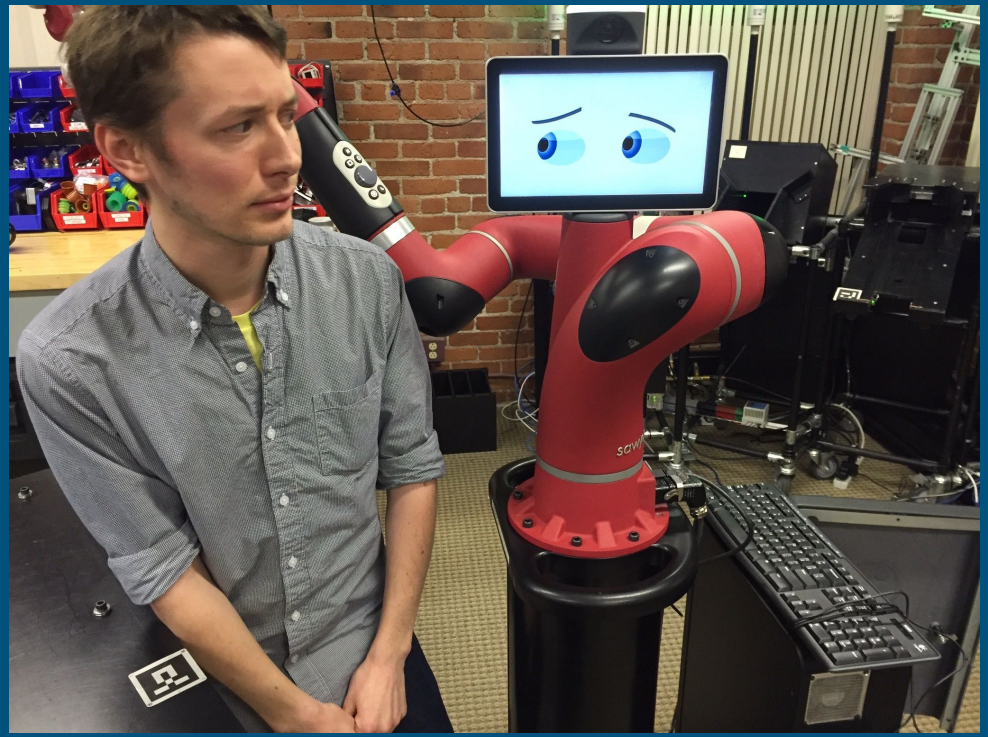
rosnodejs

ROS and the Web

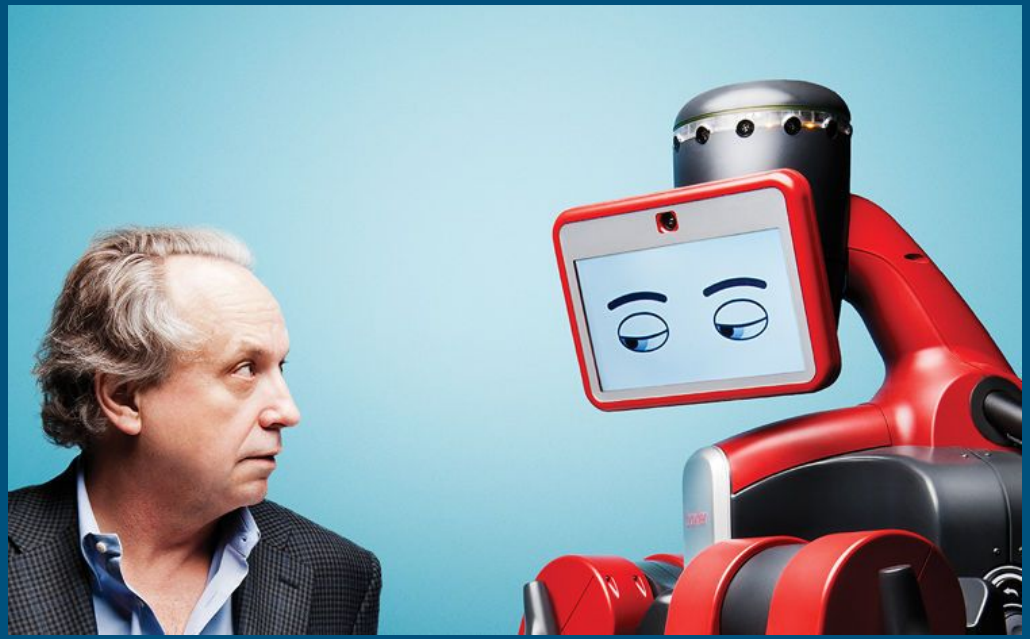
(self introduction)

Chris Smith

Software Engineer @rethinkrobotics



Rethink Robotics



http://www.generationrobots.com/blog/en/tag/baxter-research-robot-en/

http://www.rethinkrobotics.com/sawyer/

The Bones

- Heavily Abridged Histories, ROS & Rethink
- Why we built it
- Code Examples, client library comparison
- Demos
- Limits
- Future Work

ROS (An Abridged Web-Tech-History)

June 2009 - ROS: an open-source Robot Operating System, *Quigley et al*

August 2011 - Robots as web services, *Osentoski et al*

2011-2013 - Brandon Alexander works on an earlier rosnodejs at Willow

2011-present - myriad projects on RobotWebTools

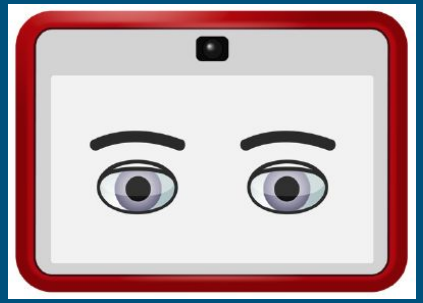
2012 - rosbridge 2.0

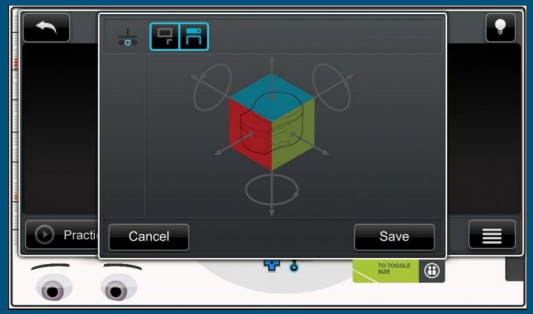
2017 - Reactive Web Interfaces (Next Talk!)

