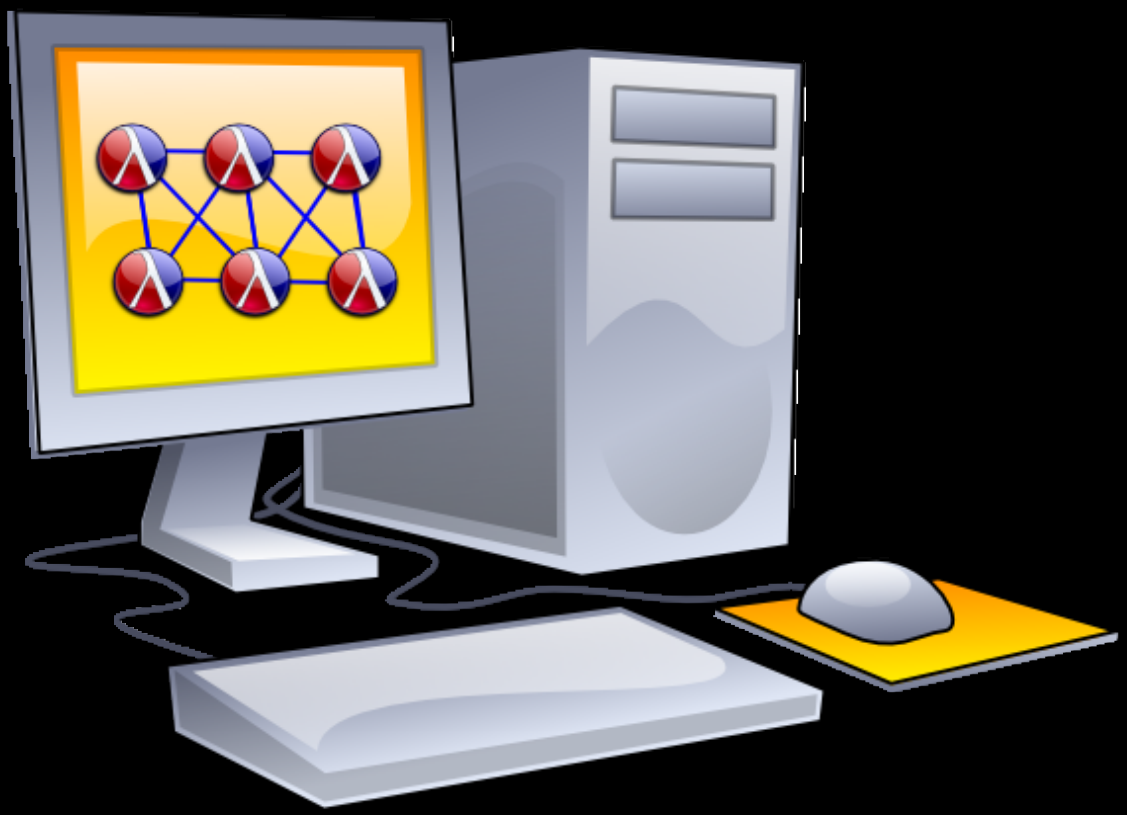
Distributed Places

Kevin Tew
University of Utah



Places Example

```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define p (dynamic-place (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```

