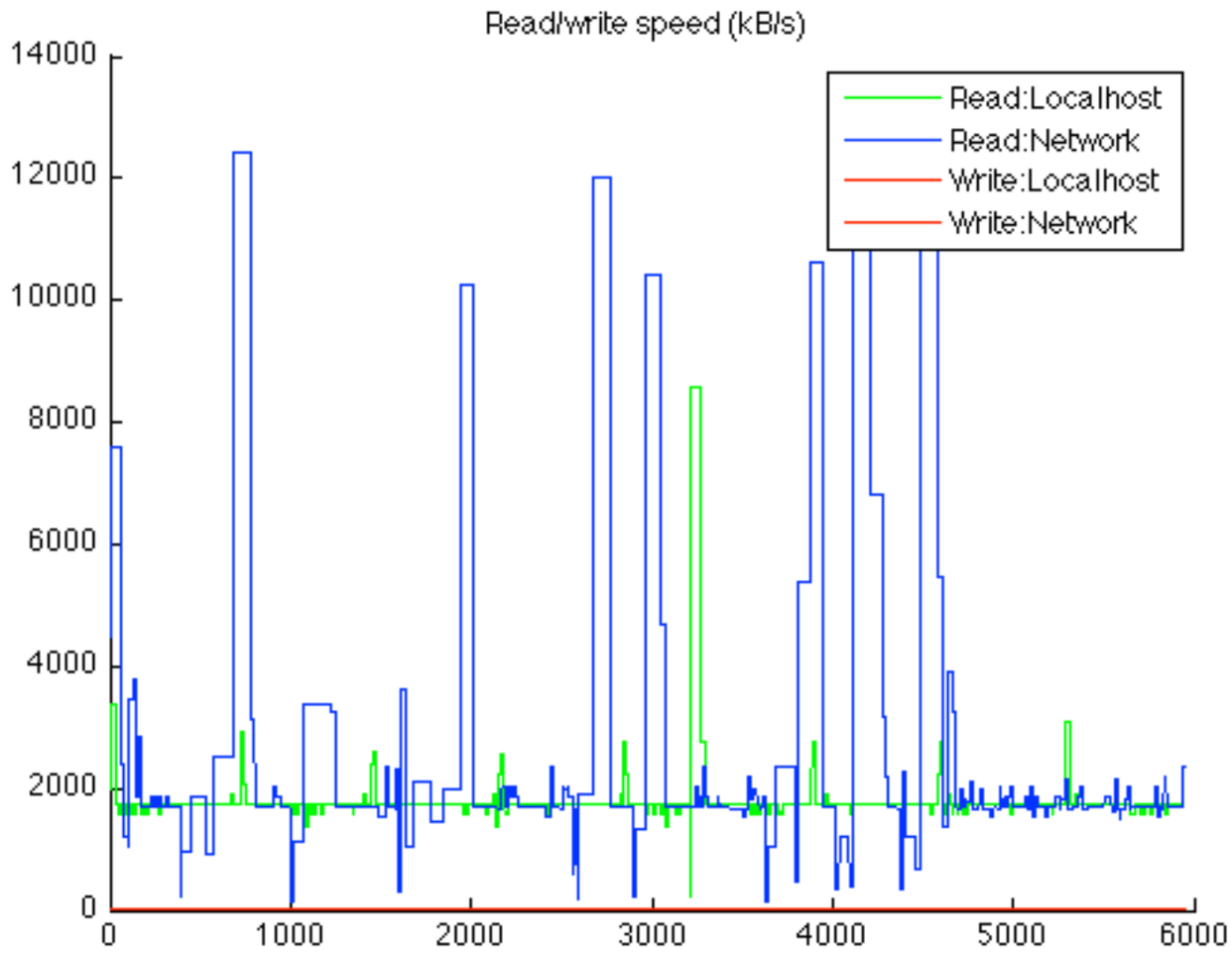
- 1 client, 1 publisher
- Message size ~9K (2000 element vector)
- Transfer rate of 20 Hz
- Data read in real time