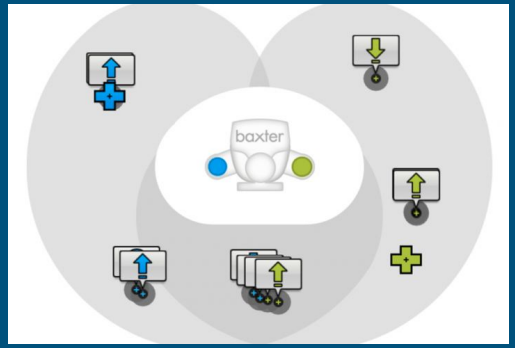
ROS at Rethink (An Abridged History)

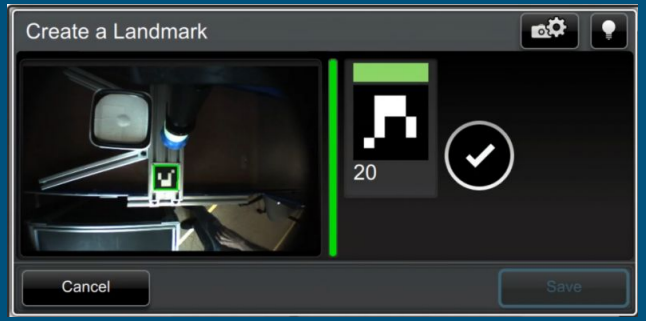
```
$ git log | grep -i “ ros “ -A 5 -B 5

Author: Michael Siracusa <msiracusa@rethinkrobotics.com>
Date:   Tue Jun 22 21:24:08 2010 +0000

    re #441 (7)

    This checkin adds basic ROS build integration…
```

Rethink (Intera 3.3 - November 2015)





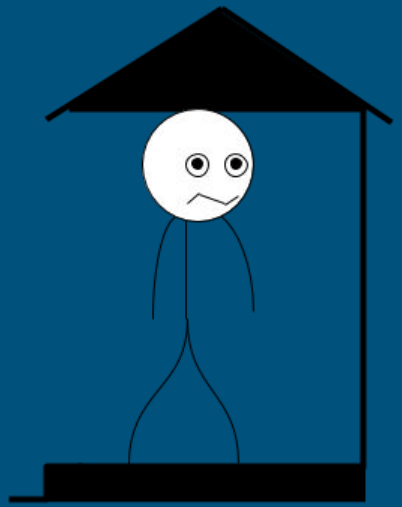




http://mfg.rethinkrobotics.com/mfg-mediawiki-1.22.2/images/8/8c/Baxter_User_Guide_3.3.pdf

http://mfg.rethinkrobotics.com

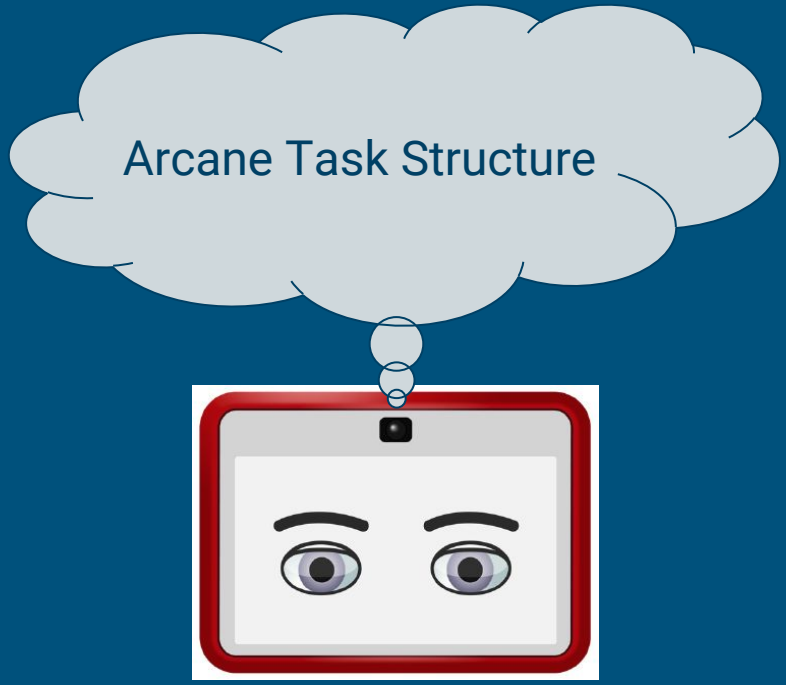
Intera 3.x - Limitations



“Simple things should be simple, complex things should be possible”

