

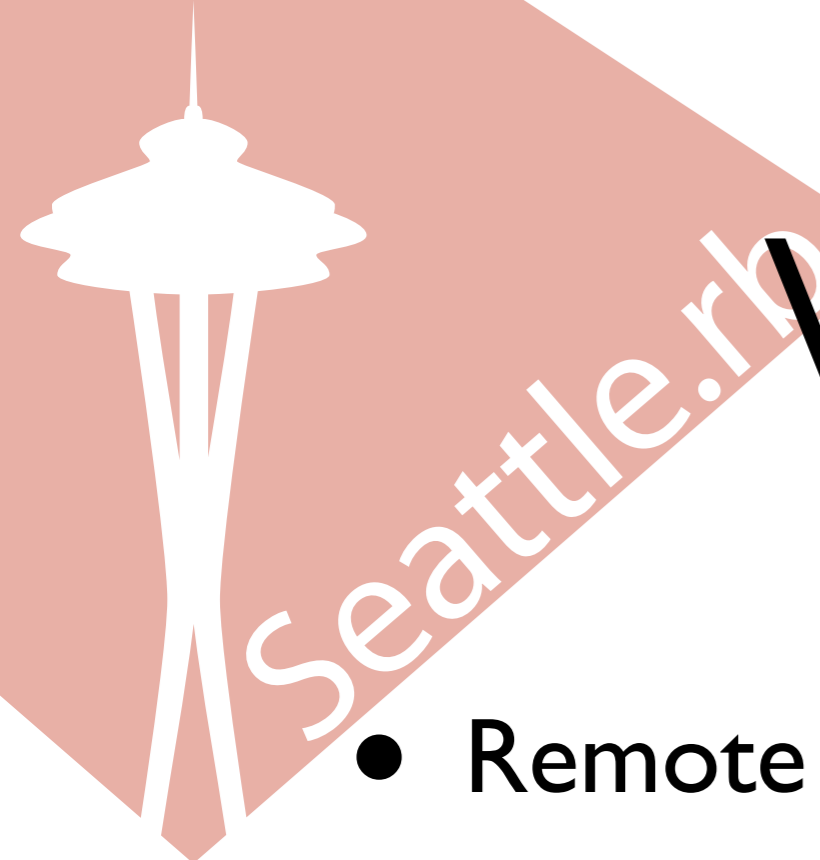


Distributed Ruby

An Introduction and Overview

Eric Hodel - drbrain@segment7.net

<http://blog.segment7.net>



What is DRb?

- Remote Method Invocation for Ruby
- Peer to peer
 - Client and server
- Multiplatform
- Multiprotocol
- More...



Why Do I Need DRb?

- Parallelism
- Distribute work
- Interprocess communication
- Monitoring




Example

```
# Server
require 'drb'
here = 'druby://127.0.0.1:10000'
DRb.start_service here, [1, 2, 'III', 4, 'five', 6]
DRb.thread.join
```

```
# Client
require 'drb'

there = 'druby://127.0.0.1:10000'
DRb.start_service
ro = DRbObject.new nil, there

p ro.size
a = ro.collect { |x| x + x }
p a
```



Example Server

```
require 'drb' # load DRb
```

```
here = 'druby://127.0.0.1:10000' # server URI
```

```
# start DRb
```

```
DRb.start_service here, # our local URI
```

```
    [1, 2, 'III', 4, 'five', 6]
```

```
    # our front object
```

```
DRb.thread.join # wait until DRb is done
```



Front Object

- Optional
- Default object
- Passed by reference
 - Only one copy

Example Client

```
require 'drb' # load DRb
there = 'druby://127.0.0.1:10000' #
DRb.start_service # start 1 drb server
ro = DRbObject.new_with_connection # connect to the server
# work with the remote
p ro.size
a = ro.collect { |x| x + x }
p a
```

How does it work?

Can't dump code!