Places Example

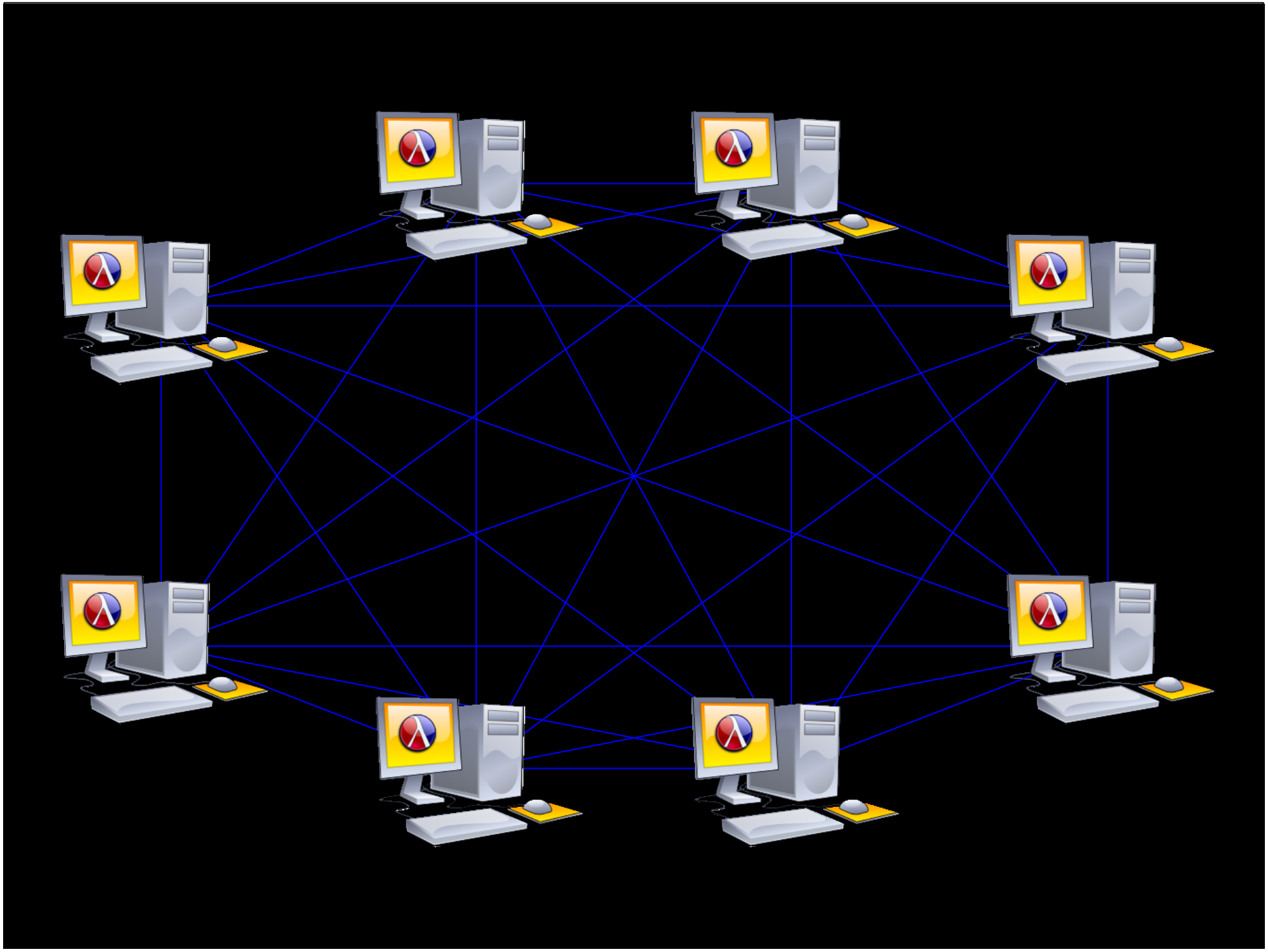
```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define p (dynamic-place (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```