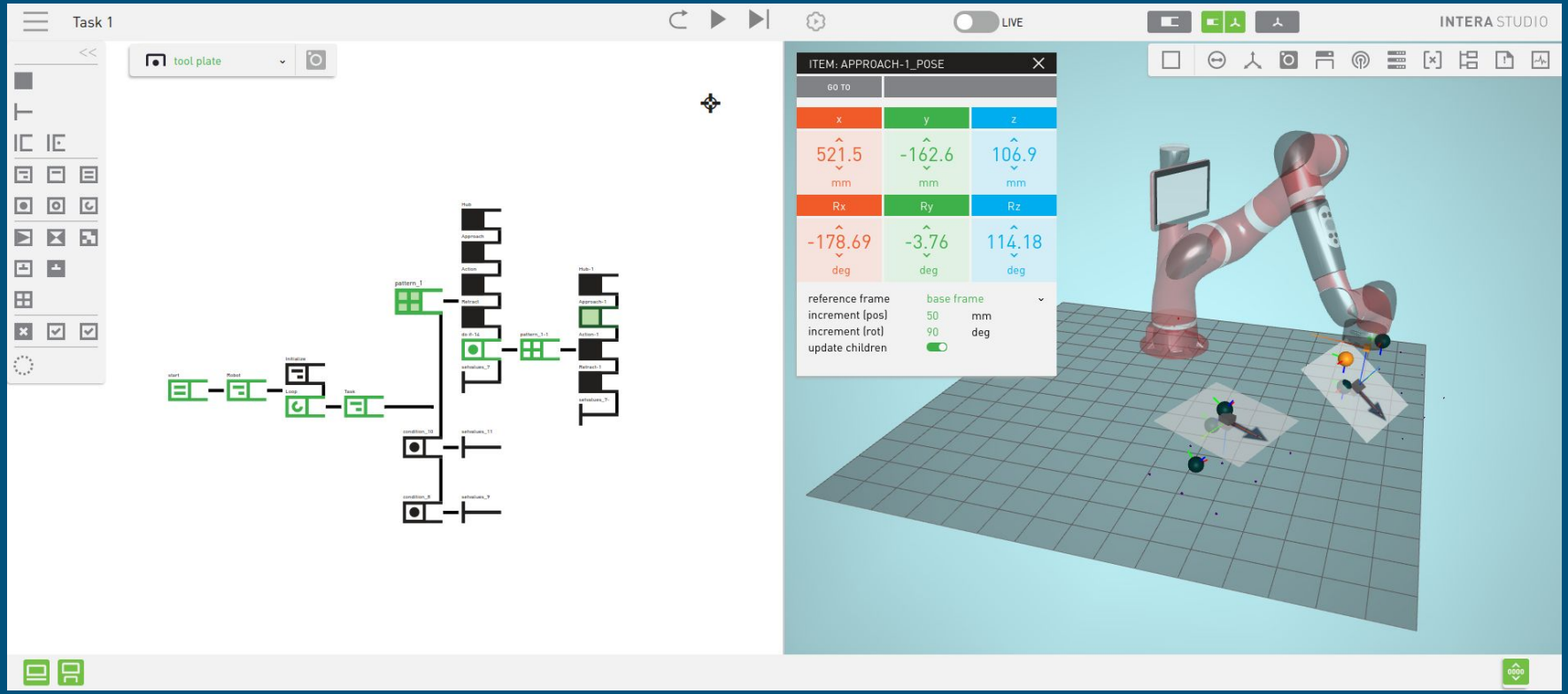
- Alan Key

Intera 3.x - Limitations

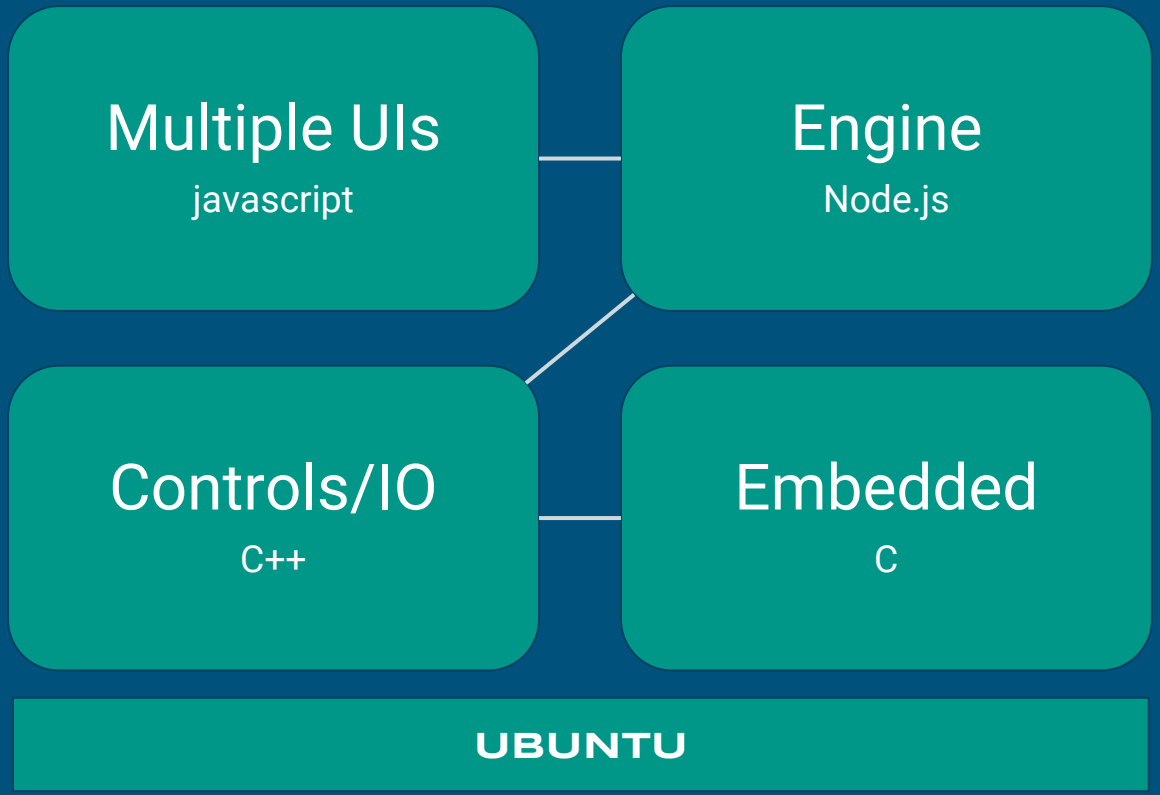


http://disney.wikia.com/wiki/Merlin

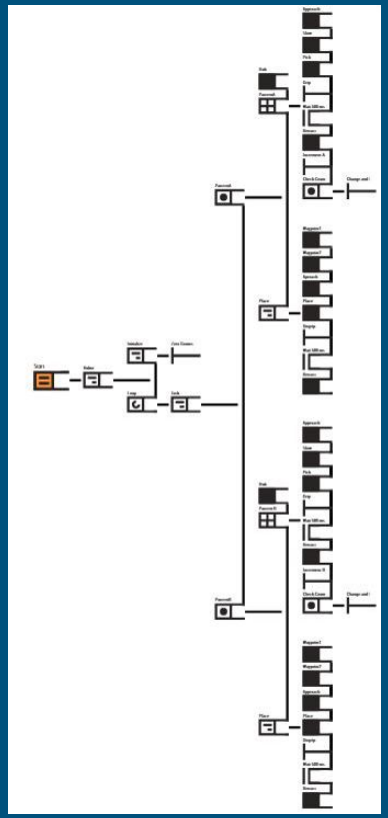
Intera 5


Task 1
LIVE
INTERA STUDIO
tool plate
ITEM: APPROACH-1_POSE
GO TO
x
y
z
521.5
mm
-162.6
mm
106.9
mm
Rx
Ry
Rz
-178.69
deg
-3.76
deg
114.18
deg
reference frame
base frame
increment (pos)
50
mm
increment (rot)
90
deg
update children

