Università della Svizzera Italiana

Faculty of Informatics

S
A Scripting Language for High-Performance RESTful Web Services

Daniele Bonetta
Achille Peternier, Cesare Pautasso, Walter Binder
Faculty of Informatics
University of Lugano - USI
Switzerland

http://sosoa.inf.usi.ch/s

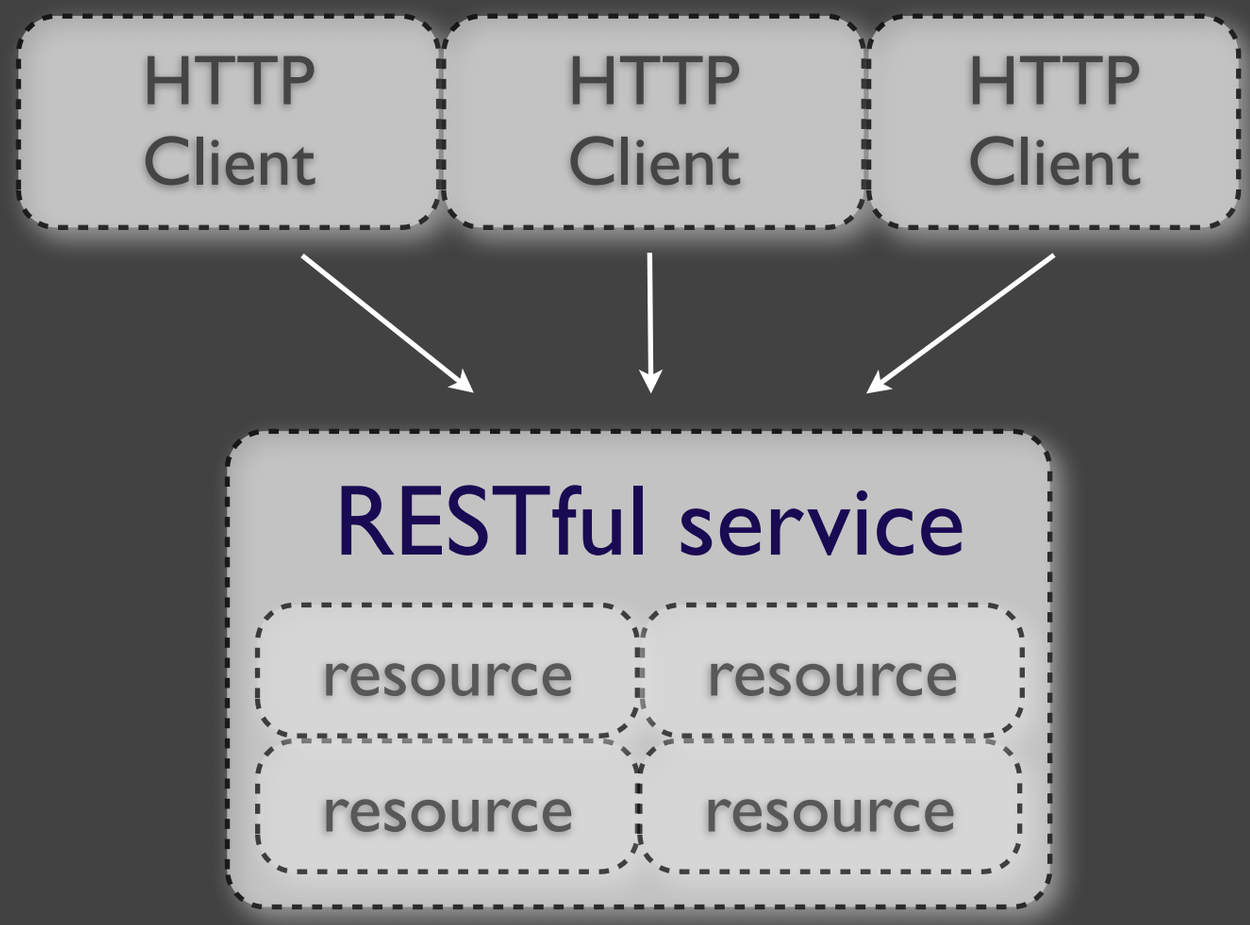
RESTful Web Services

Università della Svizzera Italiana
Faculty of Informatics

Services using the HTTP protocol

REST Architectural Style

- Client-Server Architecture
- Stateless Interaction
- Resources (Stateful, URIs)
- Uniform Interface (GET,PUT,DELETE,POST, ...)



Goal

Develop and compose RESTful services that *scale*

Goal

Develop and compose RESTful services that *scale*

#clients

Goal

Develop and compose RESTful services that *scale*

#clients

#cores

Goal

Develop and compose RESTful services that *scale*

#clients

#cores

using safe implicit parallelism

Node.JS

Asynchronous Event-loop based solution

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Asynchronous nonblocking I/O ✔

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Asynchronous nonblocking I/O ✔

No locks/synchronization ✔

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Asynchronous nonblocking I/O ✔

No locks/synchronization ✔

Sequential composition: nested callbacks ✘

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Asynchronous nonblocking I/O ✔

No locks/synchronization ✔

Sequential composition: nested callbacks ✘

Callbacks need to be short: never block ✘

Node.JS

Asynchronous Event-loop based solution

```
on('request', function(req,res) {
    // ...
})

on('response', function(req,res) {
    // ...
})
```

Asynchronous nonblocking I/O ✔

No locks/synchronization ✔

Sequential composition: nested callbacks ✘

Callbacks need to be short: never block ✘

Master worker parallelism ✘

Università della Svizzera Italiana

Faculty of Informatics

The S Scripting Language

A DSL for RESTful Web Services Development and Composition

Stateful Services

The S Scripting Language

A DSL for RESTful Web Services Development and Composition

Scalable Stateful Services

The S Scripting Language

A DSL for RESTful Web Services Development and Composition

S: Design Principles

Università della Svizzera Italiana
Faculty of Informatics

High-level architectural abstractions

Simple, clean programming model

S: Design Principles

High-level architectural abstractions

Services compose resources
No threads/processes, no synchronization/locks
Stateful services

Simple, clean programming model

S: Design Principles

High-level architectural abstractions

Services compose resources
No threads/processes, no synchronization/locks
Stateful services

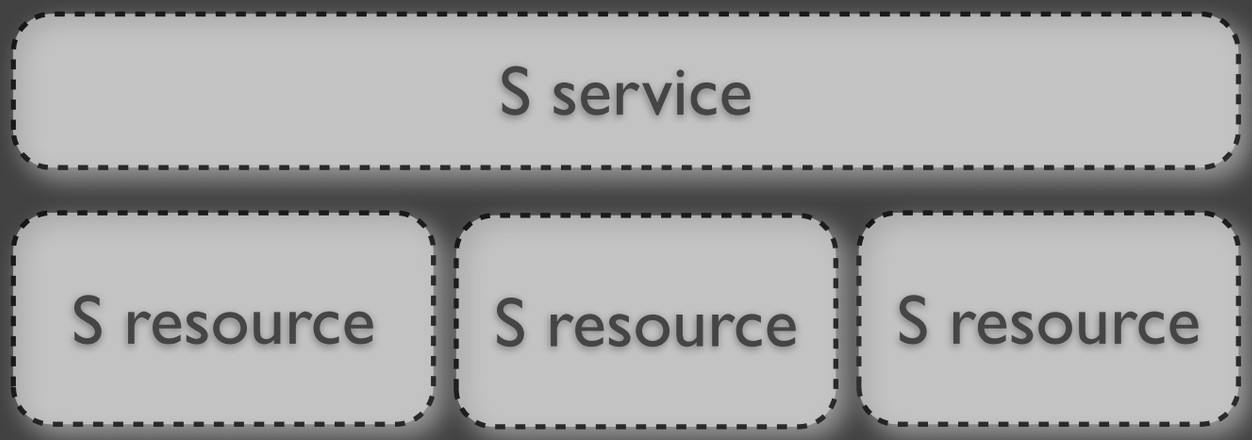
Simple, clean programming model

Allow blocking function calls
Synchronous I/O + parallel constructs
Uniform interface and URIs