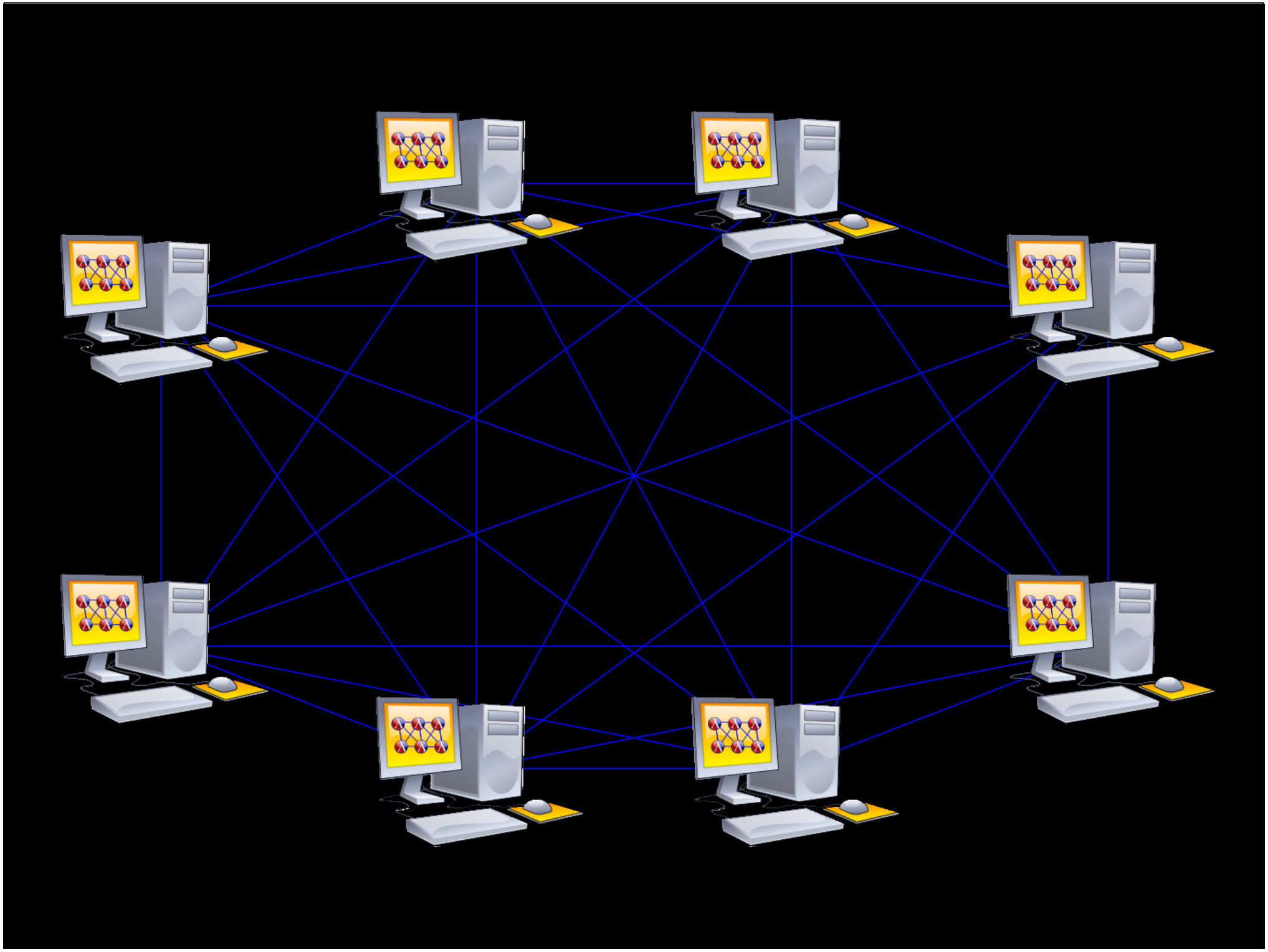
Places Example

```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define p (dynamic-place (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```

Places Example

```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define p (dynamic-place (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```