ro.collect { |x| x + x }

Server

Client

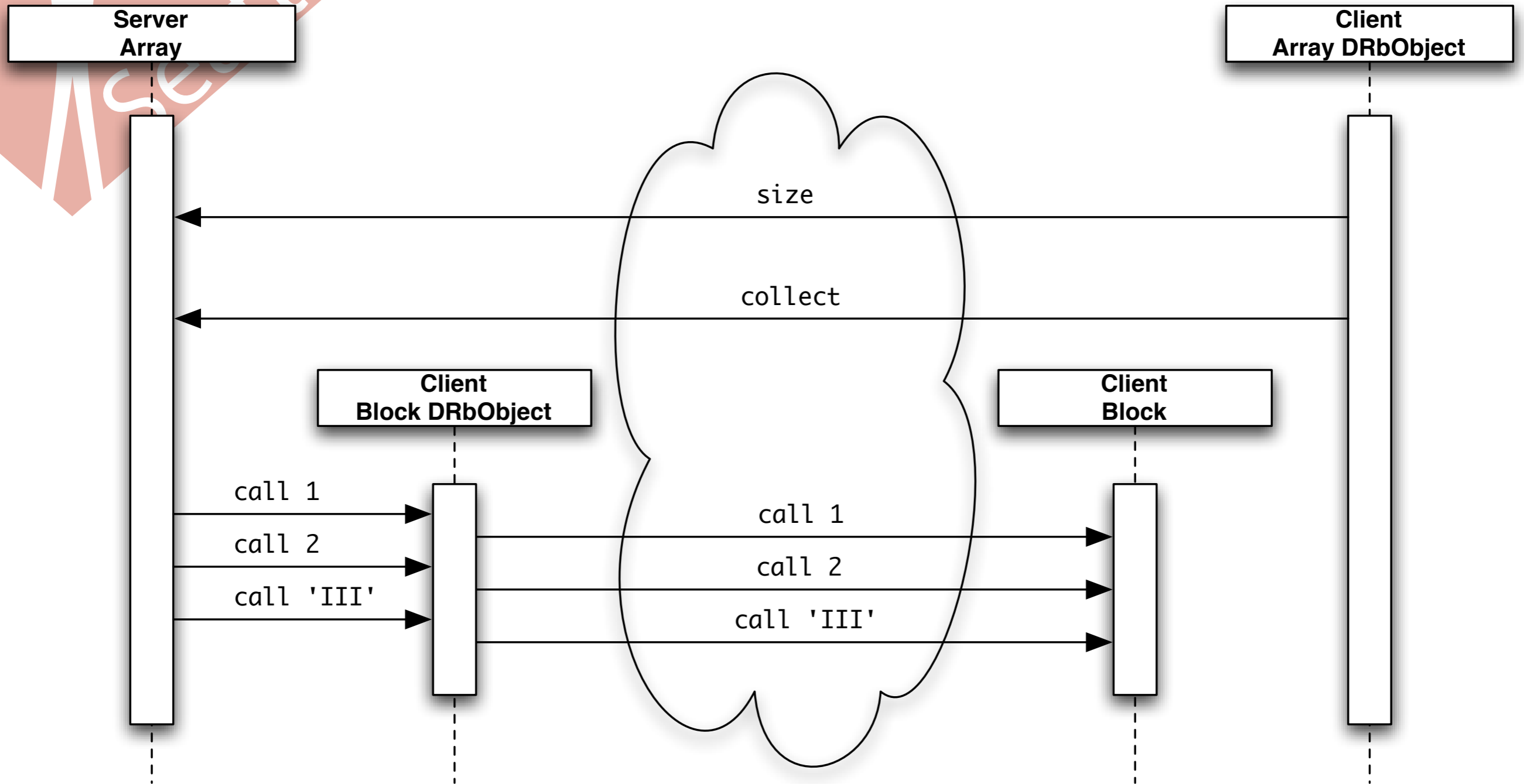
[1, 2, 'III',
4, 'five', 6]

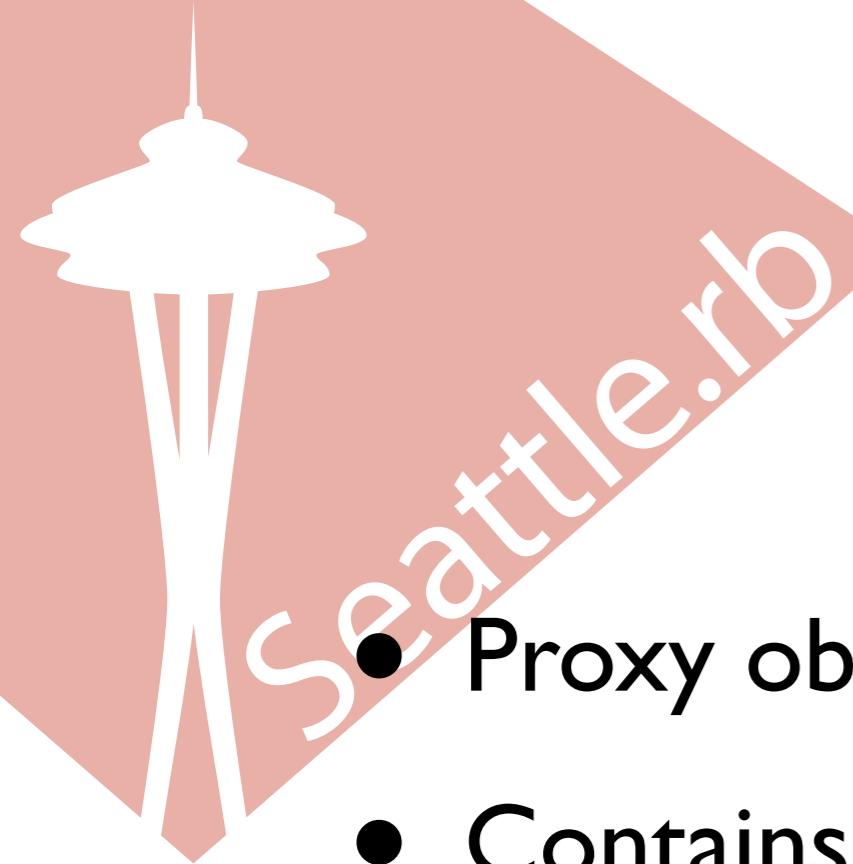
#<DRb::DRbObject>

#<DRb::DRbObject>

{ |x| x + x }

ro.collect { |x| x + x }





DRbObject

- Proxy object
- Contains URI and reference
 - Which server
 - What object
- Be aware of garbage collection
 - May be collected on server
 - Usually doesn't matter