S: Architecture

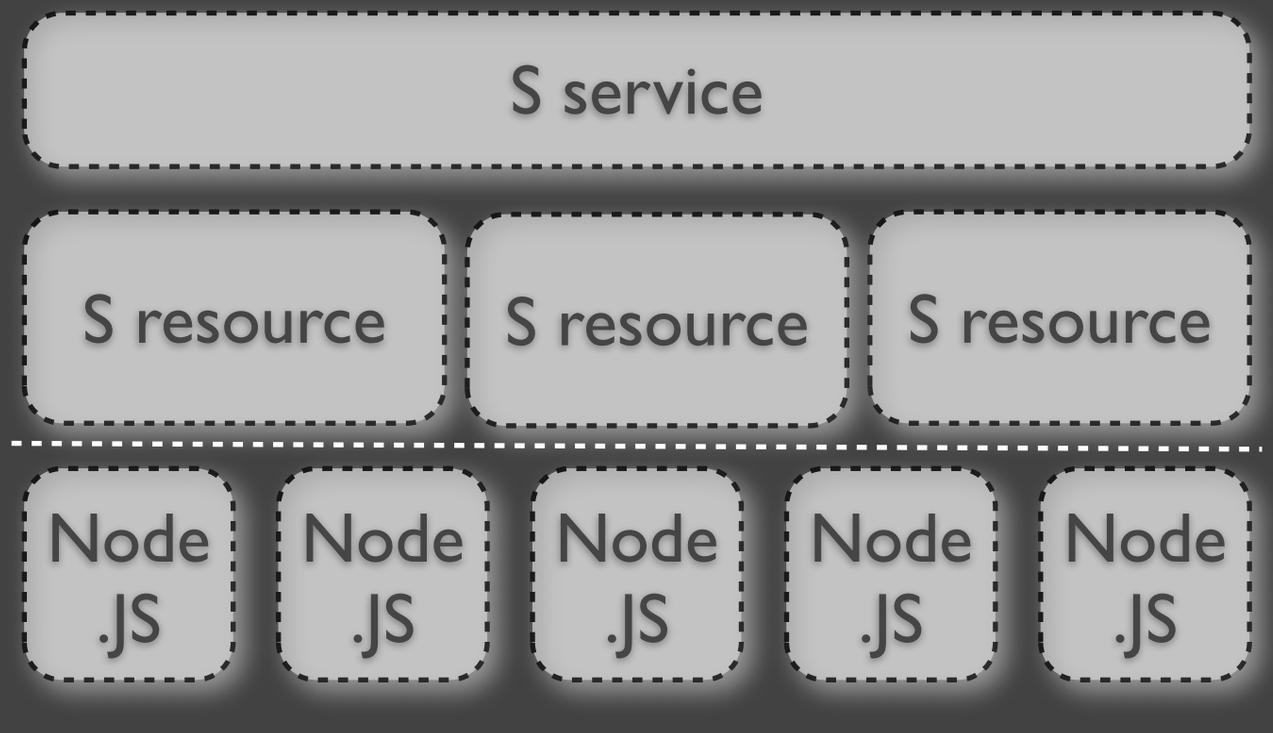
S service

S: Architecture

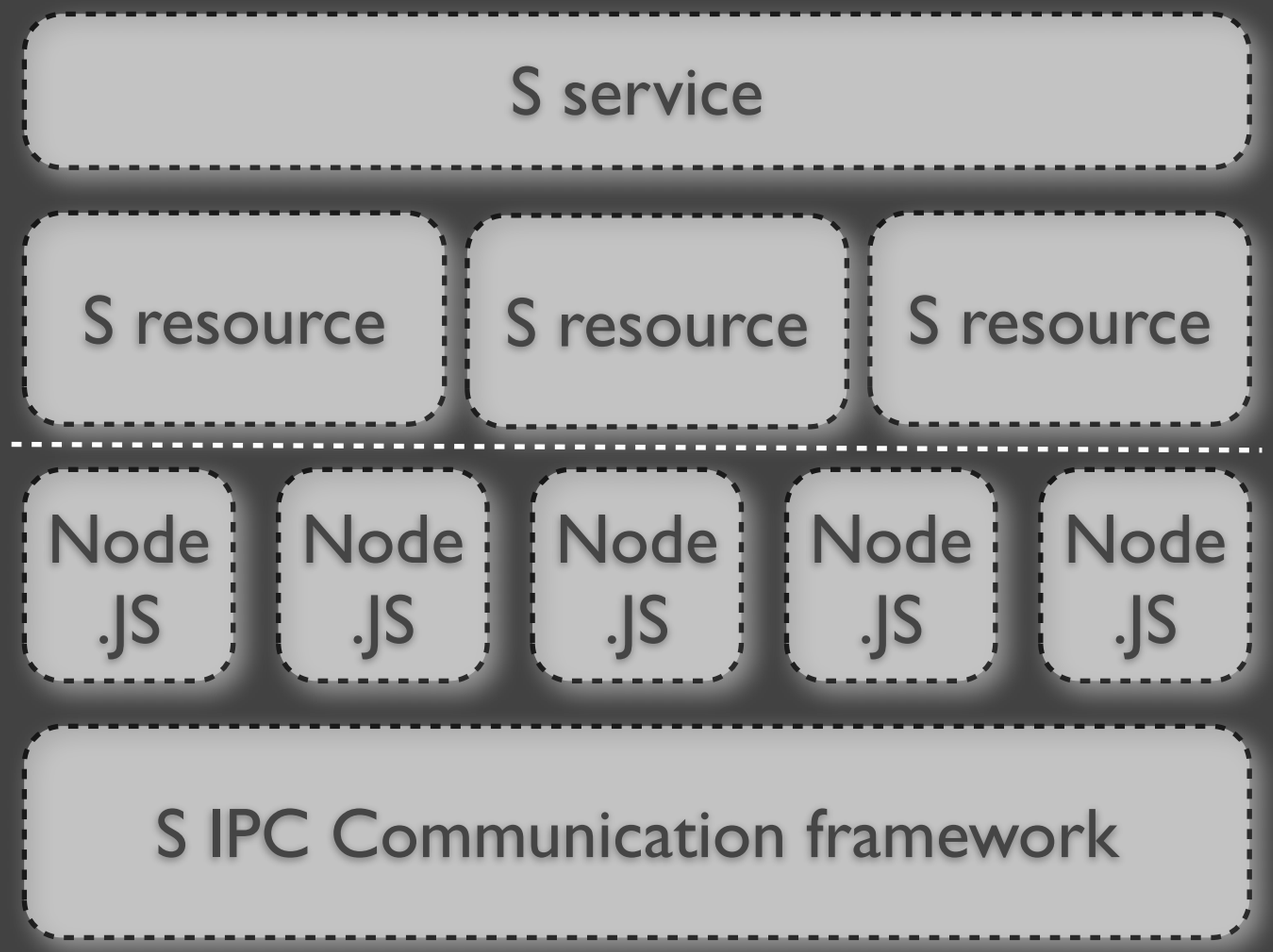


S: Architecture

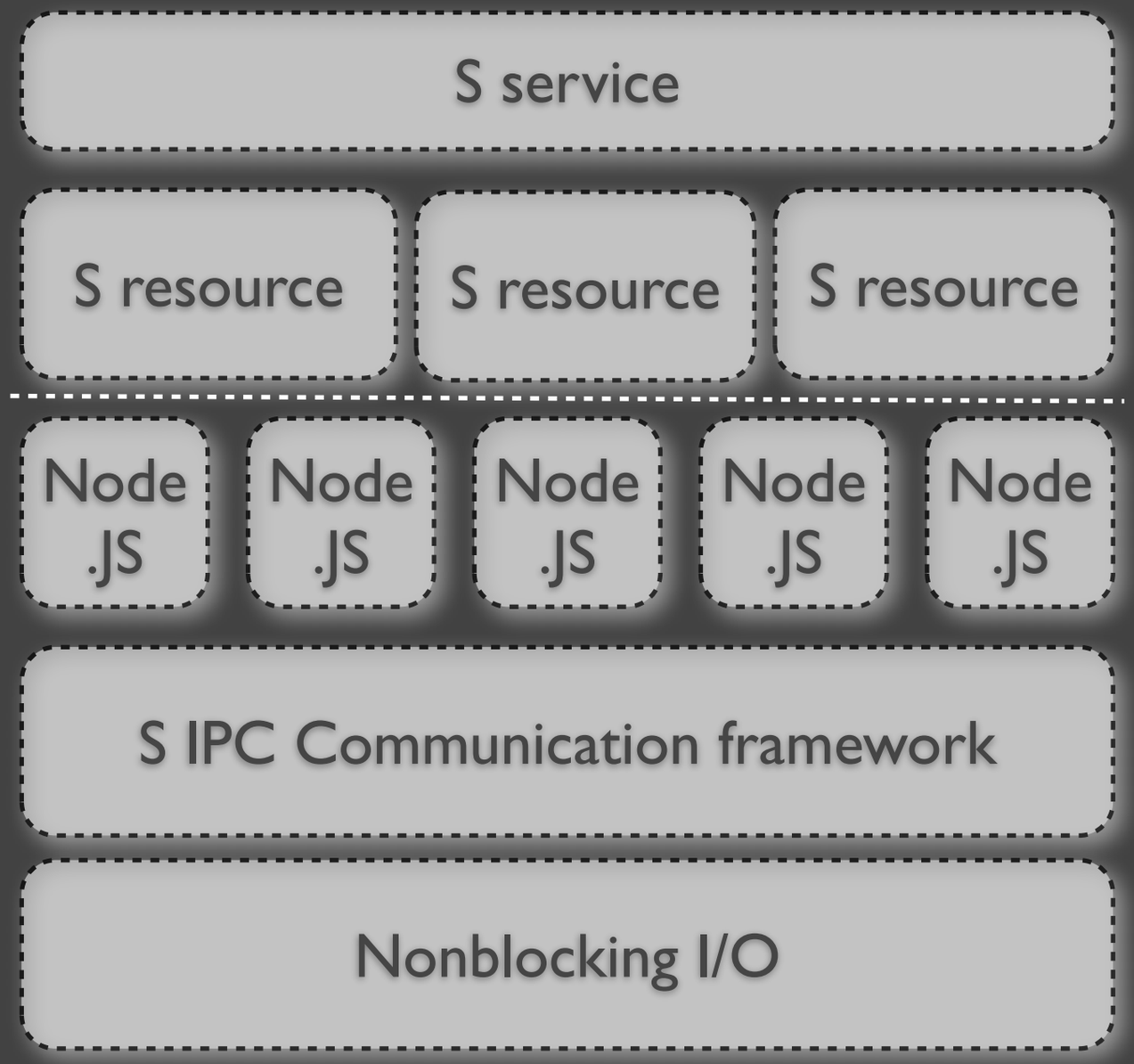
Università della Svizzera Italiana
Faculty of Informatics