Intera 5

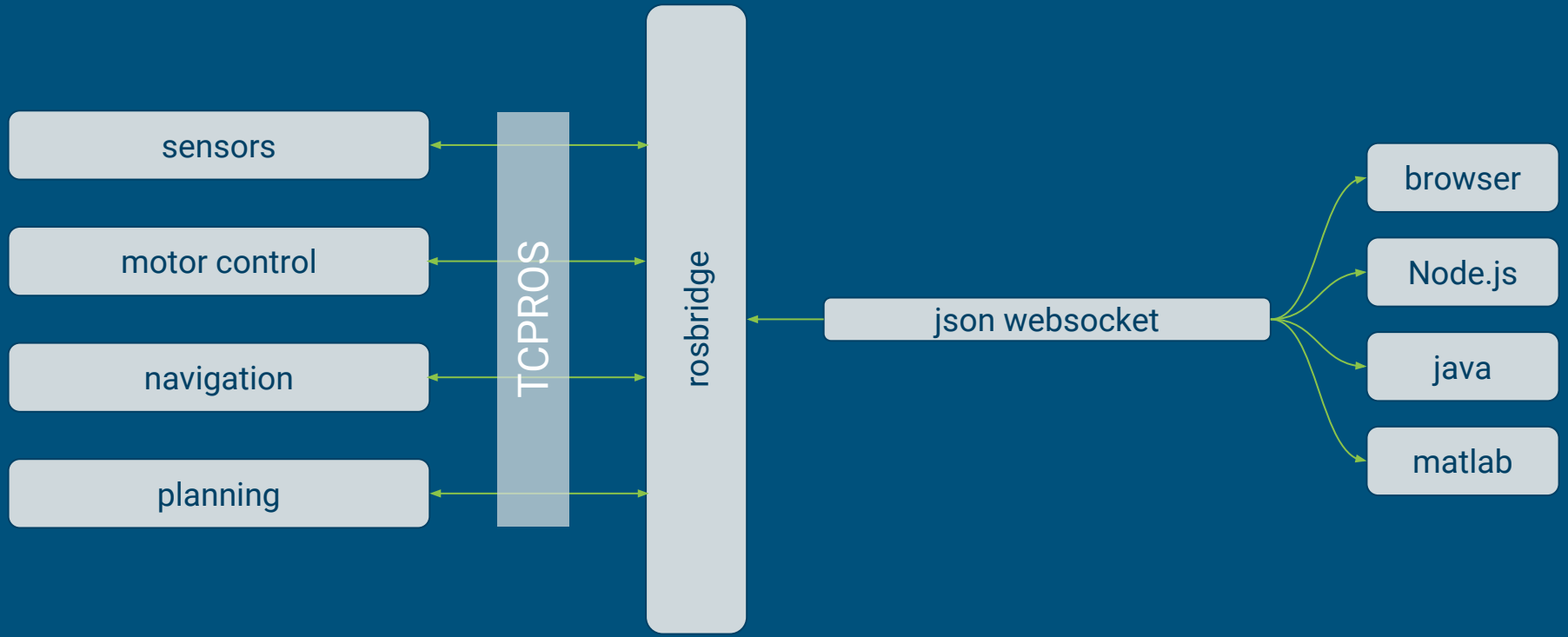

Multiple UIs
javascript
Engine
Node.js
Controls/IO
C++
Embedded
C
UBUNTU

Intera 5 - The Engine



- Web Backend
- Behavior Tree Engine/Editor
- Software Interface to Robot

ROSBridge


sensors
motor control
navigation
planning
TCPROS
rosbridge
json websocket
browser
Node.js
java
matlab

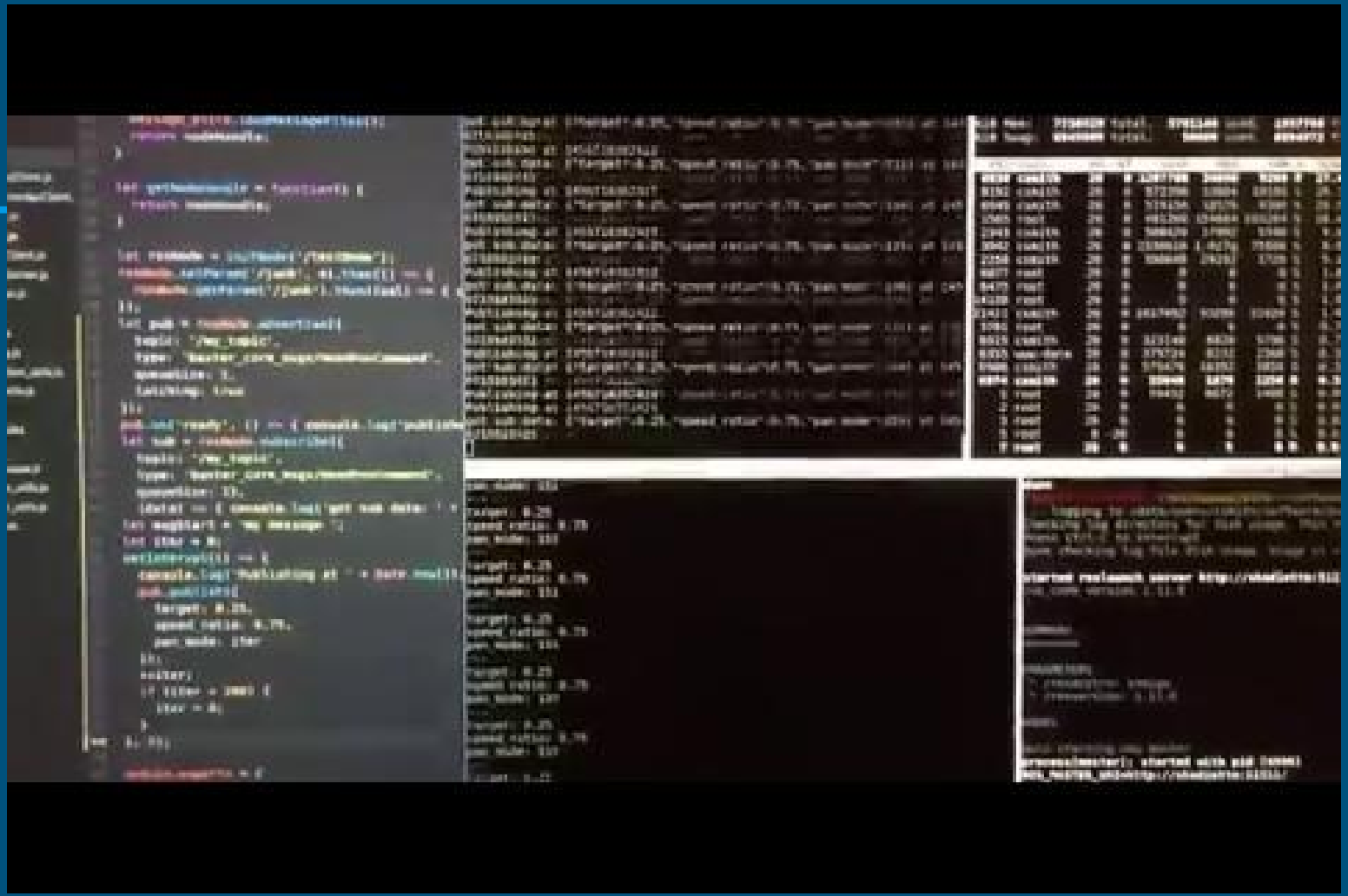
Issues...