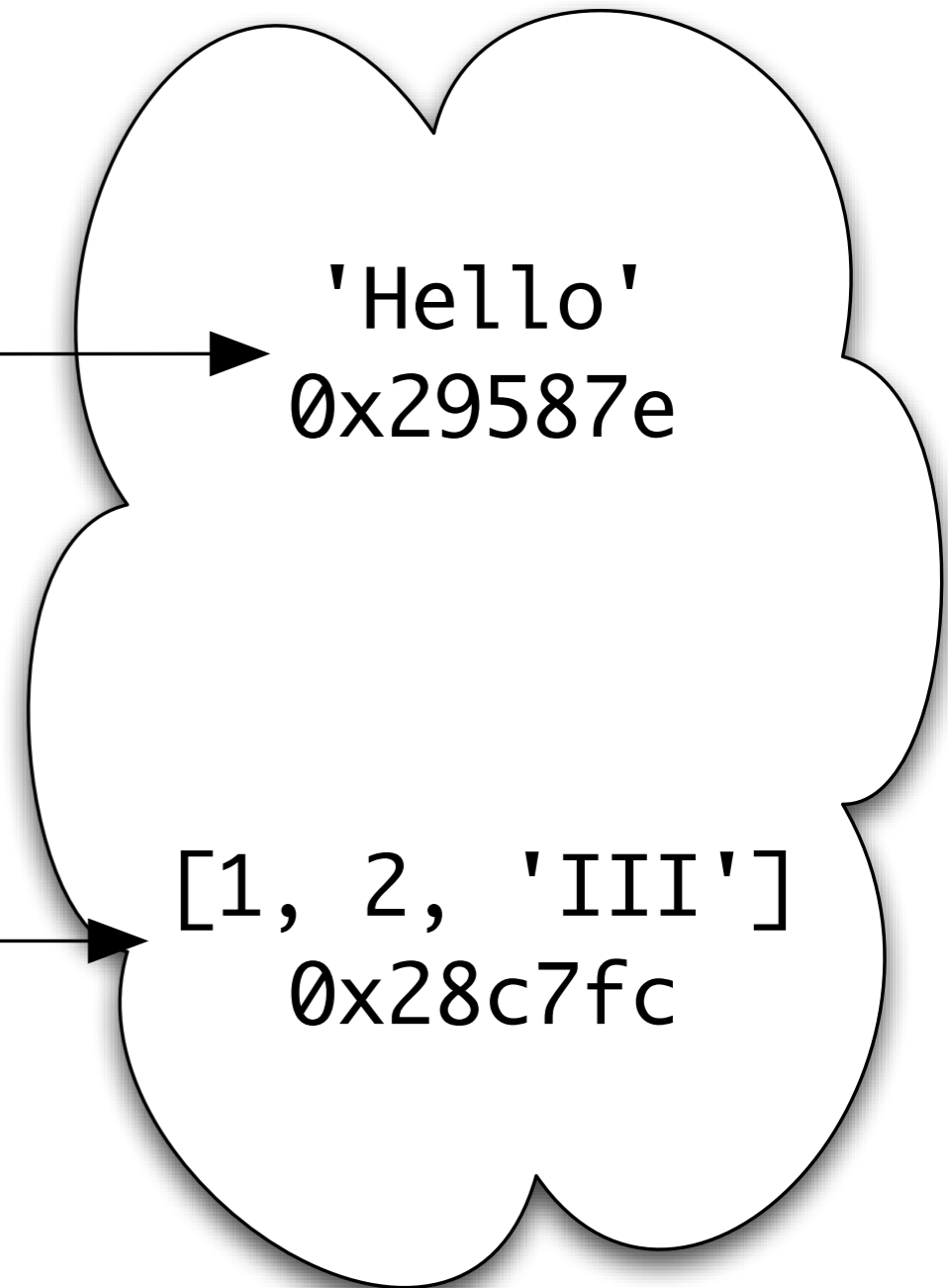


DRbObject

druby://127.0.0.1:10000

DRbObject	
uri	druby://127.0.0.1:10000
ref	0x29587e

DRbObject	
uri	druby://127.0.0.1:10000
ref	0x28c7f6





Id Conversion

- Converts reference into object on server
- DRbIdConv Default
 - object_id & ObjectSpace._id2ref
- Alternates:
 - TimerIdConv Keep alive
 - NamedIdConv Reference by name
 - GWIdConv Protocol gateway



TimerIdConv

```
require 'drb'  
require 'drb/timeridconv'  
  
id_conv = DRb::TimerIdConv.new 30  
  
DRb.default_id_conv = id_conv  
  
DRb.start_service  
  
# ...
```



NamedIdConv

- Found in `sample/drb/name.rb`
- Persistent objects
 - You must store your state
- Good example id converter