S: Architecture

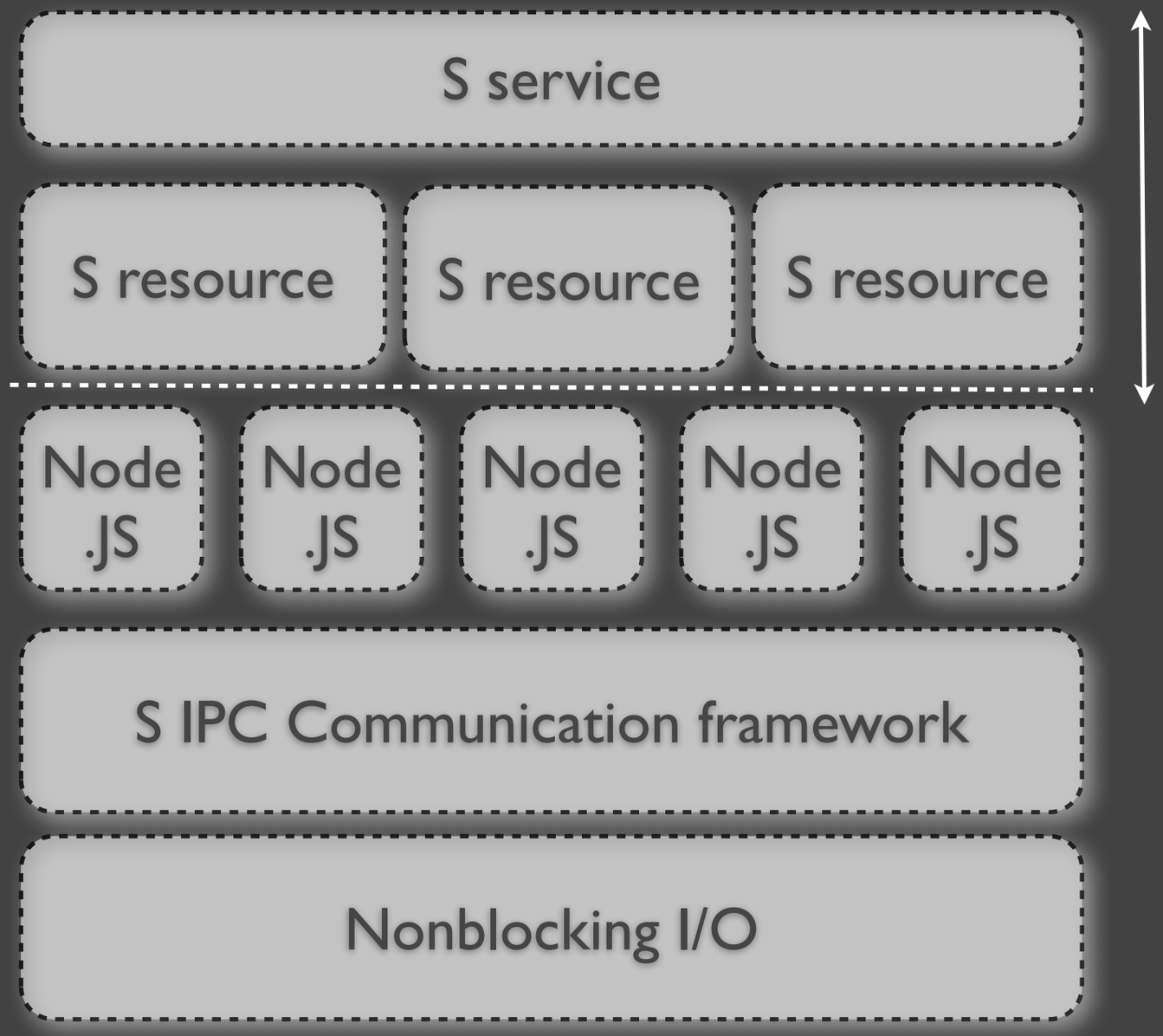


S: Architecture



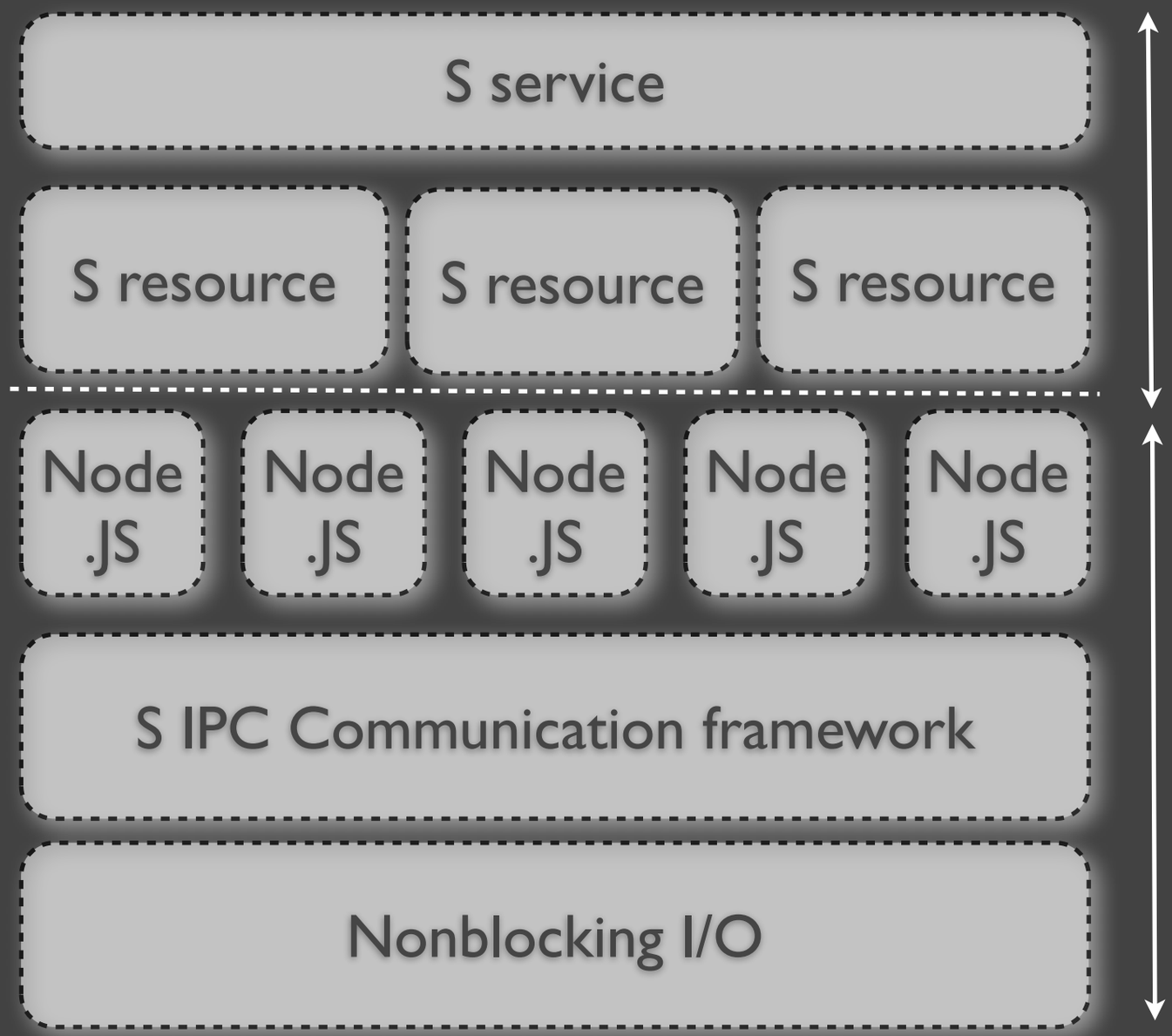
S: Architecture

Università della Svizzera Italiana
Faculty of Informatics



Synchronous blocking S code

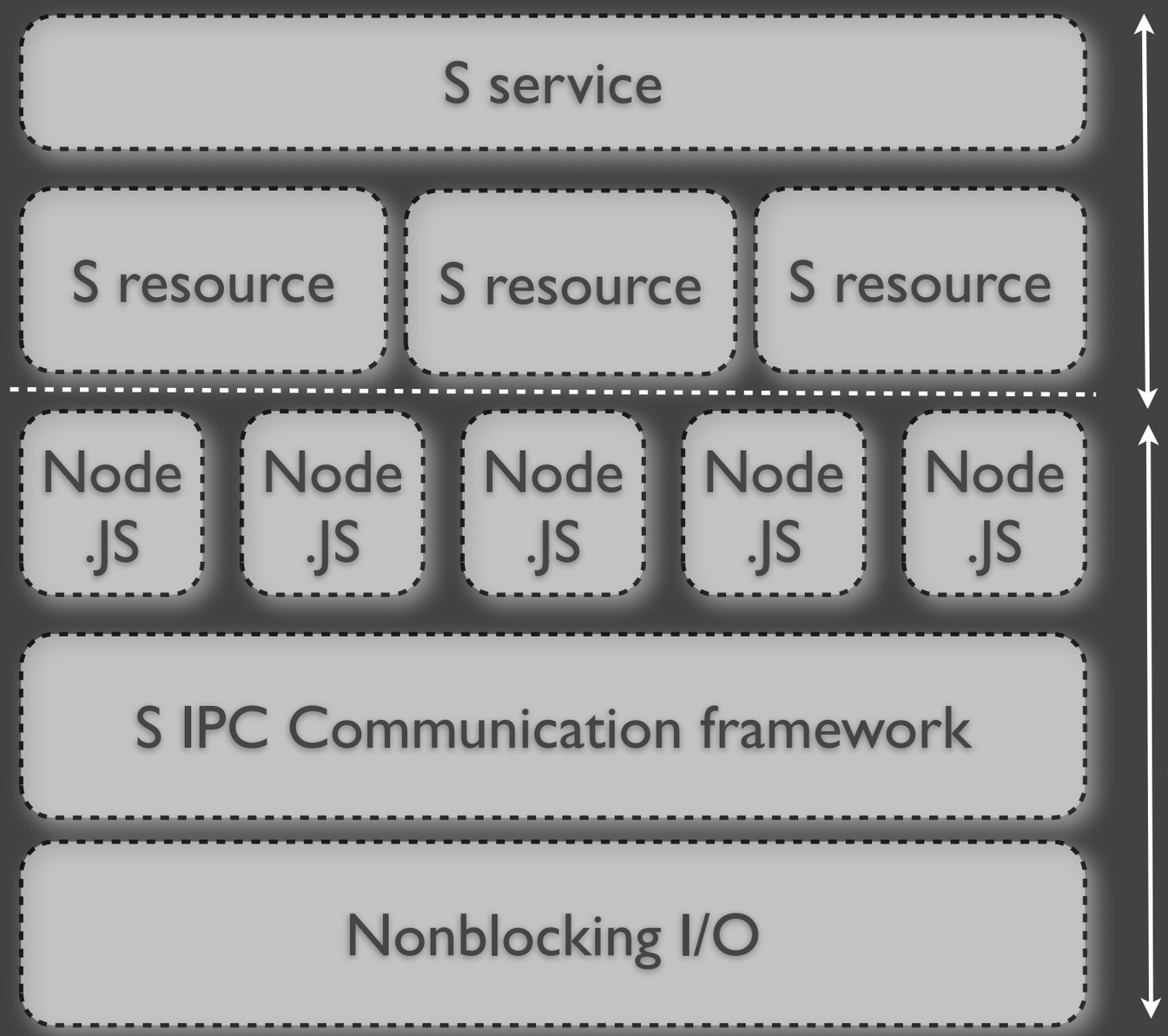
S: Architecture



Synchronous blocking S code

Asynchronous nonblocking Node.JS code

S: Architecture



No processes/threads locks/sync

Multiple Node.JS processes (IPC + shared state)

Hello S

```
res '/hello' on GET {
    respond 'World!'
}
```

Hello S

Request Handler

```
res '/hello' on GET {
    respond 'World!'
}
```

Hello S

URI

Request Handler

```
res '/hello' on GET {
    respond 'World!'
}
```

Hello S

URI

HTTP Verb

Request Handler

```
res '/hello' on GET {
  respond 'World!'
}
```

Hello S

URI

HTTP Verb

Request Handler

```
res '/hello' on GET {
    respond 'World!'
}
```

Any valid JavaScript statement

+

S specific constructs

Hello S

```
res ‘/hello’ on GET {
    respond ‘World!’
}
```

HTTP Client

S Service

Hello S

```
res ‘/hello’ on GET {
    respond ‘World!’
}
```


```
GET /hello HTTP/1.1
User-Agent: curl
Accept: */*
Host: your.host
```

Hello S

```
res '/hello' on GET {
    respond 'World!'
}
```


```
HTTP/1.1 200 OK
Content-Type: text/plain

World!
```

Compositions

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

```
res '/hello2' on GET {
    respond get '/hello1'
}
```

Compositions

Università della Svizzera Italiana
Faculty of Informatics

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

```
res '/hello2' on GET {
    respond get '/hello1'
}
```

Native HTTP support

Compositions

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

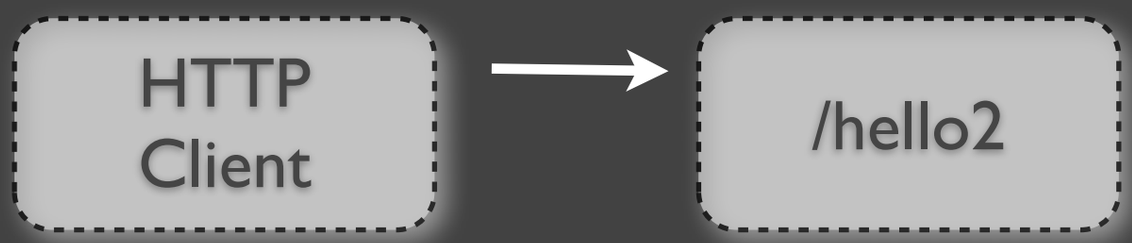
```
res '/hello2' on GET {
    respond get '/hello1'
}
```

HTTP Client

Compositions

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

```
res '/hello2' on GET {
    respond get '/hello1'
}
```



Compositions

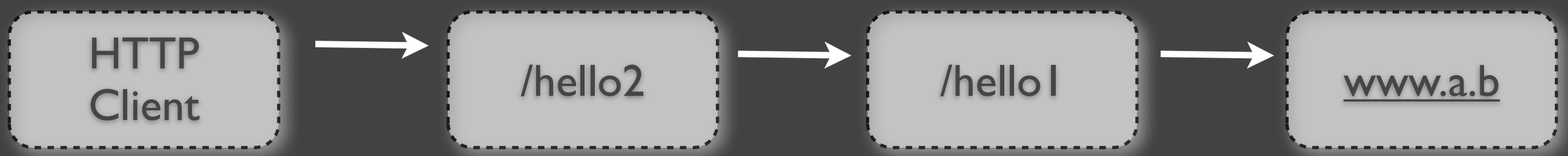
```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

```
res '/hello2' on GET {
    respond get '/hello1'
}
```

Compositions

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

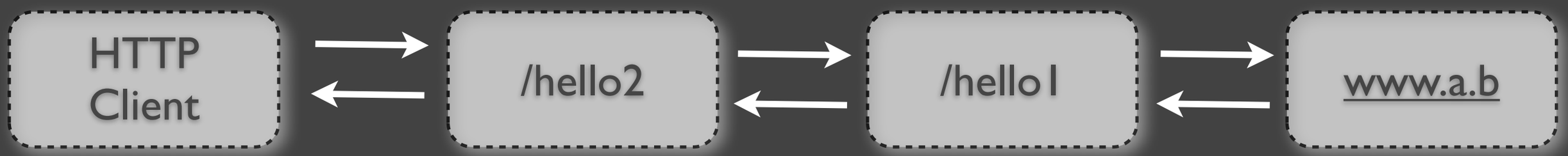
```
res '/hello2' on GET {
    respond get '/hello1'
}
```



Compositions

```
res '/hello1' on GET {
    respond get 'http://www.a.b'
}
```

```
res '/hello2' on GET {
    respond get '/hello1'
}
```



Concurrency & Parallelism

Università della Svizzera Italiana
Faculty of Informatics

```
res ‘/res1‘ on GET {

   // ...