- issues with large amounts of data

- bson encoding, web video server, tf2 web republisher, depthcloud encoder

batteries not included?

a solution



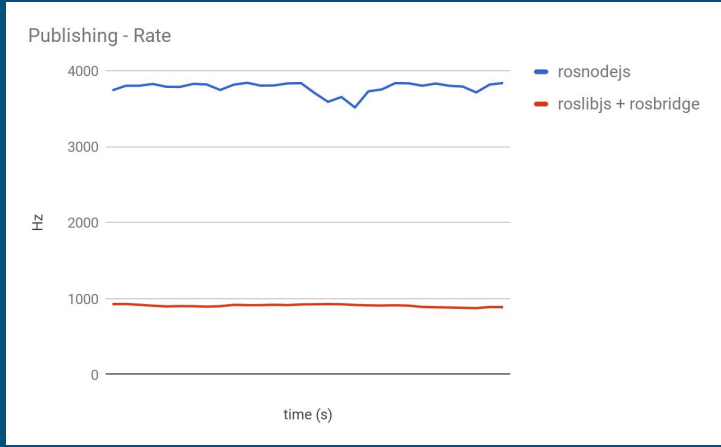
Testing

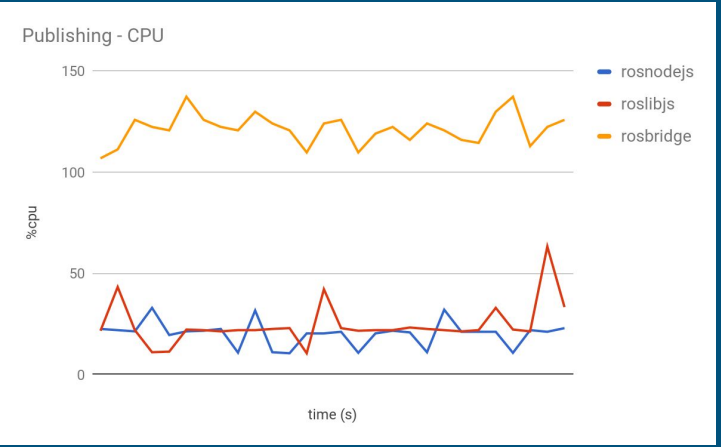
Publishing batch of 5 messages in a loop.

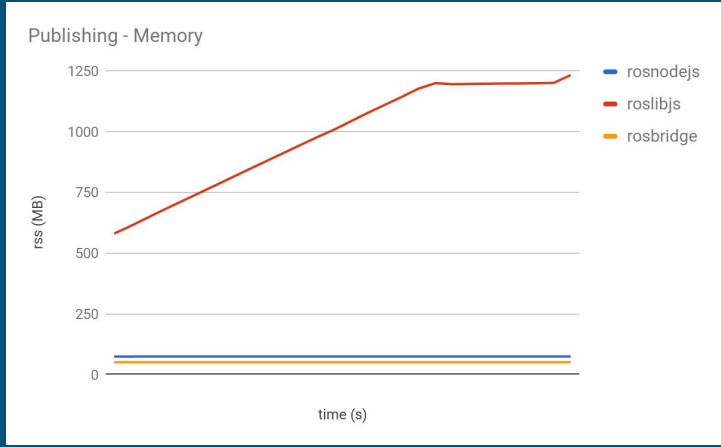
Sampling at 1Hz

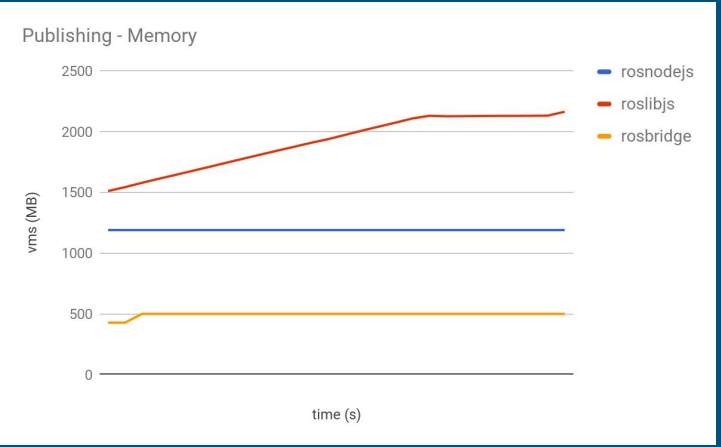
5000 message window for rate

30 seconds









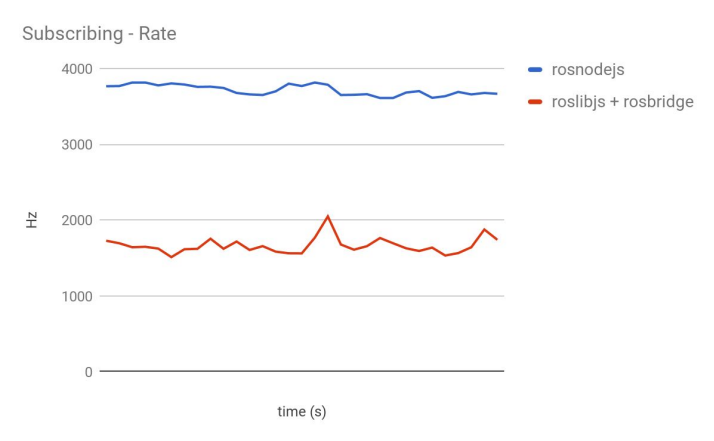
Testing

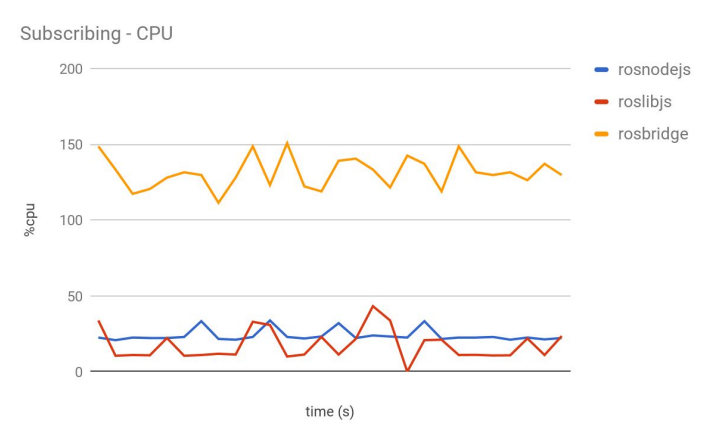
Subscribing to rosnodejs publishing from previous test

