Ruby P2P Applications

With Ruby threads and **Journeta**

by **Preston Lee** and the **OpenRain** crew.

http://journeta.rubyforge.org/

http://www.openrain.com

Today

- What
- Why
- How
- Demos!

what journeta is
why you should care about
how journeta makes its magic
wrap up with demos which will make much more sense after discussing concepts

Journeta also Ruby processes on the same LAN to communicate by providing two primary services.

(MySpace for Ruby processes?)

- **Peer discovery.**
- **Object passing.**

the "what"
allows arbitrary ruby runtimes to talk to each other!

Endless Possibilities

- Pair programming.
- Team debugging.
- Ad hoc test clusters.
- File sharing.
- Configuration sharing.
- Directory discovery.
- Grid computing.
- Swarm downloading.
- Multiple monitors.
- Presence tracking.
- Instant messaging.
- Backups.
- Games.
- *<your idea>*

Design Goals

- Applications should talk.
  A lot.
- Low learning curve.
  Hide all the complicated stuff from the developer.
- Easy to integrate.
  A library, not framework.
- Enable collaboration.
  A new paradigm of real-time tools.
- No dependencies.
- Portable.
  OS X, Solaris, Linux, Windows.

Technical Overview

- Asynchronous peer discovery
  via UDP subnet broadcast.
- Asynchronous peer I/O
  via direct TCP connections.
- Messages are YAML
  serialized/deserialized to/from ordinary objects.
- Lots of threads
  to accomplish all this asynchronous madness.

Current Requirements

- OS X or Linux
  Yeah yeah... Windows coming soon.
- 1.8.7 (maybe 1.8.6) standard runtime.
  Would love a JRuby patch. :)

Inside The Magic

Journeta‘s inner workings only require knowledge of two things outside Ruby.

- Networking.
- Multi-threading.

this stuff is NOT required to use journeta!!!
..but demos will be much more meaningful after discussion.

Teh Internets In Review

- **Internet**
  - IP Address.
    Logical network node using the "Internet Protocol".
  - Port.
    A mailbox at a given IP address
  - UDP
    http://en.wikipedia.org/wiki/User_Datagram_Protocol
  - TCP
    http://en.wikipedia.org/wiki/Transmission_Control_Protocol
  - Subnet
    A range of network IP addresses which isolates stuff that needs to talk.
  - "Border"
    Router, bridge or gateway that connects your LAN to others.
- **Journeta**
  - Peer
    A logical node with a unique IP address port number combo. Each peer assigns itself a universally-unique identifier. (UUID)
  - Peer Handler
    Application provided code to do something with incoming peer data.

Ruby Green Threads

- **Pros**
  - Ruby-specific calls.
    Cool functions not available on other runtimes.
  - Consistent across platforms.
    Native thread semantics vary by OS.

http://spec.ruby-doc.org/wiki/Ruby_Threading

- **Cons**
  - Ruby-specific calls.
    Not portable to other runtimes.
  - Single CPU.
  - Time slicing not so hot.
    Easy to encounter starvation.
  - Not scalable.
    N threads running T milliseconds each == slow.
  - Slower than native.
    The kernel will *probably* always be able to outperform a user-space scheduler,. JRuby's native thread mapping approach, for example.
  - Pain to debug and test.
    Tends to be complicated and not well understood. I haven't figured out a great approach to this in Ruby yet.. anyone?

Handling Peer Events

```
require ‘journeta’
include Journeta
include Journeta::Common
include Journeta::Common::Shutdown

class MyHandler
  def call(msg)
    if !msg.nil? && msg.class == BasicMessage
      # do stuff with the data!
      puts msg.text
    end
  end
end

j = Journeta::Engine.new(
     :peer_handler => MyHandler.new, #optional
     :peer_port => (4000 + rand(1000)), # optional
     :groups => ['my_app']) # optional
stop_on_shutdown(j)
j.start
```

Sending Peer Events

```
# anything serializable to yaml can be sent!
m = BasicMessage.new
m.name = name
m.text = input
journeta.send_to_known_peers(m)


# who are my peers?
self.known_peers(true).each do |uuid, peer|
  # uuid is an int
  # peer is a PeerConnection
end


# send your friend (of uuid 42) some data
journeta.send_to_peer(42, {:stuff => [1, 2, 3]})
```

Demos!

- **Command line.**
  - network_status.rb
  - instant_messenger.rb
  - queue_server.rb
    queue_client.rb
  - peer_fuzzer.rb
- **GUI**
  - instant_messenger_gui.rb (requires wxruby)
  - Rails integration. (journeta_status_demo)
- **Fail Whale**
  - JRuby
  - Ruby 1.9

http://code.openrain.com/rails/journeta_status_demo/

Coming Soon

- Message encryption.
- Peer authentication.
- More callback types.
- JRuby?