Remote Method Invocation

- Client
 - Marshal method arguments
 - Send message to server
 - Reference, method name, arguments
- Server
 - Receive message
 - Unmarshal arguments
 - Find local object from reference
 - Invoke method with arguments




DRbUndumped

- Default DRb operation
 - Pass by value
 - Must share code
- With DRbUndumped
 - Pass by reference
 - No need to share code



Automatic DRbObjects

- Module and Class
- IO
 - File, TCPSocket, UNIXSocket, etc.
- Blocks, Procs, Methods
- Thread, ThreadGroup
- Dir, File::Stat, Binding, Continuation



Using DRbUndumped

```
require 'drb'
```

```
class X  
  include DRb::DRbUndumped
```

```
  # ...
```

```
end
```

```
object = [1, 2, 'III', 4, 'five']  
object.extend DRb::DRbUndumped
```

The logo for Seattle 2019, featuring a stylized white umbrella on a red background with the text "Seattle 2019" written diagonally in white.

Why Use DRbUndumped?

- Big objects
- Singleton objects
- Lightweight clients
- Rapidly changing software



Connection Types

- TCP Sockets
- UNIX Sockets
- SSL Sockets



UNIX Socket Server

```
require 'drb'  
require 'drb/unix'
```

```
here = 'drbunix:///tmp/darray.sock'
```

```
DRb.start_service here, [1, 2, "III", 4, "five", 6]
```

```
DRb.thread.join
```

The logo for Seattle, featuring a white silhouette of the Space Needle tower against a red background. The word "Seattle" is written in a white, sans-serif font across the red background, partially overlapping the tower.

UNIX Socket Client

```
require 'drb'  
require 'drb/unix'
```

```
there = 'drbunix://tmp/darray.sock'
```

```
DRb.start_service
```

```
ro = DRbObject.new nil, there
```

```
p ro.size
```

```
a = ro.collect { |x| x + x }
```

```
p a
```

The logo for Seattle, featuring a white silhouette of the Space Needle tower against a red background, with the word "Seattle" written in a white, sans-serif font across the tower.

SSL Socket Server

```
require 'drb'  
require 'drb/ssl'
```

```
here = 'drbssl://127.0.0.1:10000'  
config = {}  
config[:SSLPrivateKey] = OpenSSL::PKey::RSA.new  
  File.read('server_keypair.pem')  
config[:SSLCertificate] = OpenSSL::X509::Certificate.new  
  File.read('certificate.pem')
```

```
DRb.start_service here, [1, 2, 'III', 4, 'five', 6],  
  config
```

```
DRb.thread.join
```

The logo for Seattle, featuring a white silhouette of the Space Needle tower against a red background. The word "Seattle" is written in a white, sans-serif font across the red background.

SSL Socket Client

```
require 'drb'  
require 'drb/ssl'
```

```
here = 'drbssl://127.0.0.1:10000'  
config = {}  
config[:SSLVerifyMode] = OpenSSL::SSL::VERIFY_PEER  
config[:SSLCACertificateFile] = 'ca_cert.pem'
```

```
DRb.start_service nil, nil, config
```

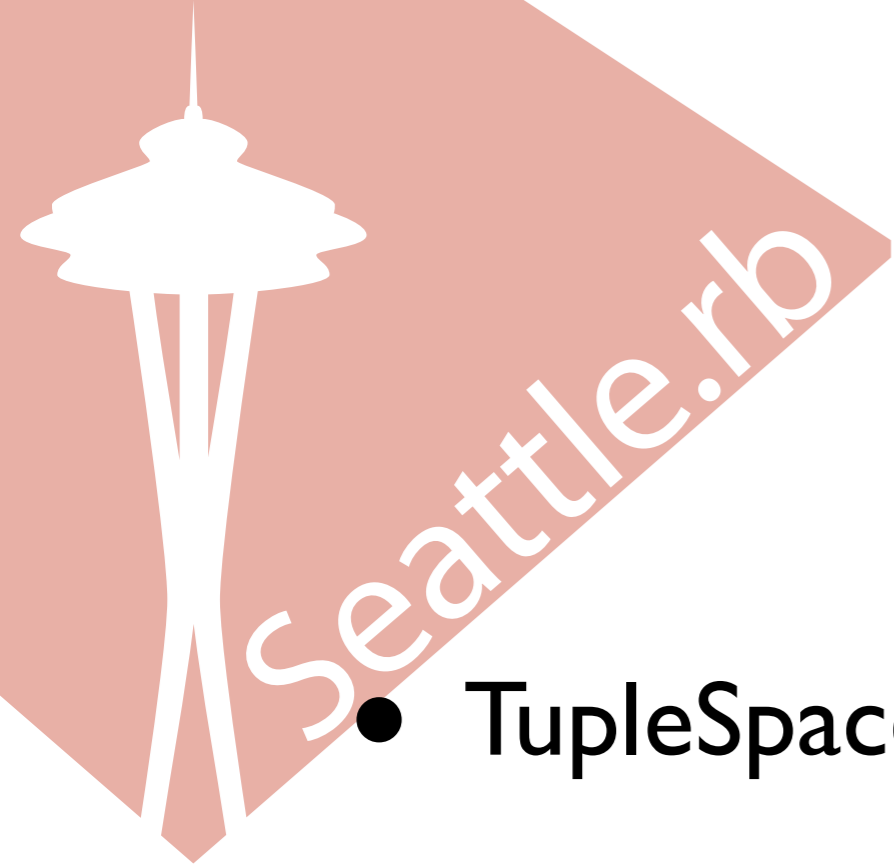
```
ro = DRbObject.new nil, there
```

```
p ro.size  
a = ro.collect { |x| x + x }  
p a
```