Results



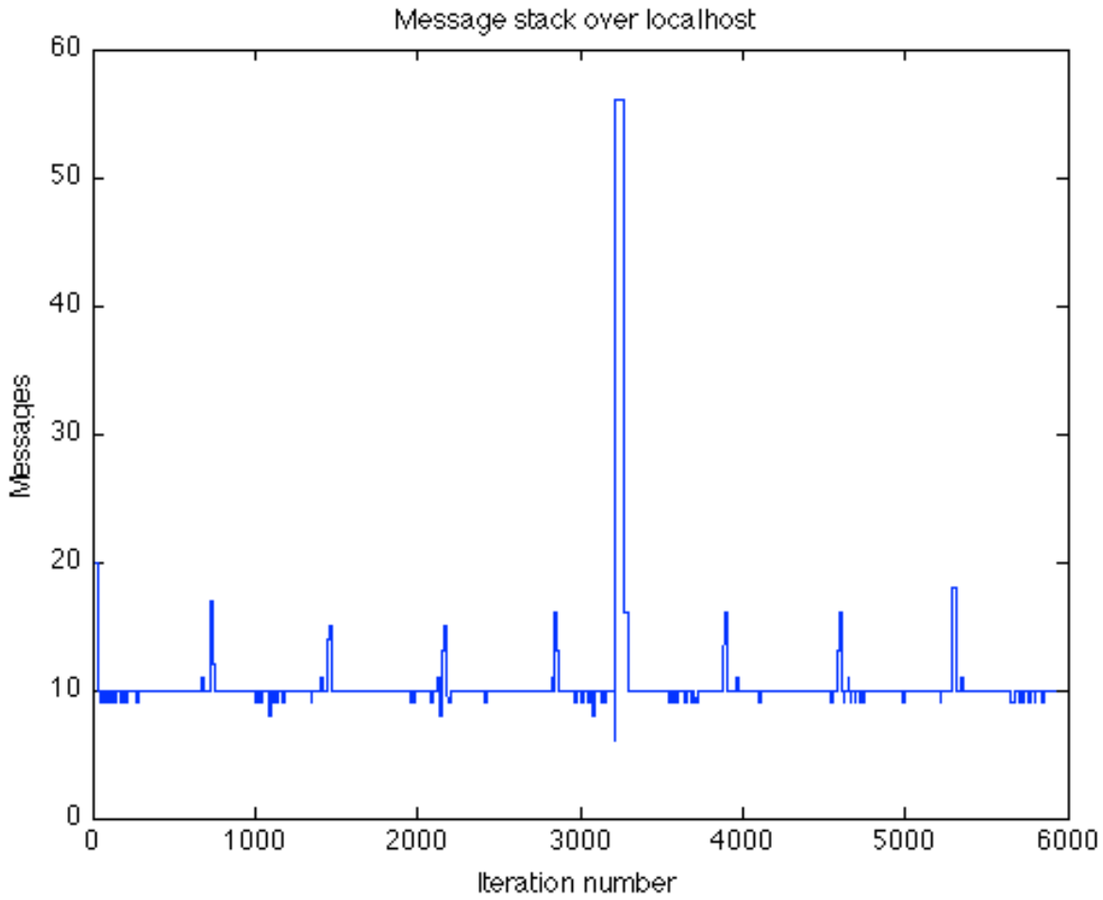
- 1 client, 3 publishers
- Message size ~9K (2000 element vector)
- Transfer rate of 20 Hz