   // CPU/bound blocking call
   var a = foo()
   respond a
}

res ‘/res2’ on GET {

   // ...
   res r = ‘http://www.google.ch/search=@’

   // I/O bound operation
   boo = r.get(‘key’)
   respond boo
}
```

Concurrency & Parallelism

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res1' on GET {
  // ...

  // CPU/bound blocking call
  var a = foo()
  respond a
}

res '/res2' on GET {
  // ...
  res r = 'http://www.google.ch/search=@'

  // I/O bound operation
  boo = r.get('key')
  respond boo
}
```

Atomically executed blocks

Concurrency & Parallelism

```
res '/res1' on GET {

    // ...

    // CPU/bound blocking call
    var a = foo()
    respond a
}
```

Parallel resources

```
res '/res2' on GET {

    // ...
    res r = 'http://www.google.ch/search=@'

    // I/O bound operation
    boo = r.get('key')
    respond boo
}
```

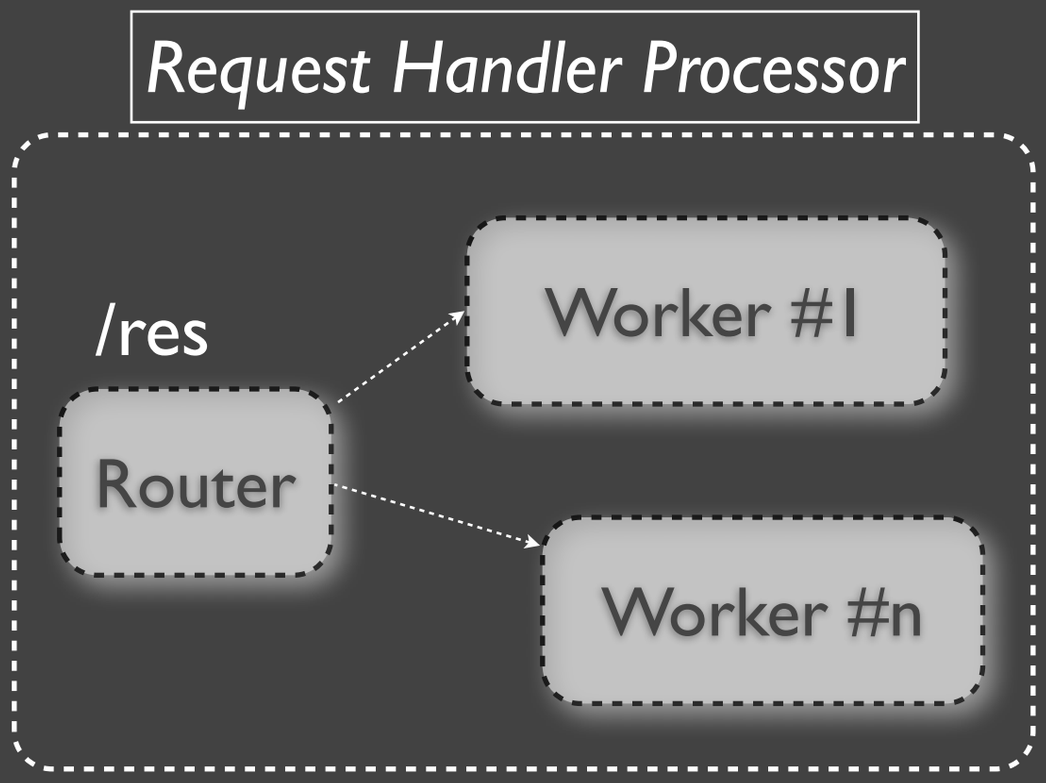
Concurrency & Parallelism

Università della Svizzera Italiana
Faculty of Informatics

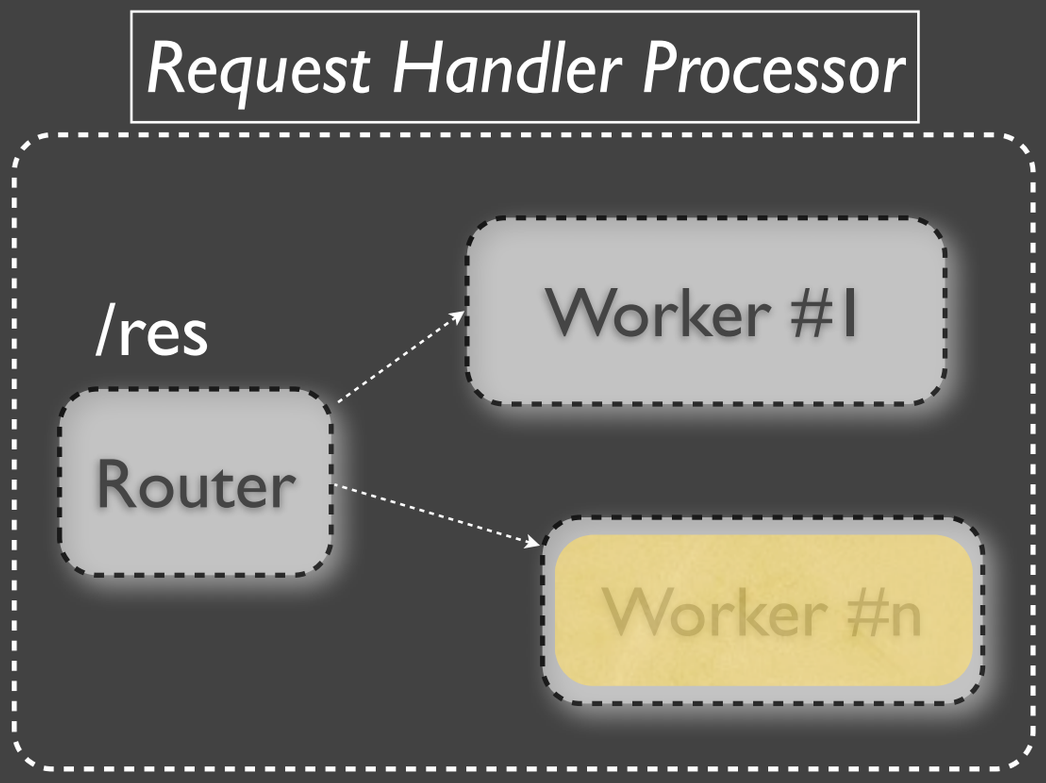
```
res ‘/res1‘ on GET {

   // ...

   // CPU/bound blocking call
   var a = foo()
   respond a
}

res ‘/res2’ on GET {

   // ...
   res r = ‘http://www.google.ch/search=@’

   // I/O bound operation
   boo = r.get(‘key’)
   respond boo
}
```