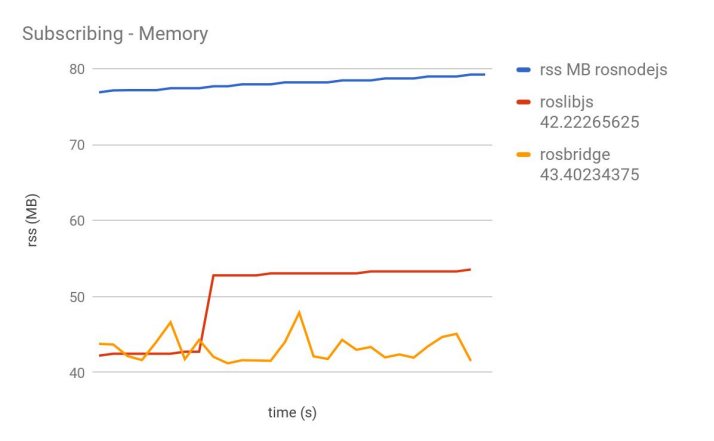
Sampling at 1Hz

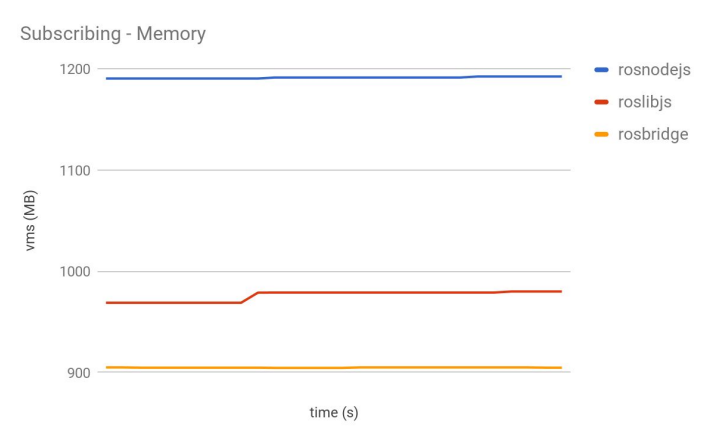
5000 message window for rate

30 seconds









rosnodejs - Features

- publishers, subscribers
- services
- params
- actions
- namespacing
- time
- logging
- message generation
- executable (like python using `#!/usr/bin/env node`)

rosnodejs - Initialization/Shutdown

```
/**
 * @param nodeName {string}
 * @param [options] {object}
 * @param [options.anonymous] {boolean}
 * @param [options.rosMasterUri] {string}
 *   ... others
 * @returns {Promise}
 */
rosnodejs.initNode(nodeName, options);

/**
 * @returns {Promise}
 */
rosnodejs.shutdown()
```

```
rosnodejs.on('shutdown', function() {
  // handle rosnodejs being shutdown
});
```

rosnodejs - NodeHandles

```
const nh = rosnodejs.nh;

nh.setNamespace(...);
nh.advertise(...);
nh.subscribe(...);
nh.advertiseService(...);
nh.serviceClient(...);
nh.actionClient(...);
nh.deleteParam(...);
nh.setParam(...);
nh.getParam(...);
nh.hasParam(...);
```

rosnodejs - Publishers

```
/**
 * @param topic {string}
 * @param type {string|object}
 * @param [options] {object}
 * @param [options.latching] {boolean}
 * @param [options.tcpNoDelay] {boolean}
 * @param [options.queueSize] {number}
 * @param [options.throttleMs] {number}
 * @returns {Publisher}
 */
nh.advertise(topic, type, options)
```

```
// advertising
var pub1 = nh.advertise('/chatter', 'std_msgs/String');

var StringMsg = rosnodejs.require('std_msgs').msg.String;
var pub2 = nh.advertise('/chatter', StringMsg);

var pub3 = nh.advertise('/chatter', StringMsg, { queueSize: 2 });

// publishing
pub1.publish({ data: 'Hello World' });

var msg = new StringMsg();
msg.data = 'Hello World';
pub2.publish(msg);

pub3.publish( new StringMsg({ data: 'Hello World' }) );

// shutdown
pub1.shutdown();
pub2.shutdown();
pub3.shutdown();
```

rosnodejs - Subscribers

```
/**
 * @param topic {string}
 * @param type {string|object}
 * @param [callback] {function}
 * @param [options] {object}
 * @param [options.queueSize] {number}
 * @param [options.throttleMs] {number}
 * @returns {Subscriber}
 */
nh.subscribe(topic, type,
             callback, options)
```

```
function callback(msg) {
  console.log('%j', msg);
}

// subscribing
var sub1 = nh.subscribe('/chatter', 'std_msgs/String', callback);

var StringMsg = rosnodejs.require('std_msgs').msg.String;
var sub2 = nh.subscribe('/chatter', StringMsg, callback);

var sub3 = nh.subscribe('/chatter', StringMsg,
                        callback, { queueSize: 2 });

var sub4 = nh.subscribe('/chatter', StringMsg);
sub4.on('message', callback);

// shutdown
sub1.shutdown();
sub2.shutdown();
sub3.shutdown();
sub4.shutdown();
```

rosnodejs - Services

```
/**
 * @param service {string}
 * @param type {string|object}
 * @param callback {function}
 * @returns {ServiceServer}
 */
nh.advertiseService(service, type, callback)

/**
 * @param service {string}
 * @param type {string|object}
 * @param [options] {object}
 * @returns {ServiceClient}
 */
nh.serviceClient(service, type, options)
```

```
var AddTwoInts = rosnodejs.require('beginner_tutorials').Srv.AddTwoInts;

function add(req, res) {
  res.sum = req.a + req.b;
  return true;
}

// create the service
var service = nh.advertiseService('/add_two_ints', AddTwoInts, add);

// create the client
var client = nh.serviceClient('/add_two_ints', AddTwoInts);
var req = new AddTwoInts.Request();
req.a = 1;
req.b = 2;

client.call(req)
.then((res) => {
  console.log('%j', res);
});
```

rosnodejs - Params

```
/**
 * @param key {string}
 * @returns {Promise}
 */
nh.getParam(key)

/**
 * @param key {string}
 * @param value {*}
 * @returns {Promise}
 */
nh.setParam(key, value)

/**
 * @param key {string}
 * @returns {Promise}
 */
nh.hasParam(key)

/**
 * @param key {string}
 * @returns {Promise}
 */
nh.deleteParam(key)
```

```
const val = await nh.getParam('my_param');

await nh.setParam('my_param', val + 4);

await nh.deleteParam('my_param');

if (await nh.hasParam('my_param')) {
  throw new Error('my_param should be deleted!');
}
```

rosnodejs - Logging/Time

```
// log messages to stdout, stderr, rosout
const log = rosnodejs.log;

log.debug('debug message');
log.info('info message');
log.warn('warn message');
log.error('error message');
log.fatal('fatal message');


log.debugThrottle(100, 'debug message');
log.infoThrottle(100, 'info message');
log.warnThrottle(100, 'warn message');
log.errorThrottle(100, 'error message');
log.fatalThrottle(100, 'fatal message');

// check and set level
if (log.getLevel() === 'info') {
  log.setLevel('warn');
}
```

```
// time conversion methods have ms precision (because js)

const Time = rosnodejs.Time;

// returns sim time or wall clock as appropriate
const now1 = Time.now();

// just use wallclock, convert between representations
const now2 = Time.dateToRosTime(Date.now());

const then = Time.rosTimeToDate(now1);
```

rosnodejs - Messages

```
// provided automatically if running in a catkin workspace in kinetic or later
const SensorMsgs = rosnodejs.require('sensor_msgs');

const Image = SensorMsgs.msg.Image;
const SetCameraInfo = SensorMsgs.srv.SetCameraInfo;


// can be built for you in indigo
rosnodejs.loadAllPackages();


// can be generated on the fly if you prefer
rosnodejs.initNode('my_node', { onTheFly: true })
.then(function() {
  const StdMsgs = rosnodejs.require('std_msgs'); // won't work before initNode completes
  ...
});

// provide constants
const GoalStatus = rosnodejs.require('actionlib_msgs').msg.GoalStatus;
const pendingStatus = GoalStatus.Constants.PENDING;
```