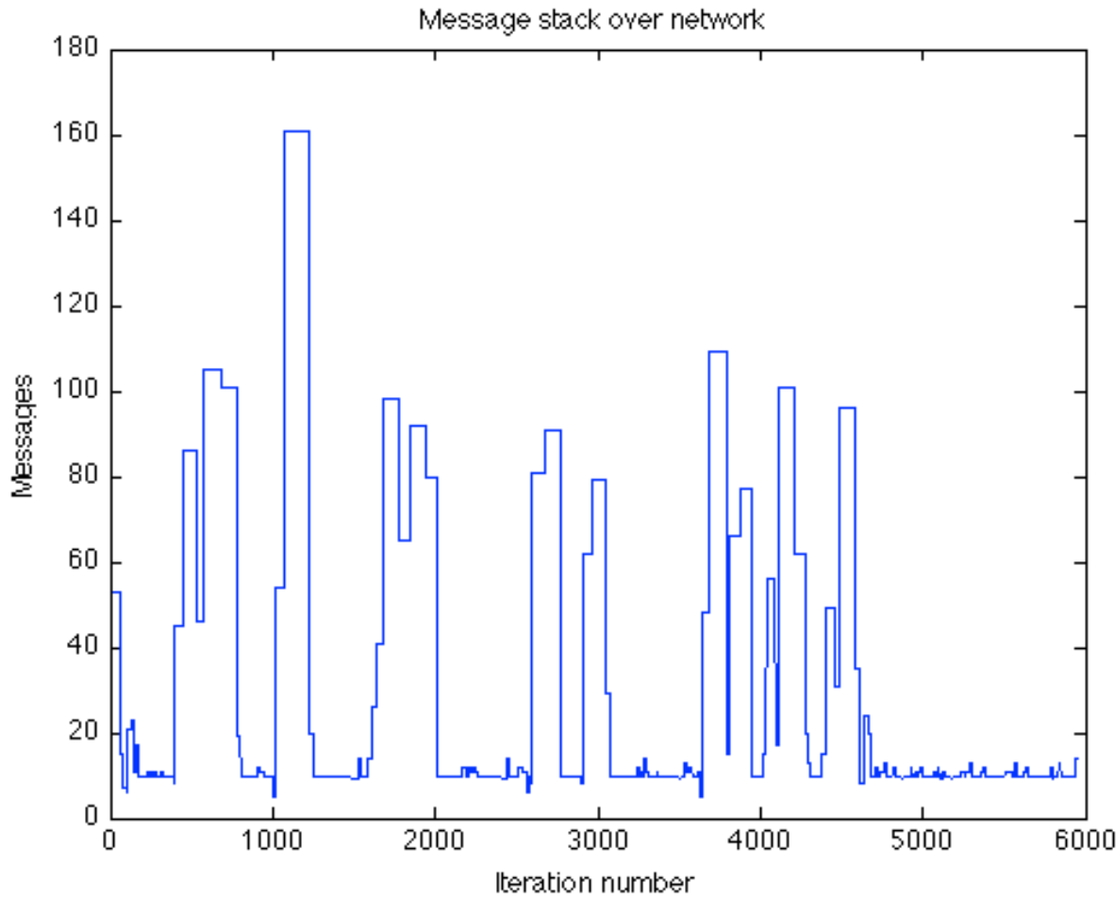
Results



- 1 client, 10 publishers
- Message size ~9K (2000 element vector)
- Transfer rate of 20 Hz
- Client lags (over 10/100 network)

Results





- 1 client, 10 publishers
- MOOS benchmark (over localhost) easily copes with higher transfer rates
- Network bottleneck visible with a larger number of clients

Gumstix

- Overo series
- TI/OMAP Processor
- ARM Cortex-A8 CPU, 720 MHz, 256 MB RAM
- openembedded OS, Linux kernel 2.6
- Use of bitbake and the associated recipes to create code

Building/Configuration

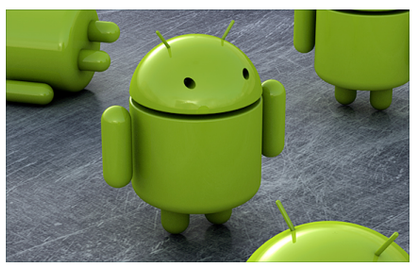
- Gumstix has recently moved to openembedded
- Use of bitbake and the associated recipes to create code

iPhone/iPod Touch

- ARM CPU, 532 MHz, 128 MB RAM
- iPhone OS (now iOS)