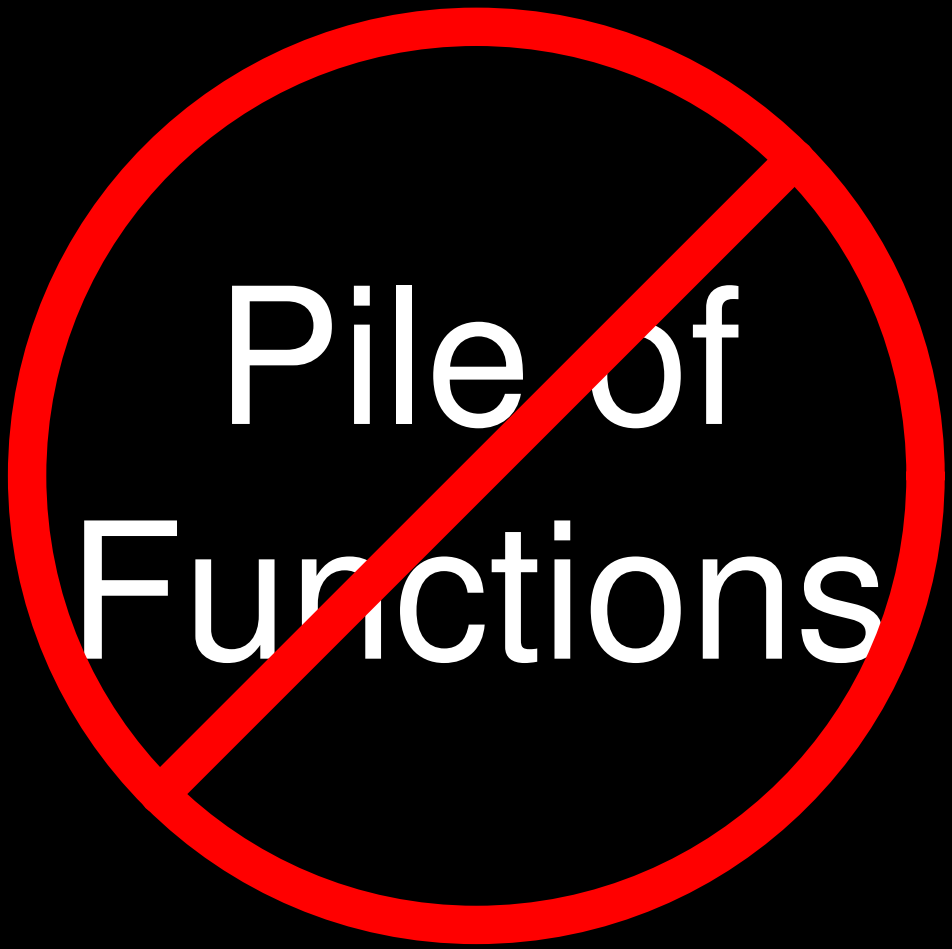

Distributed Places

Distributed Places

Pile of Functions

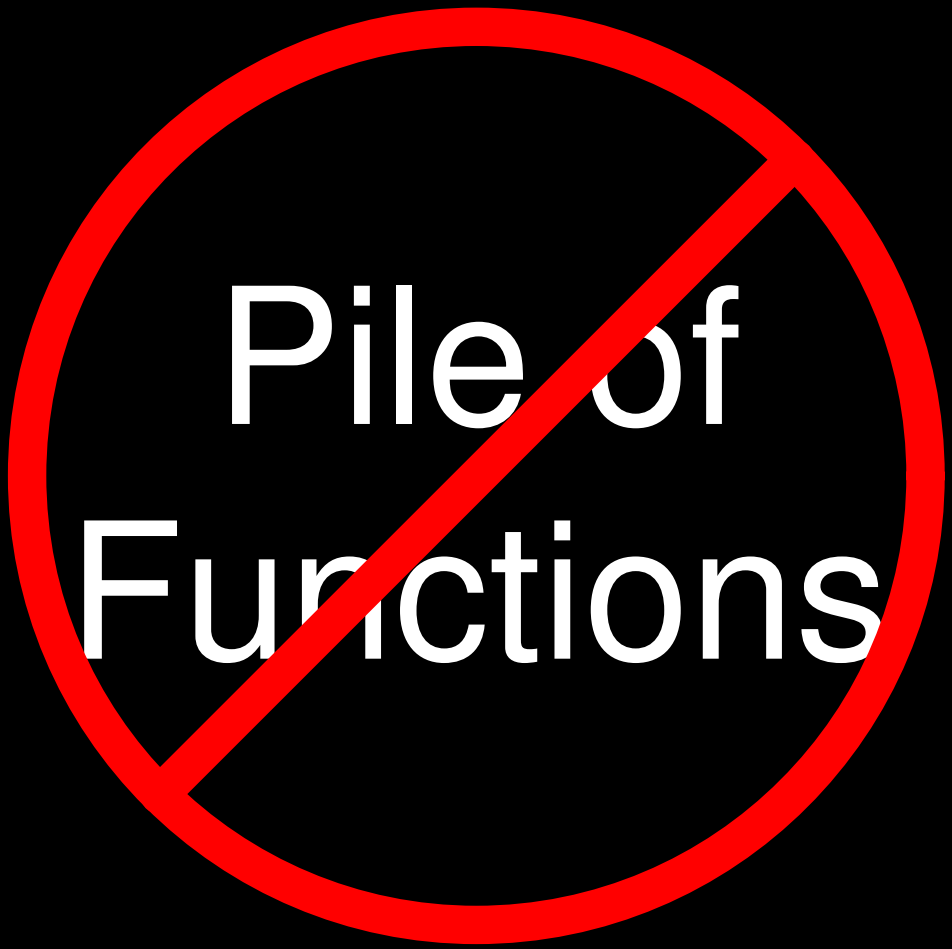
FFI

Distributed Places

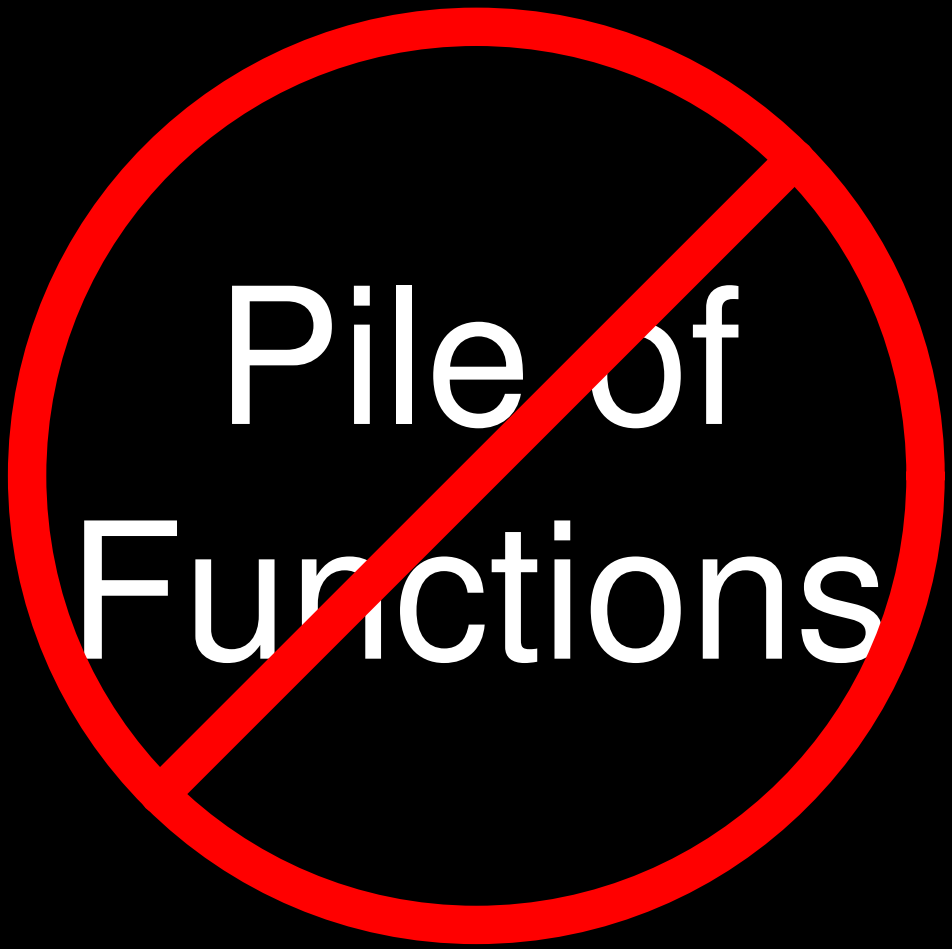


FFI

Distributed Places



Distributed Places



Language Extension

Distributed Places Example

Places

```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define p (dynamic-place (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```

Distributed Places

```
(require racket/place
         racket/place/distributed)
(provide hello-world)

(define (hello-world ch)
  (printf/f "hello-world received: ~a\n"
            (place-channel-get ch))
  (place-channel-put ch "Hello World\n")
  (printf/f "hello-world sent: Hello World\n"))

(module+ main
  (define n (create-place-node "host2"
                               #:listen-port 6344))
  (define p (dynamic-place #:at n
                           (quote-module-path "..")
                           'hello-world))
  (place-channel-put p "Hello")
  (printf/f "main received: ~a\n"
            (place-channel-get p))
  (place-wait p))
```

Distributed Places

API

create-place-node

```
(create-place-node
  hostname
  [#:listen-port port
   #:racket-path racket-path
   #:ssh-bin-path ssh-path
   #:distributed-launch-path launcher-path
   #:use-current-ports use-current-ports])
  → (is-a?/c remote-node%)
  hostname : string?
  port : port-no? = DEFAULT-ROUTER-PORT
  racket-path : string-path? = (racket-path)
  ssh-path : string-path? = (ssh-bin-path)
  launcher-path : string-path?
    = (path->string distributed-launch-path)
  use-current-ports : boolean? = #t
```

dynamic-place