DRb + SSL

- Need to generate and distribute certificates
 - Use QuickCert (on the RAA)
- Lots more config options
 - Verify client and server
- Do it right
 - SSL is hard
 - Not magic



Rinda

- TupleSpace
 - Shared object space
 - Atomic access
- Ring
 - DNS for DRb
 - Automatically locate services



TupleSpace Operations

- Add a tuple
 - `ts.write [:Add, 1, 2]`
- Read a tuple matching a template
 - `ts.read [:Add, nil, nil]`
- Read all tuples matching a template
 - `ts.read_all [:Add, nil, nil]`
- Remove a tuple matching a template
 - `ts.take [:Add, nil, nil]`



TupleSpace Templates

```
include Rinda
```

```
# All match
```

```
Template.new([:foo, 5]).match      Tuple.new([:foo, 5])  
Template.new([:foo, nil]).match   Tuple.new([:foo, 5])  
Template.new([/11/]).match        Tuple.new(['hello'])  
Template.new([String]).match      Tuple.new(['hello'])
```

```
# No match
```

```
Template.new([:foo, 6]).match      Tuple.new([:foo, 5])  
Template.new([:foo, 6]).match      Tuple.new([:foo])  
Template.new([:foo, nil]).match    Tuple.new([:foo])  
Template.new([:foo]).match         Tuple.new([:foo, 5])
```



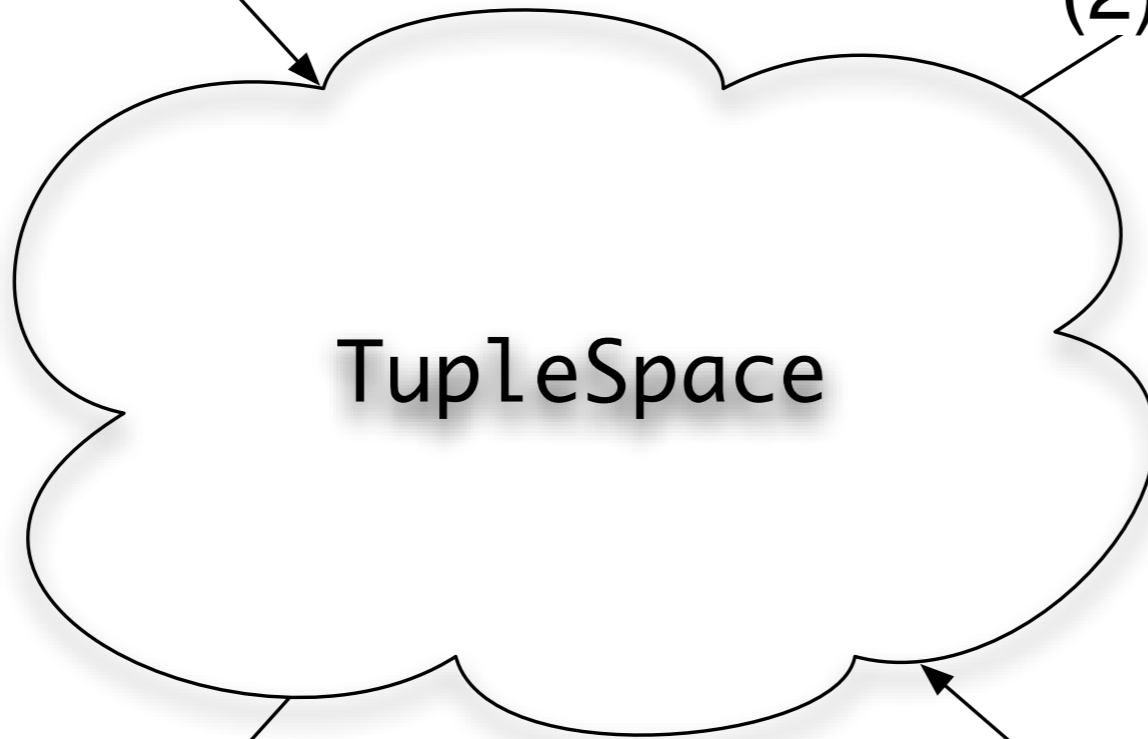
TupleSpace Usage

```
require 'drb'  
require 'rinda/tuplespace'  
  
ts = Rinda::TupleSpace.new  
  
ts.write [:Add, id, 1, 2]  
  
tuple = ts.take [:Add, nil, nil, nil]  
  
ts.write [:Sum, tuple[1], tuple[2] + tuple[3]]  
  
p ts.read_all([:Sum, nil, nil]).map { |t| t[2] }
```

TupleSpace Activity

[[:Add, id, 1, 2]]

(1) write



(2) take

[[:Add, id, 1, 2]]

Calculate!
1 + 2 = 3

(3) write

[[:Sum, id, 3]]

(4) read_all

[[[:Sum, id, 3]]]



Rinda::RingServer

```
require 'rinda/ring'  
require 'rinda/tuplespace'
```

```
# start DRb
```

```
DRb.start_service
```

```
# Create a TupleSpace to hold named services,
```

```
# start running
```

```
Rinda::RingServer.new Rinda::TupleSpace.new
```

```
DRb.thread.join
```

A logo for Seattle, featuring a white silhouette of the Space Needle tower against a red background. The word "Seattle" is written in a white, sans-serif font across the red background.