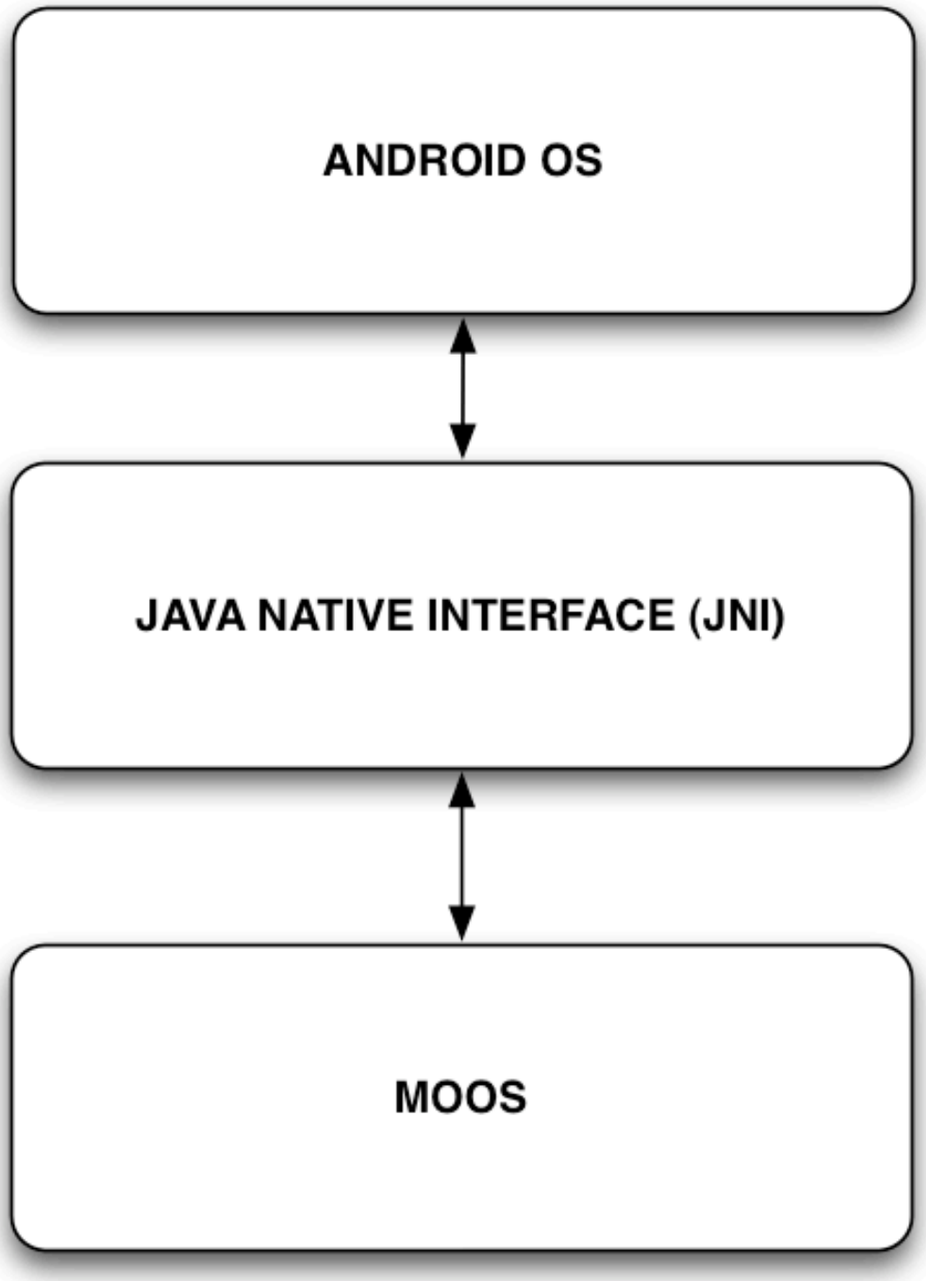
Android

- Qualcomm Snapdragon processor
- ARM Cortex-A8 CPU, 1GHz, 576 MB RAM
- Android OS, Linux kernel 2.6