(Pseudo) Code Examples

```
#include <ros/ros.h>
#include <std_msgs/String.h>

ros::init(argc, argv, “/my_node”);
ros::NodeHandle nh;

void chatterCallback(const std_msgs::String& msg)
{
  ROS_INFO(“Hello, %s”, msg.data.c_str());
}

ros::Publisher pub = nh.advertise<std_msgs::String>(“/chatter”, 1);
ros::Subscriber sub = nh.subscribe(“/chatter”, 1, chatterCallback);

std_msgs::String msg;
msg.data = "ROSCON";

ros::Rate r(1);
while (ros::ok())
{
  pub.publish(msg);
  r.sleep()
}
```

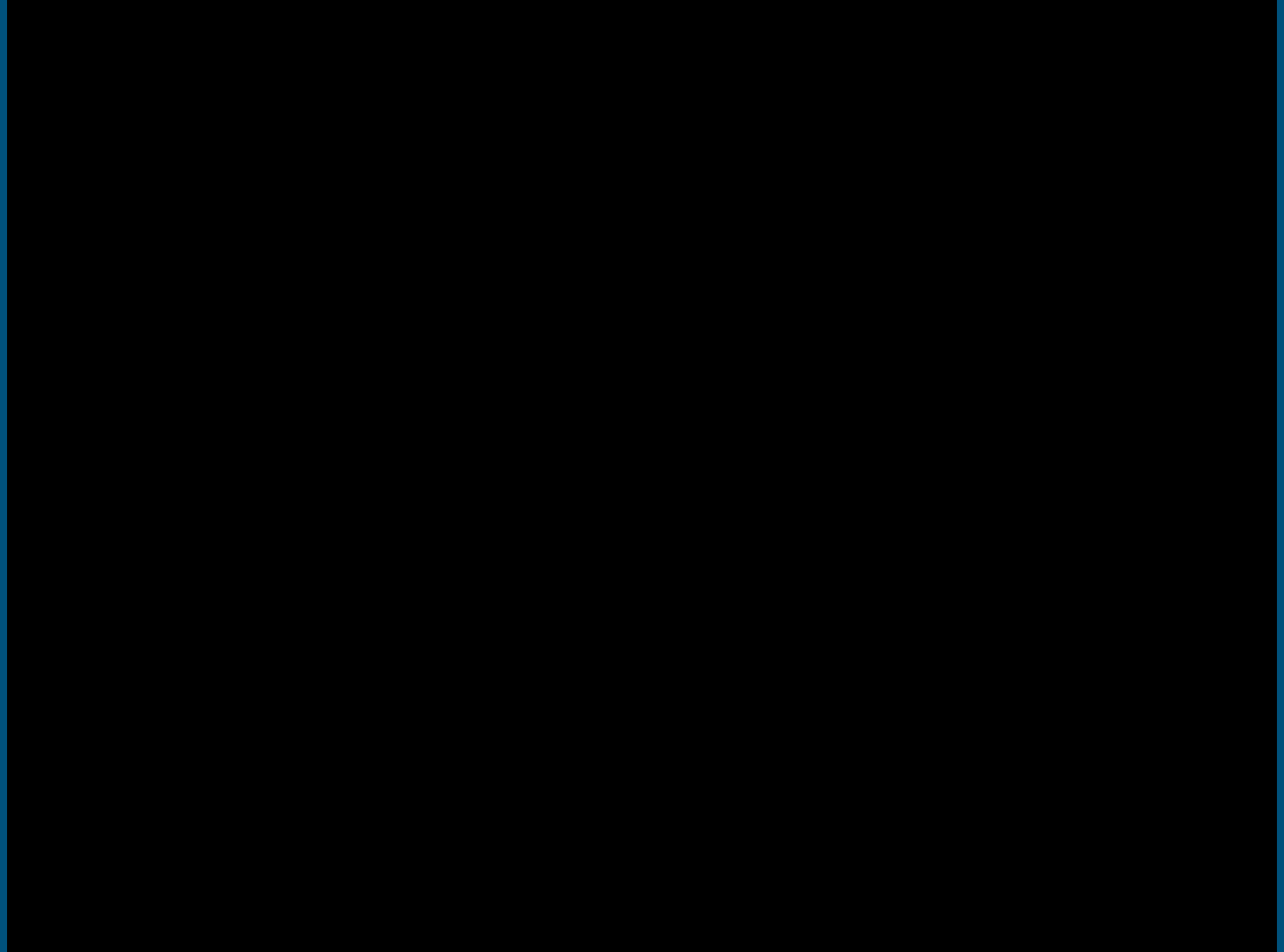
Code Examples

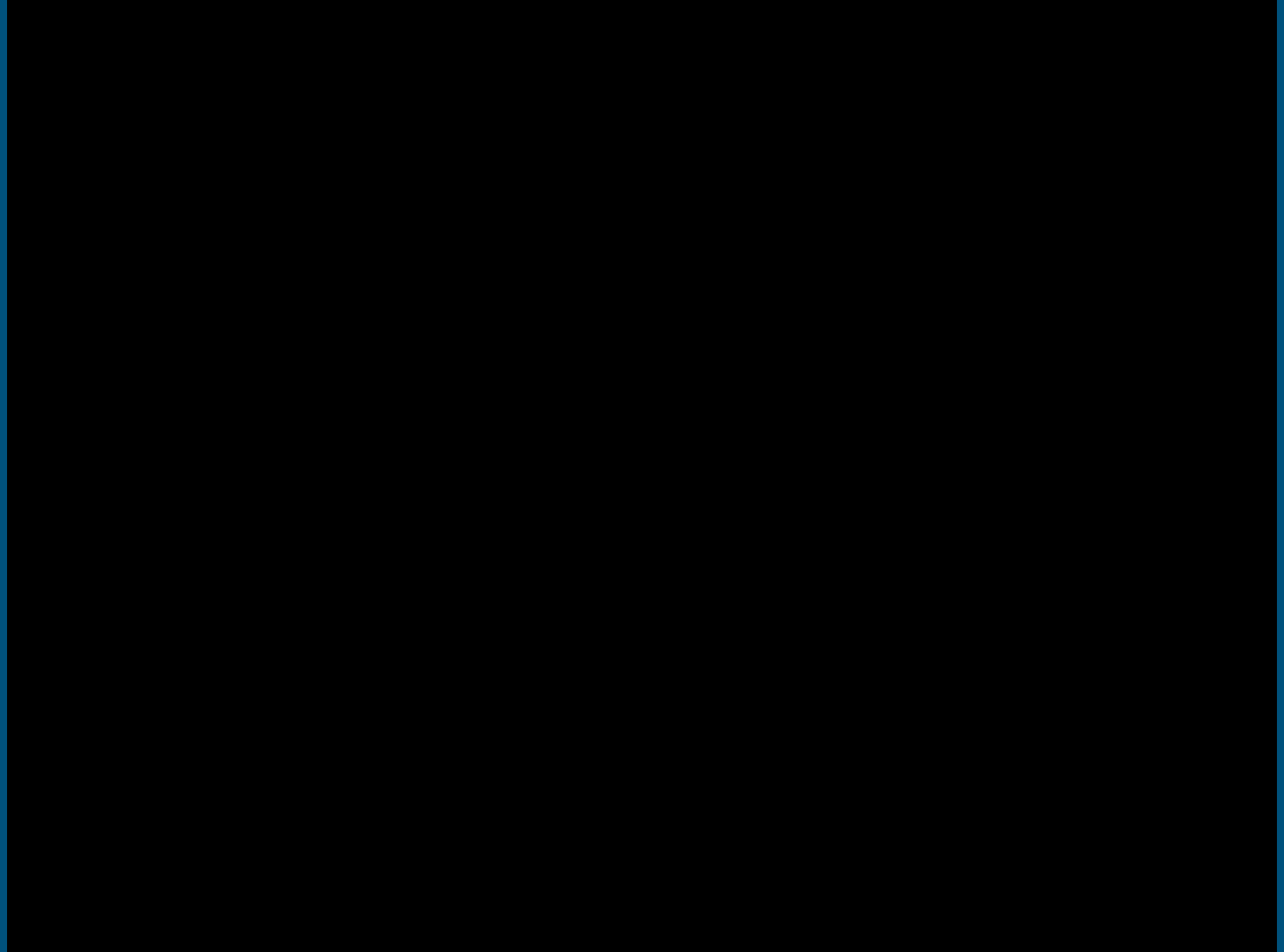
```cpp
#include <ros/ros.h>
#include <std_msgs/String.h>

ros::init(argc, argv, "/my_node");
ros::NodeHandle nh;

void callback(const std_msgs::String& msg)
{
  ROS_INFO("Hello, %s", msg.data.c_str());
}

ros::Publisher pub = nh.advertise<std_msgs::String>("/chatter", 1);
ros::Subscriber sub = nh.subscribe("/chatter", 1, chatterCallback);

std_msgs::String msg;
msg.data = "ROSCON";

ros::Rate r(1);
while (ros::ok())
{
  pub.publish(msg);
  r.sleep()
}
```

```javascript
const rosnodejs = require('rosnodejs');
const StringMsg = rosnodejs.require('std_msgs').msg.String;