```
(dynamic-place
  module-path
  start-name
  [#:at node
   #:named named]) → place?
    module-path : (or/c module-path? path?)
    start-name : symbol?
    node : (or/c #f remote-node%) = #f
    named : (or/c #f string?) = #f
```

Implementing Distributed Places

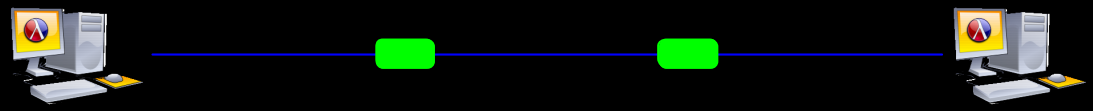
Implementing Distributed Places

Spawn Remote Processes

Implementing Distributed Places

Spawn Remote Processes

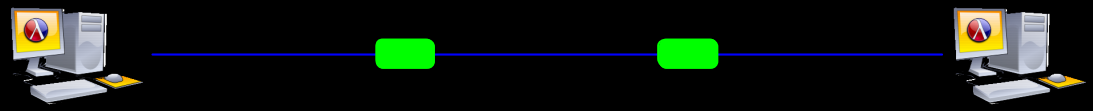
Send Place Messages Across TCP Sockets



Implementing Distributed Places

Spawn Remote Processes

Send Place Messages Across TCP Sockets



Integrate with Racket Concurrency Primitives

```
(sync place-channel-1 socket place-descriptor)
```

Higher-level Distributed Frameworks

RPC Server

Definition - tuple.rkt

```
(require racket/match
         racket/place/define-remote-server)

(define-named-remote-server tuple-server

  (define-state h (make-hash))

  (define-rpc (set k v)
    (hash-set! h k v)
    v)

  (define-rpc (get k)
    (hash-ref h k #f))

  (define-cast (hello)
    (printf "Hello from define-cast\n")
    (flush-output)))
```

Use

```
(require racket/place/distributed
         racket/place
         "tuple.rkt")

(module+ main
  (define c (connect-to-named-place remote-node
                                    'tuple-server))
  (tuple-server-hello c)
  (displayln (tuple-server-set c "user0" 100))
  (displayln (tuple-server-get c "user0")
  (tuple-server-hello)))
```

MPI

```
(require racket/place/distributed
         racket/place/distributed/rmpi)

(provide prod-id-place)

(define (prod-id-place ch)
  (define-values (comm args tc) (rmpi-init ch))
  (define sum (rmpi-allreduce comm
                              *
                              (+ 1 (rmpi-id comm))))
  (printf/f "~a - ~a\n" (rmpi-id comm) sum)
  (sleep 3)

  (rmpi-finish comm tc))
```

```
(module+ main
  (define args null)
  (time
    (rmpi-launch
      (rmpi-build-default-config
        #:mpi-module (quote-module-path "..")
        #:mpi-func   'prod-id-place
        #:mpi-args    args)

        (list (list "localhost" 6341 'kmeans0 0)
              (list "localhost" 6342 'kmeans1 1)
              (list "localhost" 6343 'kmeans2 2)
              (list "localhost" 6344 'kmeans3 3)
              (list "localhost" 6345 'kmeans4 4)
              (list "localhost" 6346 'kmeans5 5)
              (list "localhost" 6347 'kmeans6 6)
              (list "localhost" 6348 'kmeans7 7)))))
```

Map Reduce

```
(define/provide (mapper kvs)
  (for/first ([kv kvs])
    (match kv
      [(cons k v)
       (with-input-from-file
         v
         (lambda ()
           (let loop ([result null])
             (define l (read-line))
             (if (eof-object? l)
                 result
                 (loop (cons (cons l 1)
                             result))))))])))

(define/provide (reducer kvs)
  (for/list ([kv kvs])
    (match kv
      [(cons k v)
       (cons k (list (for/fold ([sum 0])
                               ([x v])
                       (+ sum x))))])))

(define/provide (outputer kvs)
  (displayln
    (for/fold ([sum 0]) ([kv kvs])
      (printf "~a - ~a\n" (car kv) (cadr kv))
      (+ sum (cadr kv)))))
```

```
(define config (list (list "host2" 6430)
                     (list "host3" 6430)))

(define tasks (list (list (cons 0 "/tmp/w0"))
                    (list (cons 1 "/tmp/w1"))
                    ...))

(define workers (make-map-reduce-workers config))

(map-reduce workers config tasks
            (list (quote-module-path "..") 'mapper)
            (list (quote-module-path "..") 'reducer)
            #:outputer (list (quote-module-path "..")
                             'outputer))
```

Conclusions

- Language extension enables concise implementation and use
- Reuses dynamic-place syntax and place channels
- Foundation for building higher-level parallel and distributed frameworks
- Roadmap for dynamic language implementers to follow