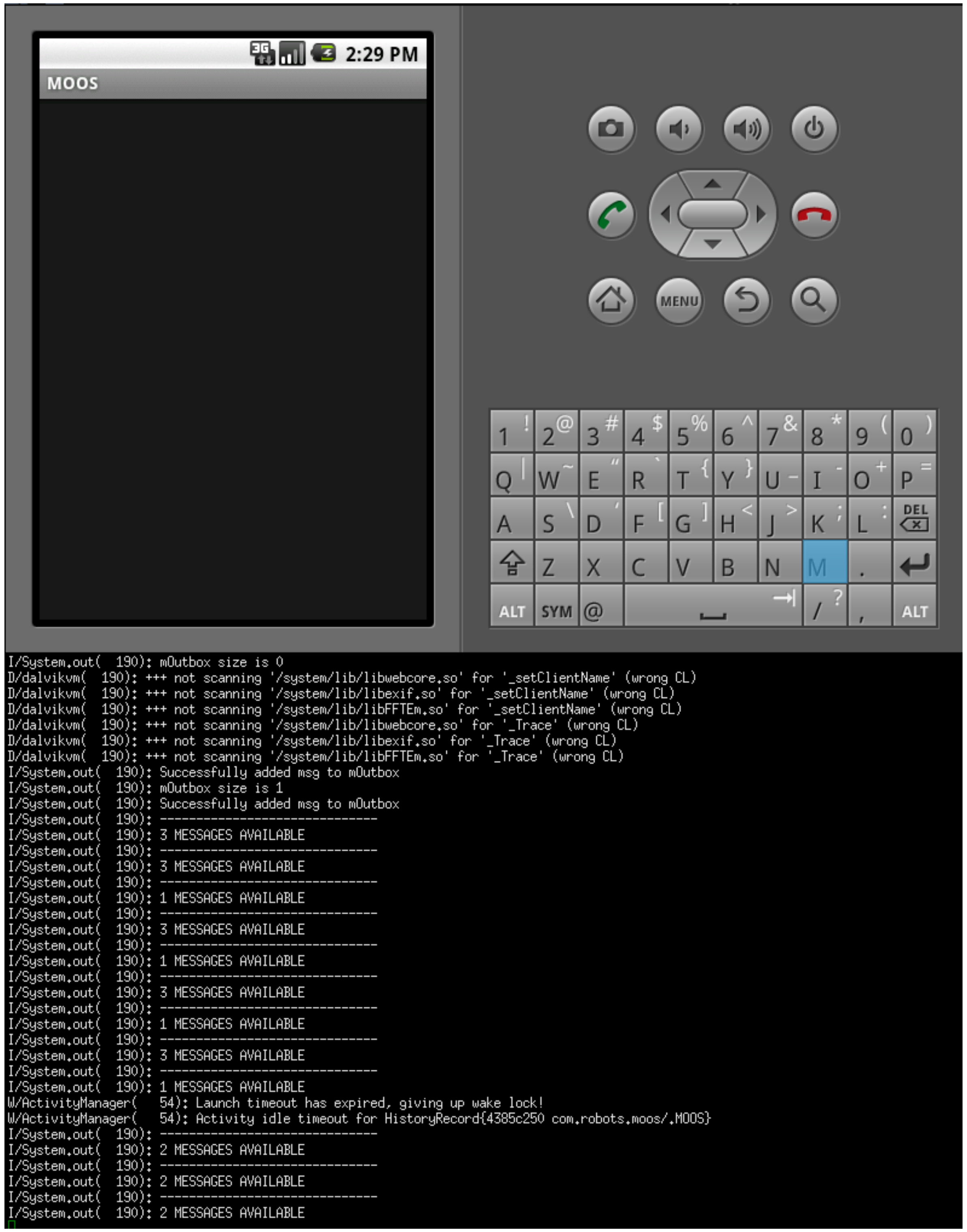
QUALCOMM

Android Interface



- Android OS (tested on API levels 4-6 : v1.6 - v 2.01)
- Without a full-fledged Java Implementation, require interface code
- Tested on MOOS v2307

Building/Configuration



- As of Android API level 6, no way of using C++ to develop native code on the device
- Slight modifications to the Android JDK to allow for C++ compatibility (exceptions, RTTI, standard C++)
- As C++ development is in Google's roadmap, may not be compatible with later versions

pyMOOS

MOOSCommClient.py:30 ▸ MOOSCommClient

```
#!/usr/bin/python

from threading import Thread, Lock, Event;

from time import *

from XPCTcpSocket import *
from CMOOSMsg  import *
from CMOOSCommObject import *
from CMOOSCommPkt import *

class MOOSCommClient( Thread ):

    def __init__(self):
        super( Thread, self).__init__()
        Thread.__init__(self)

        #Initialise sockets
        self.sock = XPCTcpSocket();
        self.comms = MOOSCommObject();

        self.bConnected = False;
        self.m_bQuit = False;
        self.host    = 'localhost'
        self.m_sMyName = 'test_python'

        self.m_Outbox = []
        self.m_Inbox = []

        self.m_Outbox_Lock = Lock();

    def Register( self, variable, interval ):
        if not variable:
            return False

        MsgR = MOOSMsg( 'R', variable, interval, -1 )

        if self.__Post( MsgR ):
            return True
        else:
            return False
```

- Uses Boost::Python library to expose various aspects of the MOOS codebase