await rosnodejs.initNode('/my_node');
const nh = rosnodejs.nh;

function callback(msg) {
  rosnodejs.log.info('Hello, %s', msg.data);
}


const pub = nh.advertise('/chatter', StringMsg);
const sub = nh.subscribe('/chatter', StringMsg, callback);

const msg = new StringMsg();
msg.data = 'ROSCON';

setInterval(function() {
  pub.publish(msg);
}, 1000);
```

demos

chatter/add two ints

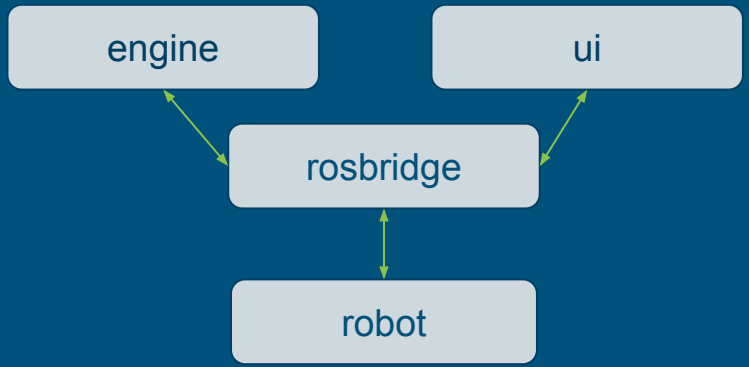


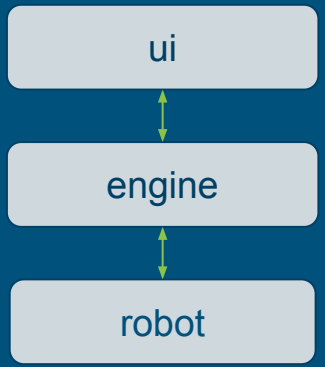


limits / future work

Limits

- JS number representation (no native 64bit integers, yet)
- JS time resolution (millisecond)
- Can’t run in the browser (rosbridge + roslibjs)
  - still need a bridge - ours ties in to our intra-engine event system

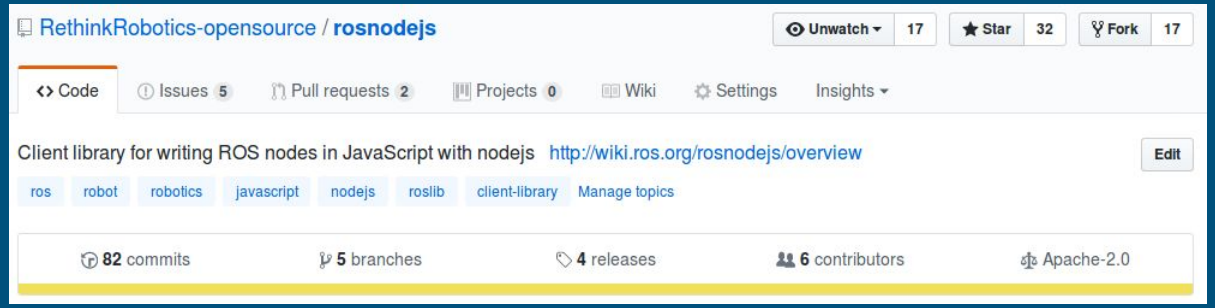




Future Work

- More ROS JS tools and utilities
  - tf, URDF, Math
  - platform agnostic (work in browser or backend engines)
    - roslibjs already has some
- rclnodejs
  - at Intel’s Open Source Technology Center (?)
  - (could share tools/utilities ?)

rosnodejs



https://github.com/RethinkRobotics-opensource/rosnodejs

Ian needs an intern