Synchronous operations

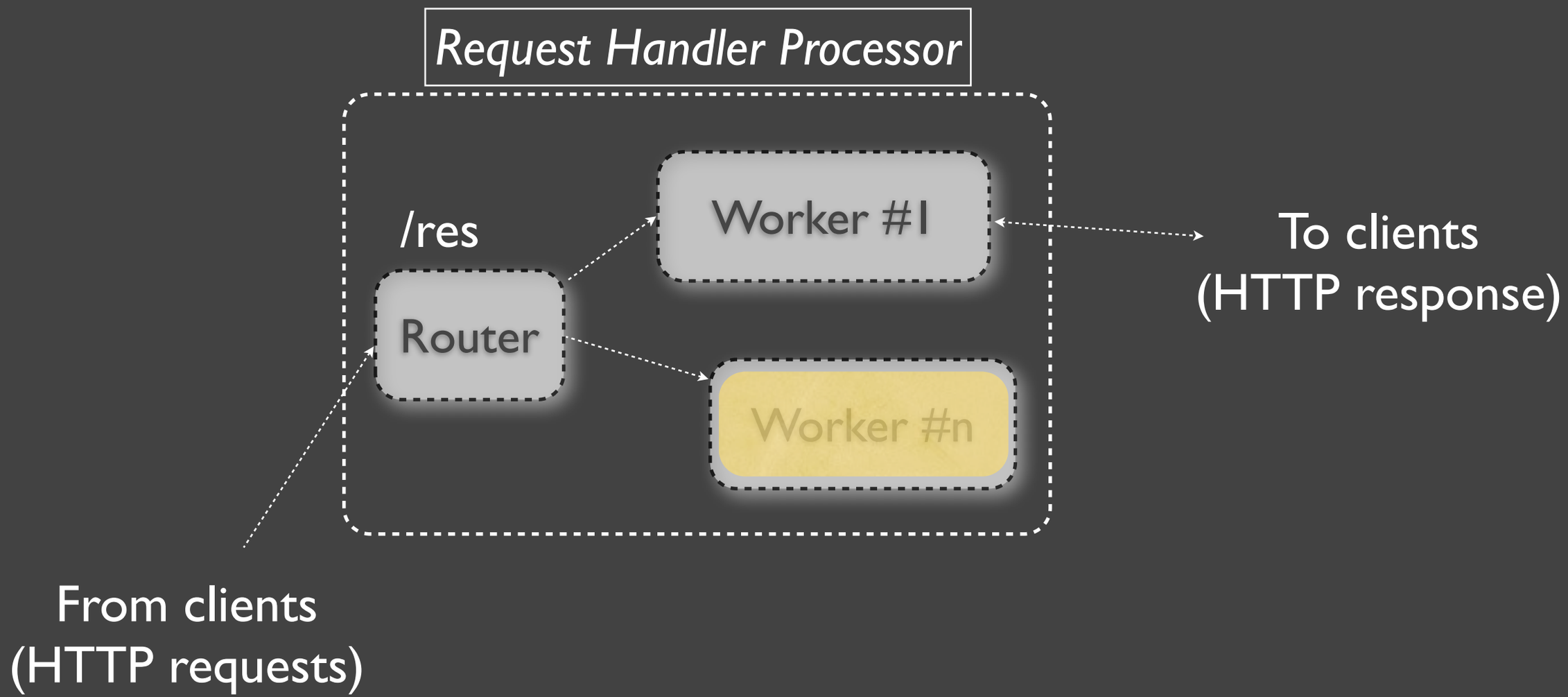
(1) CPU-bound Operations



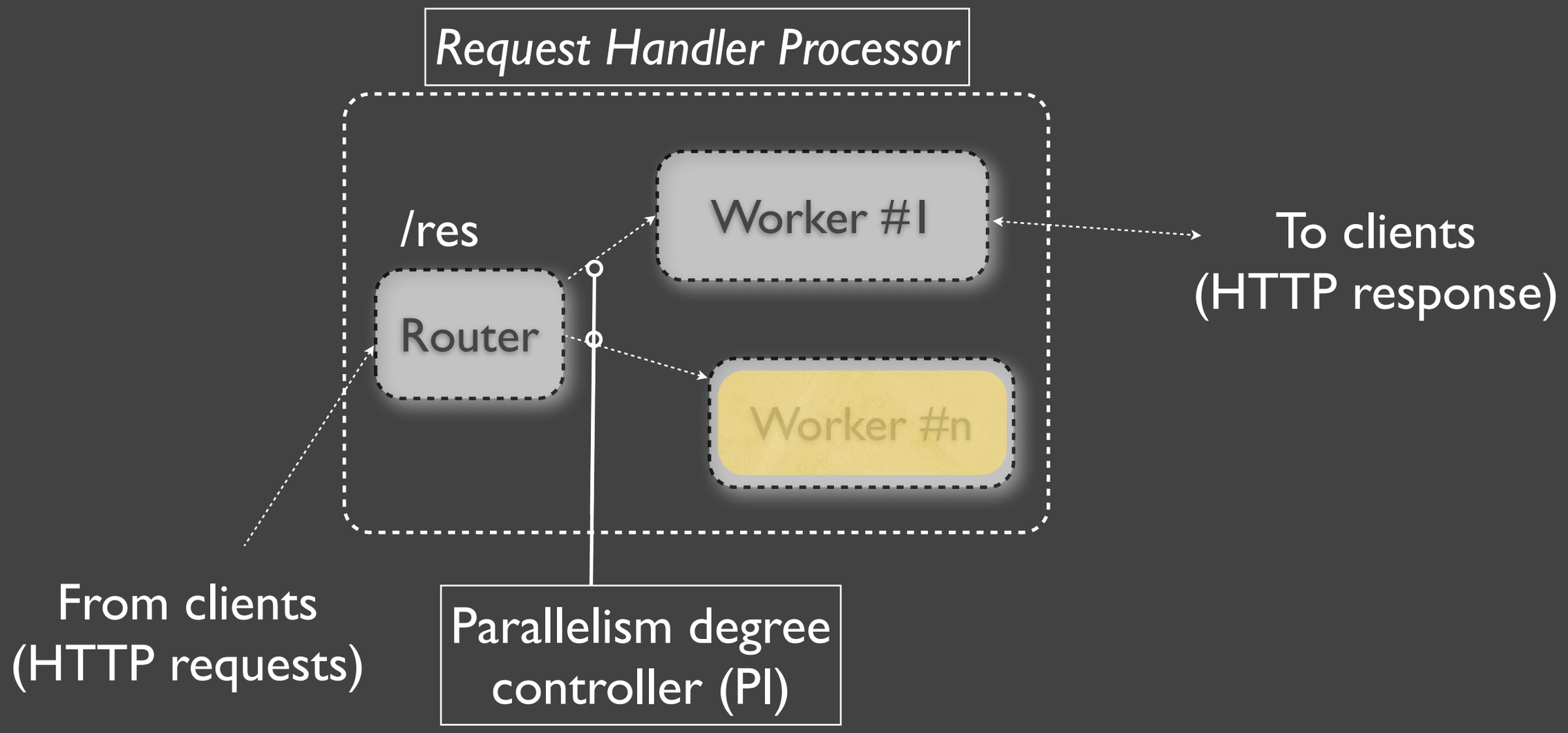
(1) CPU-bound Operations



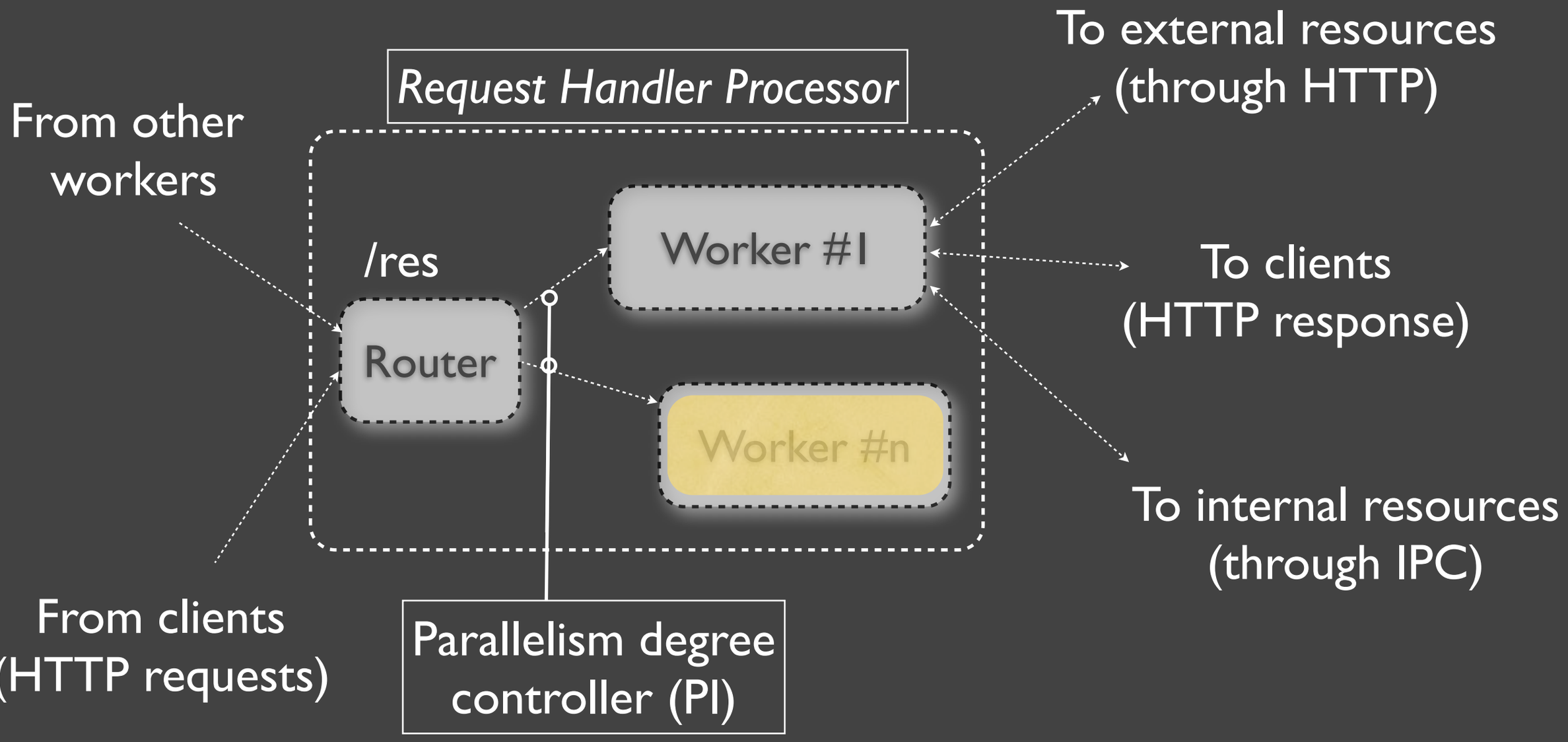
(1) CPU-bound Operations



(1) CPU-bound Operations
(2) I/O-bound Operations



(1) CPU-bound Operations
(2) I/O-bound Operations



(2) I/O-bound Operations

Università della Svizzera Italiana
Faculty of Informatics

Event-based compilation output targeting Node.JS

(2) I/O-bound Operations

Event-based compilation output targeting Node.JS

- Synchronous I/O operations are desynchronized by the compiler using multiple callbacks

(2) I/O-bound Operations

Event-based compilation output targeting Node.JS

- Synchronous I/O operations are desynchronized by the compiler using multiple callbacks

- Callbacks are scheduled by the runtime using an event-based approach.

Sync to Async Compilation

S vs. Node.JS

```
res '/example' on GET {
    var a = get 'www.google.com'
    var b = get 'www.bing.com'
    var c = get 'www.yahoo.com'
    respond a+b+c
}
```

Sync to Async Compilation

Università della Svizzera Italiana
Faculty of Informatics

S vs. Node.JS

```
res '/example' on GET {
    var a = get 'www.google.com'
    var b = get 'www.bing.com'
    var c = get 'www.yahoo.com'
    respond a+b+c
}
```

```
http.createServer(function(creq,cres){
    if(creq.method=='GET'&&creq.url=='/example') {
        var a,b,c = ''
        startGet('www.google.com',
            function(req,res) {
                a = res.body
                startGet('www.bing.com',
                    function(req,res) {
                        b = res.body
                        startGet('www.yahoo.com',
                            function(req,res) {
                                c = res.body
                                cres.end(a+b+c)
                            })
                    })
            })
    }
})
```

Sync to Async Compilation

S vs. Node.JS

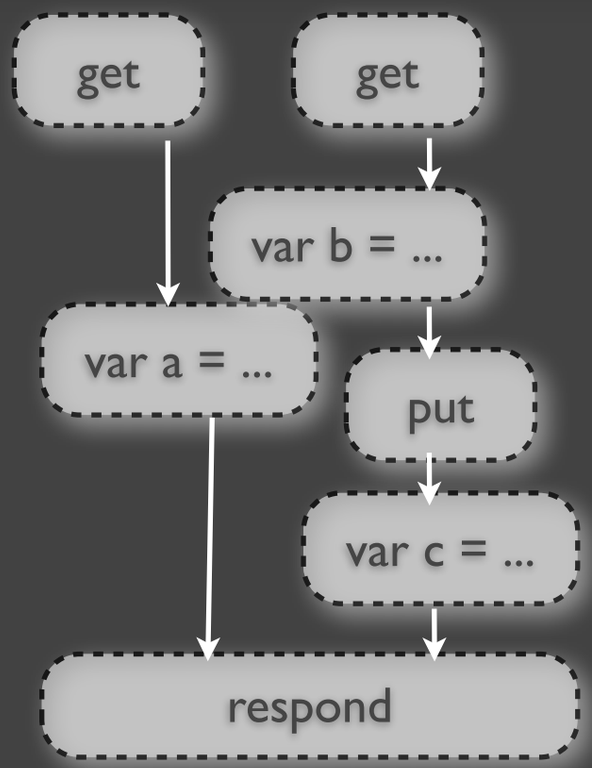
```
res ‘/example’ on GET {
  par {
    var a = get ‘www.google.com’
    var b = get ‘www.bing.com’
    var c = put ‘www.a.b/?d=’+b
    respond a+c
}}
```