Rinda::RingProvider

```
require 'rinda/ring'  
require 'rinda/tuplespace'  
require 'thread'
```

```
DRb.start_service
```

```
queue = Queue.new
```

```
# Registers [:name, :WorkQueue, queue, 'Work Queue']  
provider = Rinda::RingProvider.new :WorkQueue, queue,  
                                   'Work Queue'
```

```
provider.provide  
DRb.thread.join
```



Rinda::RingFinger

```
require 'rinda/ring'
```

```
DRb.start_service
```

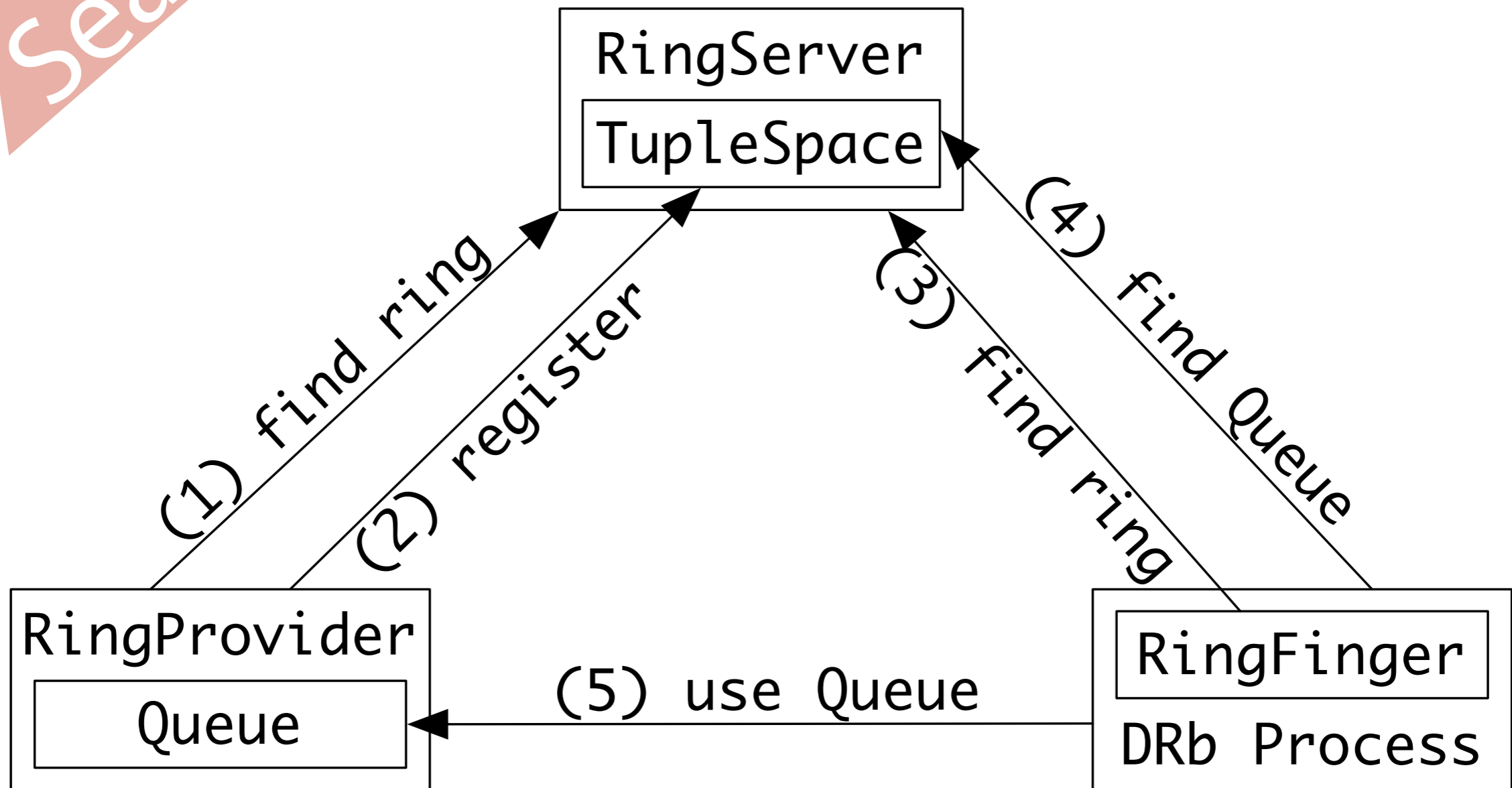
```
ring_server = Rinda::RingFinger.primary
```

```
tuple = ring_server.read [:name, :WorkQueue, nil, nil]
```

```
queue = tuple[2]
```

```
queue.push [1, 2]
```

Rinda::Ring Interactions





Hardening DRb

- Can't completely
- ACL
- \$SAFE
- default_load_limit
- SSL
- Don't expose to untrusted parties



Why Not?