- Currently, consists of a MOOSCommClient class - easy to use to rapidly test new clients in the Python language
- Currently implementing the full MOOS client structure

jMOOS

```java
package com.robots.MOOS;

import java.net.*;
import java.nio.*;
import java.io.*;

import java.util.Vector;
import java.util.Iterator;

import com.robots.MOOS.JMOOSMsg;
import com.robots.MOOS.JXPCSocket;
import com.robots.MOOS.JMOOSGlobalHelper;


public class JMOOSCommClient extends JMOOSCommObject implements Runnable
{
    //Core data types
    final char  MOOS_NOTIFY     ='N';
    final char  MOOS_REGISTER   ='R';
    final char  MOOS_UNREGISTER ='U';
    final char  MOOS_NOT_SET    ='~';
    final char  MOOS_COMMAND    ='C';
    final char  MOOS_ANONYMOUS  ='A';
    final char  MOOS_NULL_MSG   ='.';
    final char  MOOS_DATA       ='i';
    final char  MOOS_POISON     ='K';
    final char  MOOS_WELCOME    ='W';
    final char  MOOS_SERVER_REQUEST = 'Q';

    //MESSAGE DATA TYPES
    final char  MOOS_DOUBLE     ='D';
    final char  MOOS_STRING     ='S';

    private JMOOSGlobalHelper   helper;
    private JXPCSocket          _socket;
    private int                 _port;
    private String              _hostname;
    private JMOOSMSG_LIST       mOutbox;
    private JMOOSMSG_LIST       mInbox;
    private String              mName;
    private int                 mReadMessages;

    private Vector<String>      mPublishing;
    private Vector<String>      mSubscribing;

    private Thread      mThread;

```

- jMOOS (not JMOOS, developed by Brass Rat development (http://brassratdev.com/)
- Uses JNI, requires interface/wrapper code
- No changes to MOOS source