Sync to Async Compilation

Università della Svizzera Italiana
Faculty of Informatics

S vs. Node.JS

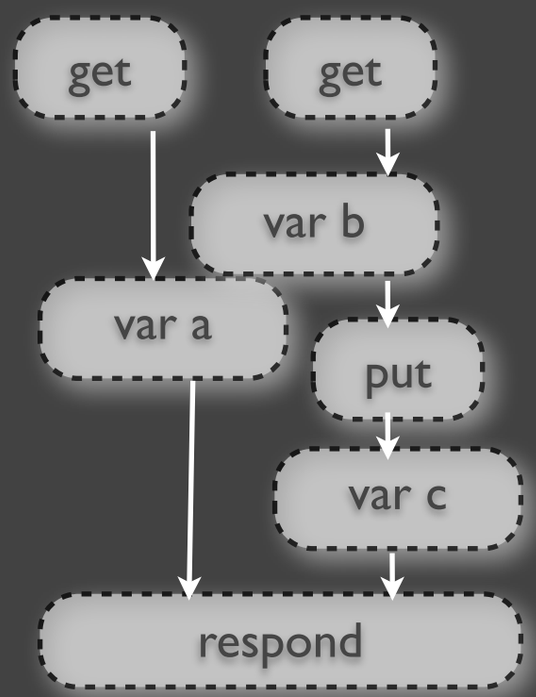
```
res '/example' on GET {
  par {
    var a = get 'www.google.com'
    var b = get 'www.bing.com'
    var c = put 'www.a.b/?d='+b
    respond a+c
}}
```




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

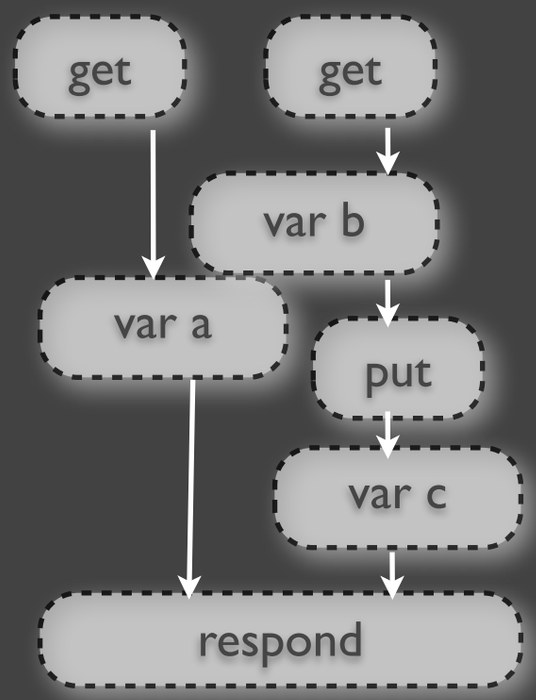
Università della Svizzera Italiana
Faculty of Informatics




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

Callbacks registration

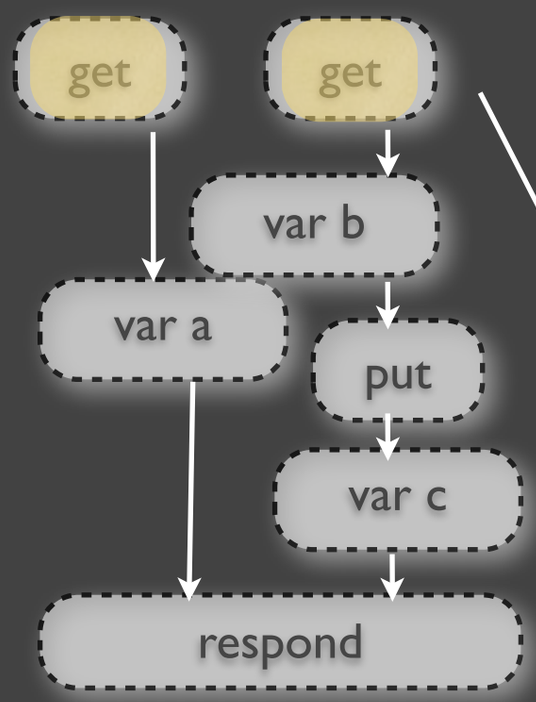



```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

Università della Svizzera Italiana
Faculty of Informatics

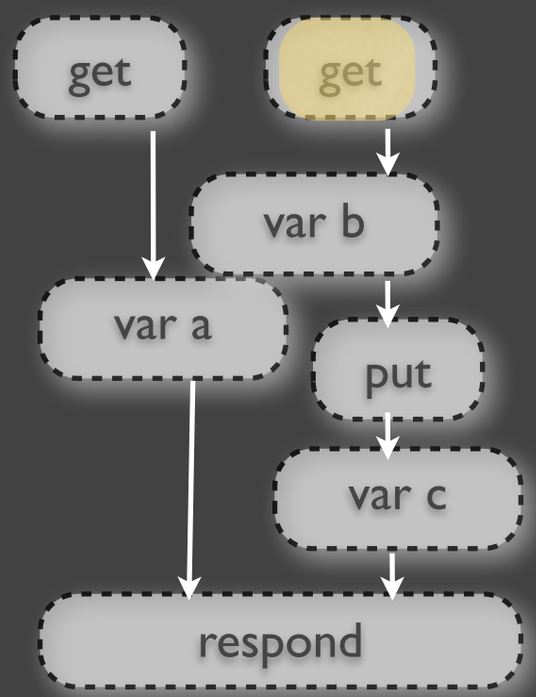
Callbacks registration




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

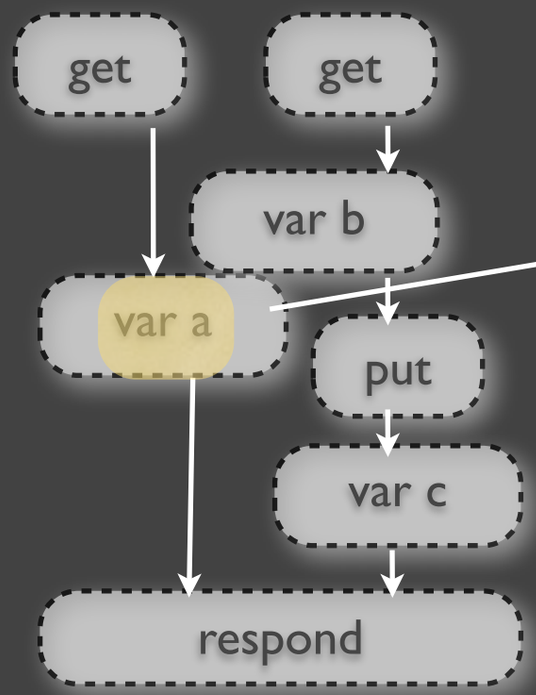
Università della Svizzera Italiana
Faculty of Informatics




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

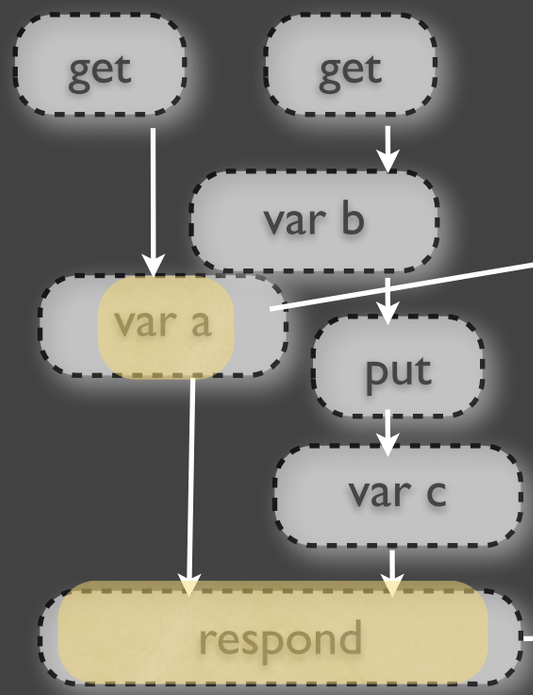
Università della Svizzera Italiana
Faculty of Informatics




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Sync to Async Compilation

Università della Svizzera Italiana
Faculty of Informatics




```
http.createServer(function(req,res) {
  if(creq.method=='GET'&&creq.url=='/example') {
    var G = {}
    on('done1', function(result){
        G.a = result; emit('done3')})
    on('done2', function(result){
        G.b = result;
        startPut('www.a.b/d?='+G.b, 'done4')
    })
    on('done4', function(result) {
        G.c = result;
        emit('done5')
    })
    onAll(['done3','done5'], function() {
        res.end(G.a+G.c)
    })
    startGet('www.google.com', 'done1')
    startGet('www.bing.com', 'done2')
  }
})
```

Other Constructs

Università della Svizzera Italiana
Faculty of Informatics

pfor

```
res '/pfor' on GET {
    var L = [1,2,3]
    var a = ''
    res r = 'www.a.b/val=@'
    pfor (var i in L) {
        a += r.get(L[i])
    }
    respond a
  }
}
```

res creation

```
res '/example' {
    state s = 0
    on POST {
      s++
      res '/newResource'+s {
        on GET { respond s }
        on DELETE {}
      }
    }
}
```

Stateful Services

Università della Svizzera Italiana
Faculty of Informatics

```
res '/helloState' {
  state s = 'world'
  on GET {
    respond 'hello' + s
  }
  on PUT {
    s = req.name
  }
}
```

Stateful Services

Università della Svizzera Italiana
Faculty of Informatics

```
res '/helloState' {
  state s = 'world'
  on GET {
    respond 'hello' + s
  }
  on PUT {
    s = req.name
  }
}
```

Shared state

Stateful Services

Università della Svizzera Italiana
Faculty of Informatics

```
res '/helloState' {
  state s = 'world'
  on GET {
    respond 'hello' + s
  }
  on PUT {
    s = req.name
  }
}
```

Shared state

Read-only operations

Stateful Services


```
res '/helloState' {
  state s = 'world'
  on GET {
    respond 'hello' + s
  }
  on PUT {
    s = req.name
  }
}
```

Shared state

Read-only operations

Write operation

Uniform Interface and Parallelism

HTTP's Uniform Interface

HTTP Verb	Properties	Parallelism
HTTP GET	Idempotent, safe, stateless	Yes
HTTP PUT	Idempotent, Side-effects	If stateless
HTTP DELETE	Idempotent, Side-effects	If stateless
HTTP POST	Side-effects	If stateless

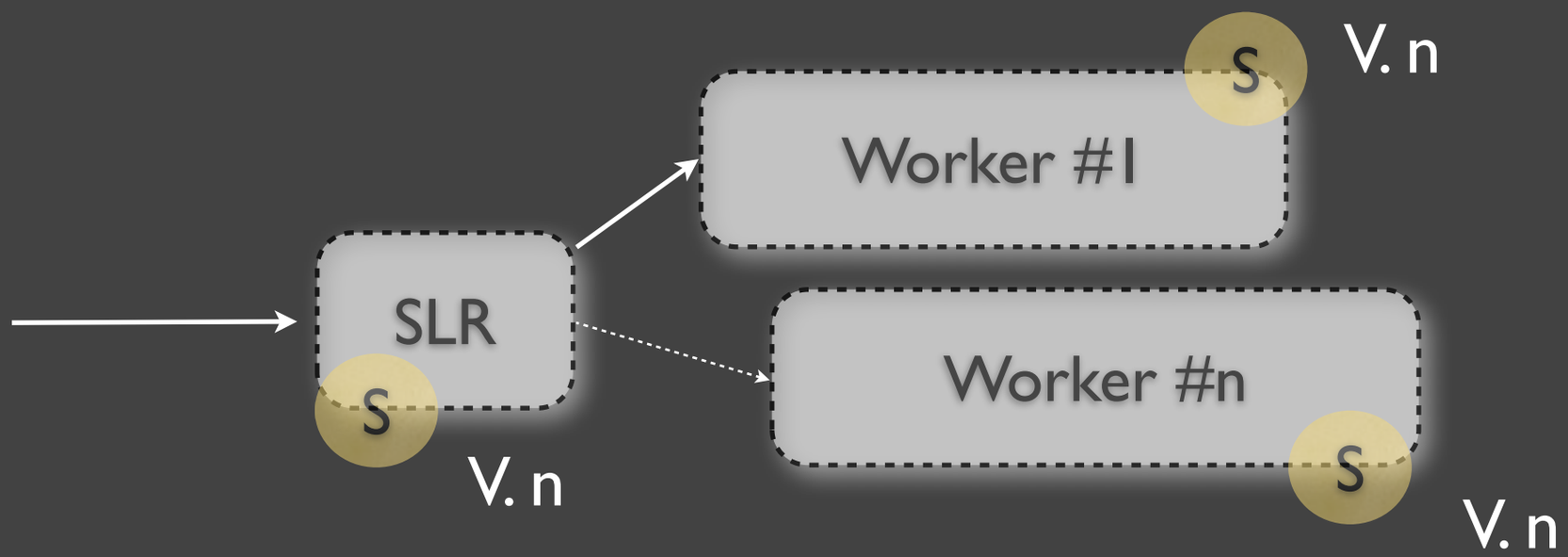
Versioned State Management

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res'
    state s
    on GET {
        // read s
    }
    on PUT {
        // alter s
    }
}
```