- Access arbitrary objects
- Run arbitrary methods
 - Infinite loops
 - Deadlocks

The logo for Seattle, featuring a white silhouette of the Space Needle tower against a red background. The word "Seattle" is written in a white, sans-serif font across the red background.

Access Control List

```
require 'drb'
```

```
require 'drb/acl'
```

```
acl = ACL.new %w[  
  deny all  
  allow 127.0.0.1  
]
```

```
DRb.install_acl acl
```

```
DRb.start_service # ...
```

A logo in the top-left corner featuring a stylized white street lamp on a red background with the word "Seattle" written in white script.

`$$SAFE` & `default_load_limit`

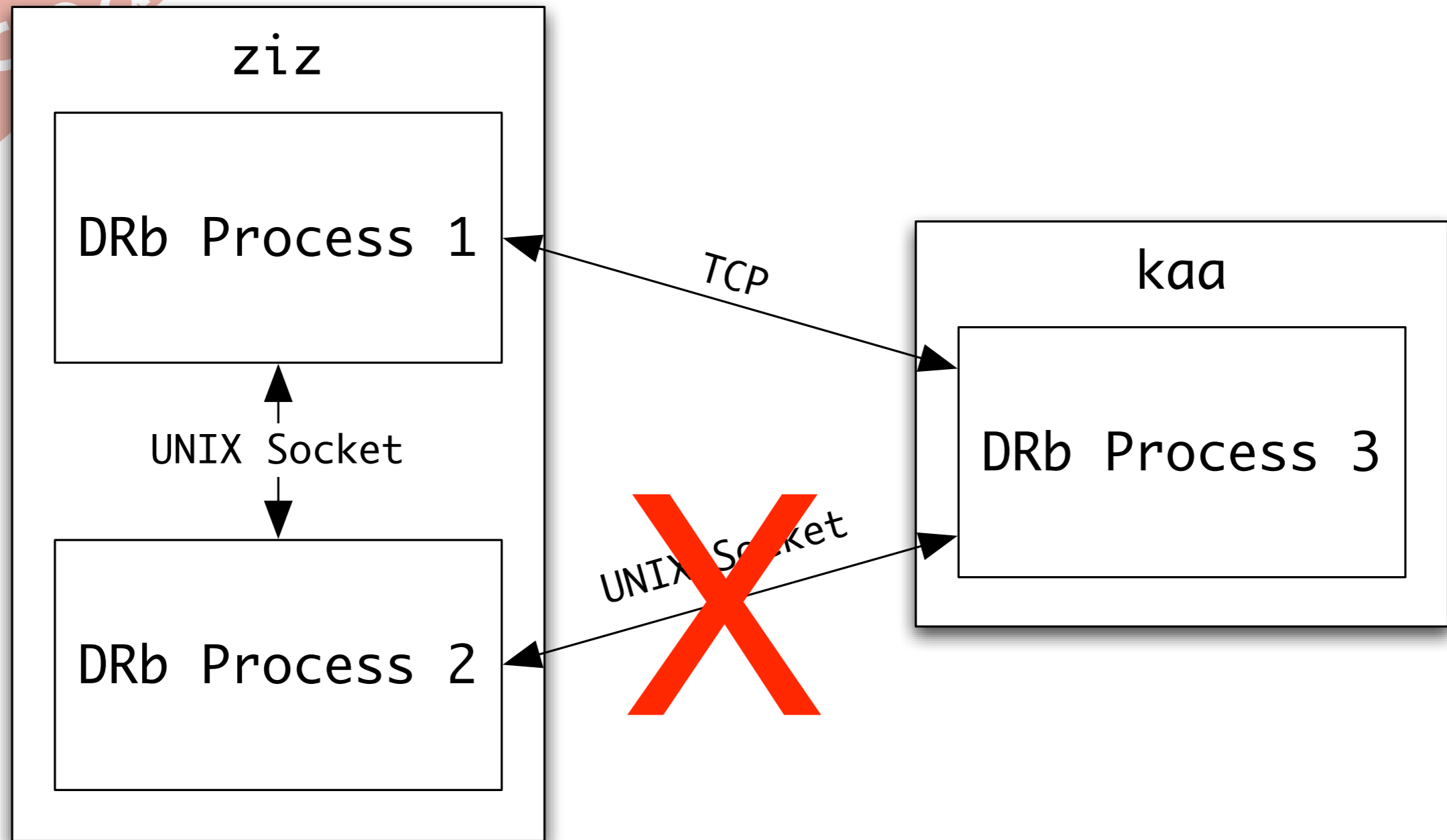
- `DRb::DRbServer.default_safe_level`
 - Sets `$$SAFE` for remote method invocation
 - Protects from some dangerous operations
 - <http://www.rubycentral.com/book/taint.html>
- `DRb::DRbServer.default_load_limit`
 - Maximum message size accepted by server
 - Defaults to 25MB