Guarantees:

- Eventual (C)onsistency
- (A)vailability
- (P)artition Tolerance



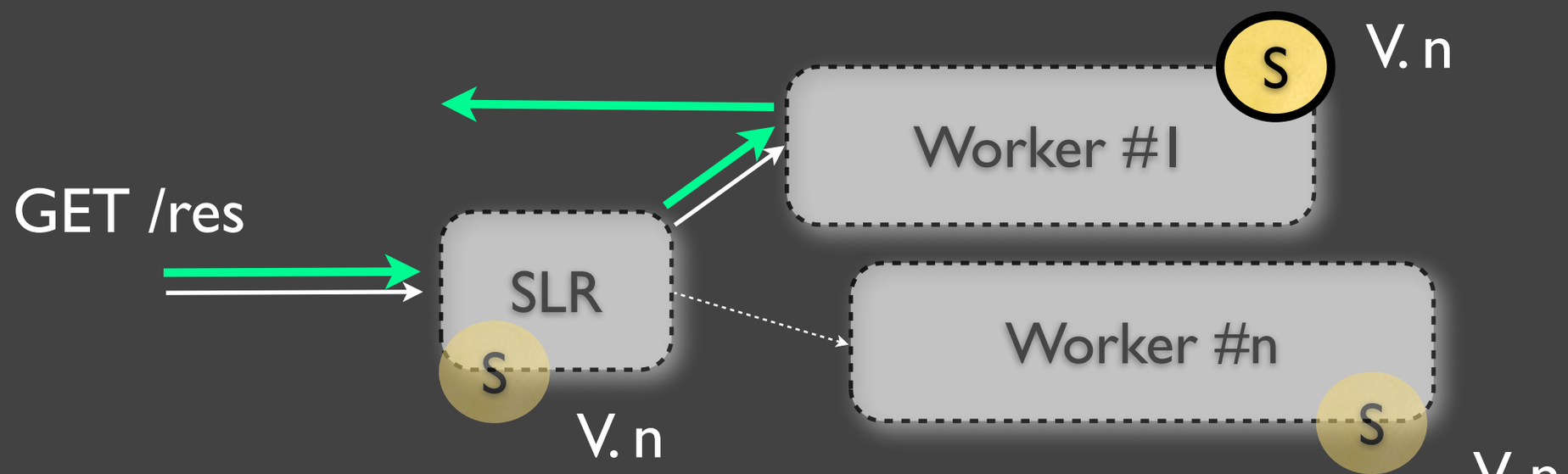
Versioned State Management

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res'
    state s
    on GET {
        // read s
    }
    on PUT {
        // alter s
    }
}
```

Guarantees:

- Eventual (C)onsistency
- (A)vailability
- (P)artition Tolerance



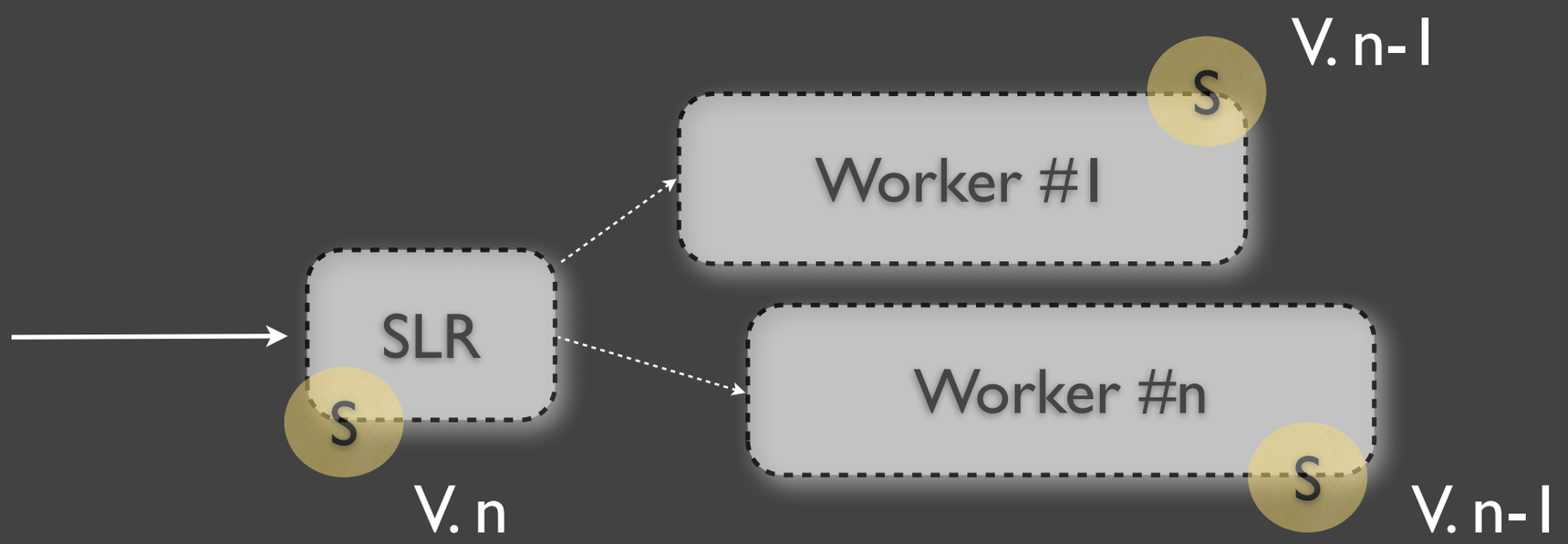
Versioned State Management

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res'
    state s
    on GET {
        // read s
    }
    on PUT {
        // alter s
    }
}
```

Guarantees:

- Eventual (C)onsistency
- (A)vailability
- (P)artition Tolerance



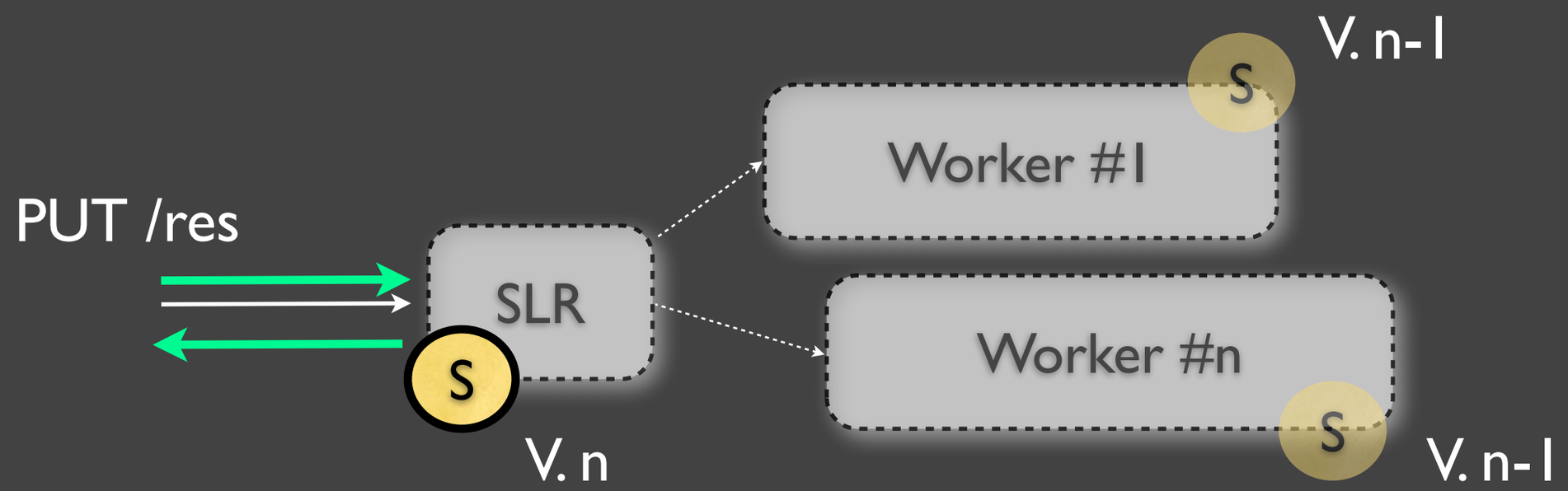
Versioned State Management

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res'
    state s
    on GET {
        // read s
    }
    on PUT {
        // alter s
    }
}
```

Guarantees:

- Eventual (C)onsistency
- (A)vailability
- (P)artition Tolerance



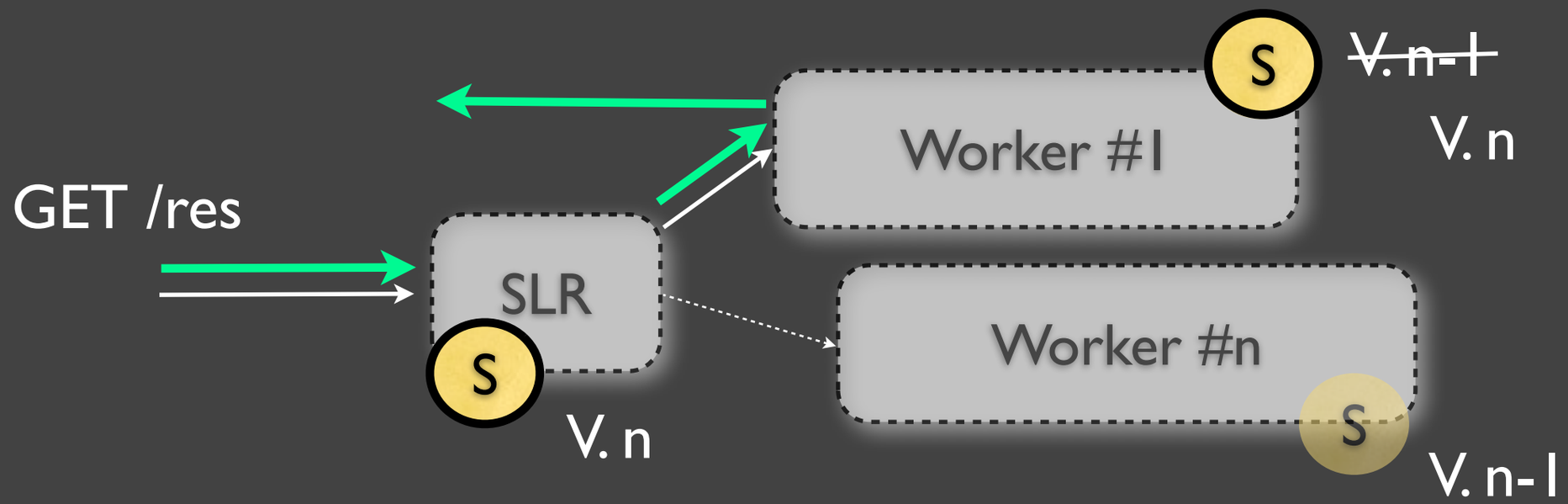
Versioned State Management

Università della Svizzera Italiana
Faculty of Informatics