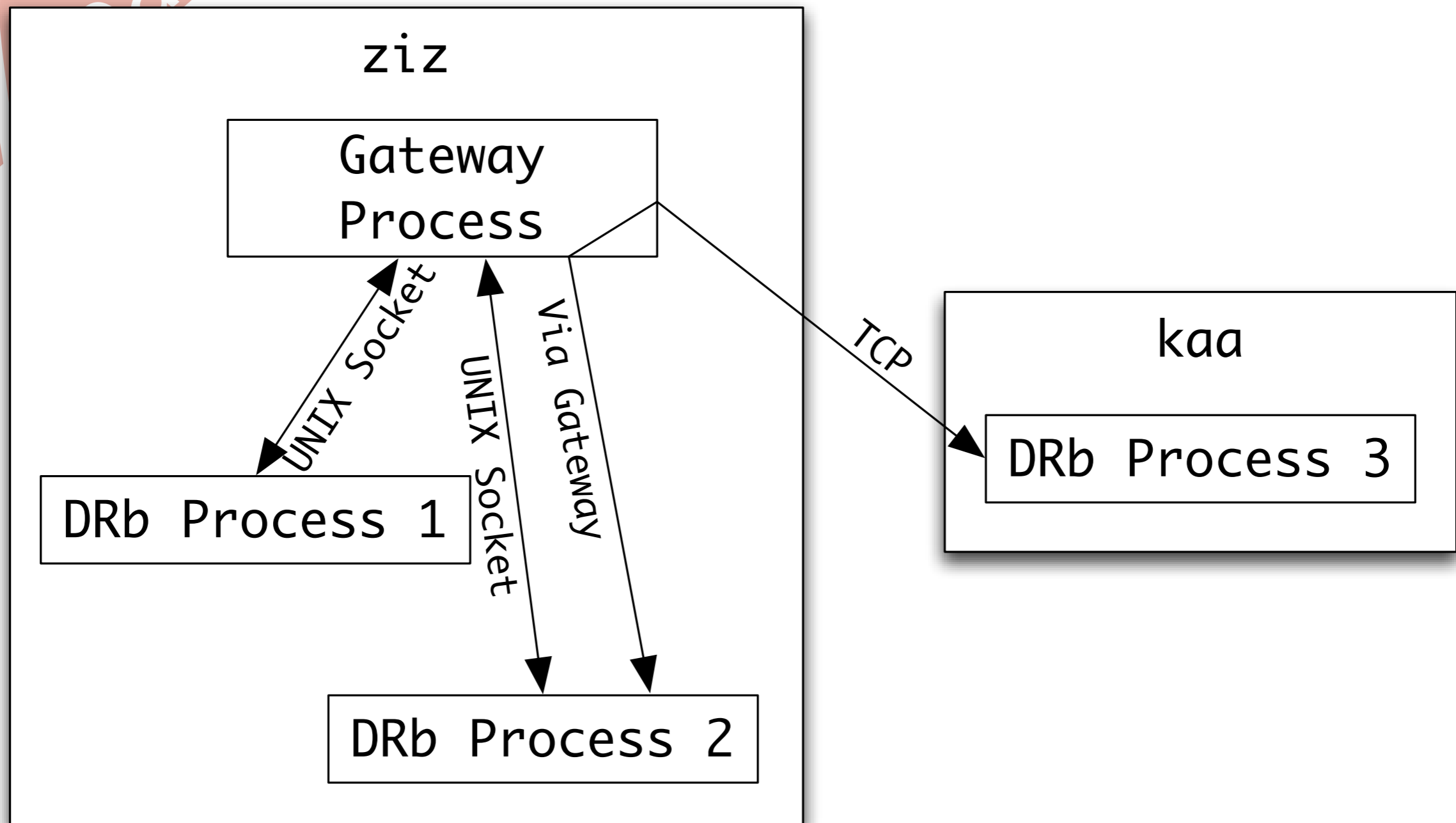
Advanced Features

- Gateway Id Conversion
 - Cross-protocol communication
 - DRb over SSH
- DRbObservable
- sample/drbrb
 - In Ruby tarball

Gateway Conversion



Gateway Conversion



Gateway Conversion

DRbObject	
uri	druby://ziz:10000
ref	

GWIdConv
Unwraps

DRbObject	
uri	drbunix://tmp/drb.sock
ref	0x29587e

Re-Dispatch
to Target



Gotchas

- DRb relies on accurate DNS
 - IPv6 needs DNS too!
- Sharing
 - Is it OK to copy?
- Peer to peer
 - Connects back to other processes
- Garbage Collection



Further Reading



Question
Time!

- dRubyによる分散・Webプログラミング
 - (Distributed and Web Programming with dRuby)
- <http://www2a.biglobe.ne.jp/~seki/ruby/druby.html>
- via <http://excite.co.jp/world/english/web>
- <http://www.chadfowler.com/ruby/drb.html>
- <http://segment7.net/projects/ruby/drb/>