```
res '/res'
    state s
    on GET {
        // read s
    }
    on PUT {
        // alter s
    }
}
```

Guarantees:

- Eventual (C)onsistency
- (A)vailability
- (P)artition Tolerance



Example: // Crawler


```
service crawl {
    res '/crawl' on PUT {
        var filter = require('HTMLparser.js').filter
        res store = '/store?value=@'
        res crawl = '/crawl?list=@'

        pfor(var i in req.list) {
            var list = filter(get req.list[i])
            par {
                store.put(list)
                crawl.put(list)
            }
        }
    }

    res '/store' {
        state s = ''
        on PUT {
            s += req.value
        }
        on GET {
            respond s
        }
    }
}
```

Example: // Crawler

Università della Svizzera Italiana
Faculty of Informatics

```
service crawl {
    res '/crawl' on PUT {
        var filter = require('HTMLparser.js').filter
        res store = '/store?value=@'
        res crawl = '/crawl?list=@'

        pfor(var i in req.list) {
            var list = filter(get req.list[i])
            par {
                store.put(list)
                crawl.put(list)
            }
        }
    }

    res '/store' {
        state s = ''
        on PUT {
            s += req.value
        }
        on GET {
            respond s
        }
    }
}
```

JavaScript import

Example: // Crawler


```
service crawl {
    res '/crawl' on PUT {
        var filter = require('HTMLparser.js').filter
        res store = '/store?value=@'
        res crawl = '/crawl?list=@'

        pfor(var i in req.list) {
            var list = filter(get req.list[i])
            par {
                store.put(list)
                crawl.put(list)
            }
        }
    }

    res '/store' {
        state s = ''
        on PUT {
            s += req.value
        }
        on GET {
            respond s
        }
    }
}
```

JavaScript import

External resources declaration

Example: // Crawler

```
service crawl {
    res '/crawl' on PUT {
        var filter = require('HTMLparser.js').filter
        res store = '/store?value=@'
        res crawl = '/crawl?list=@'

        pfor(var i in req.list) {
            var list = filter(get req.list[i])
            par {
                store.put(list)
                crawl.put(list)
            }
        }
    }

    res '/store' {
        state s = ''
        on PUT {
            s += req.value
        }
        on GET {
            respond s
        }
    }
}
```

JavaScript import

External resources declaration

Parallel recursive crawling

Example: // Crawler


```
service crawl {
    res '/crawl' on PUT {
        var filter = require('HTMLparser.js').filter
        res store = '/store?value=@'
        res crawl = '/crawl?list=@'

        pfor(var i in req.list) {
            var list = filter(get req.list[i])
            par {
                store.put(list)
                crawl.put(list)
            }
        }
    }

    res '/store' {
        state s = ''
        on PUT {
            s += req.value
        }
        on GET {
            respond s
        }
    }
}
```

JavaScript import

External resources declaration

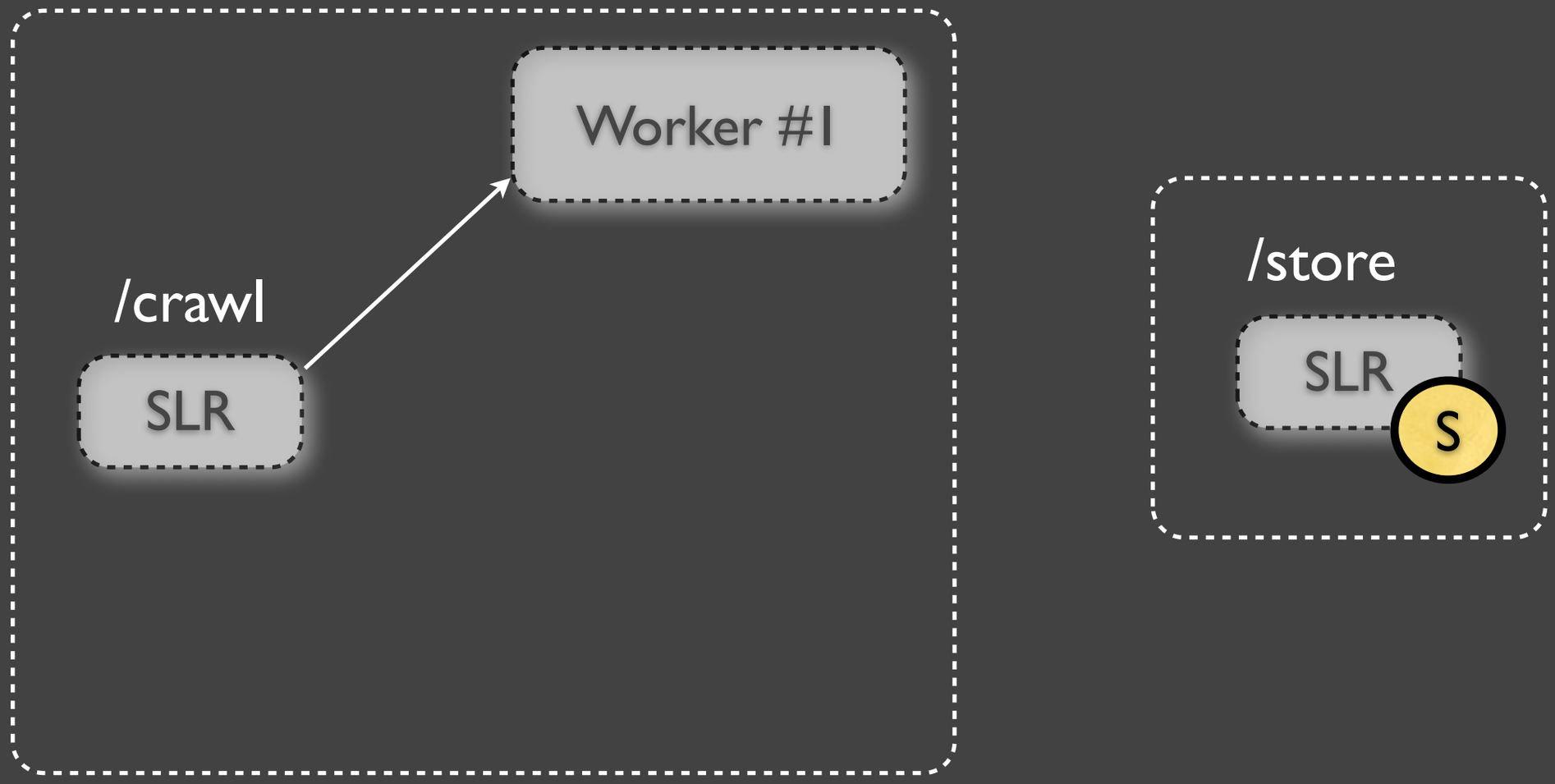
Parallel recursive crawling

Result accumulation

Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

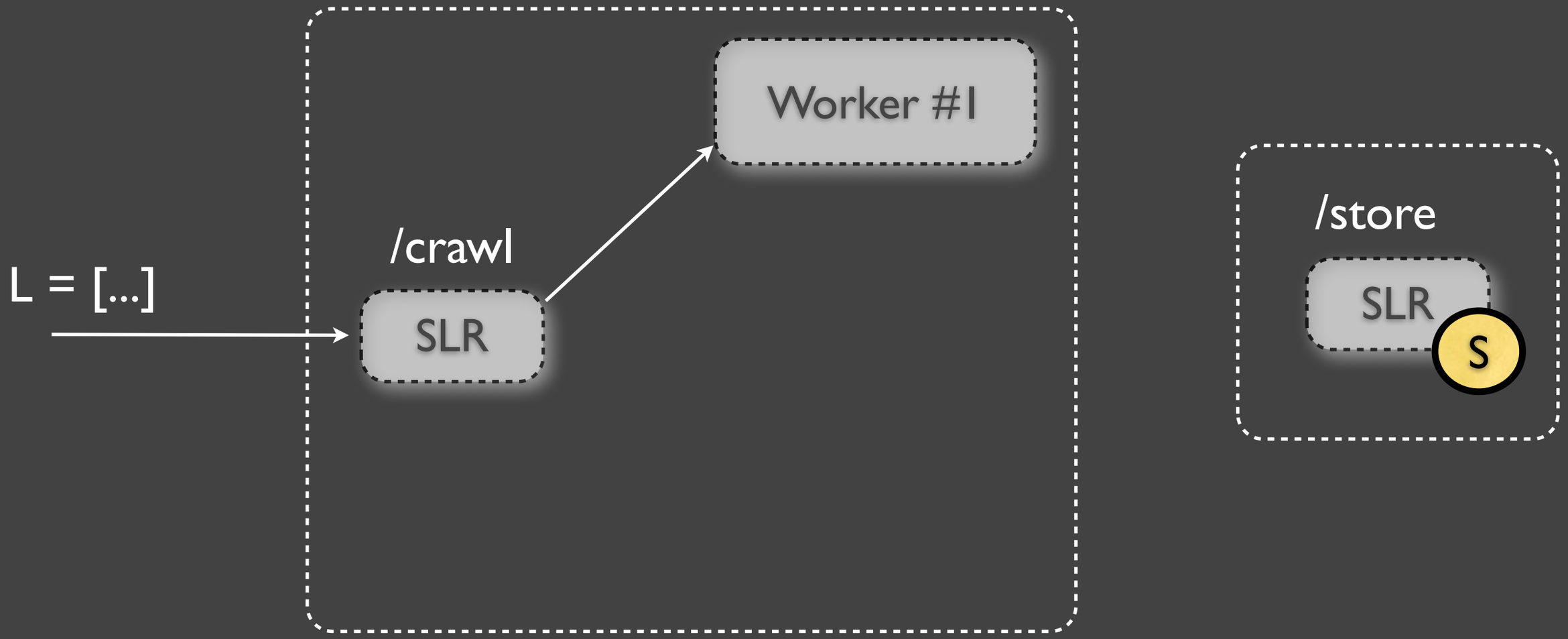
```
res '/store' on PUT {
    // ...
}
```



Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

```
res '/store' on PUT {
    // ...
}
```

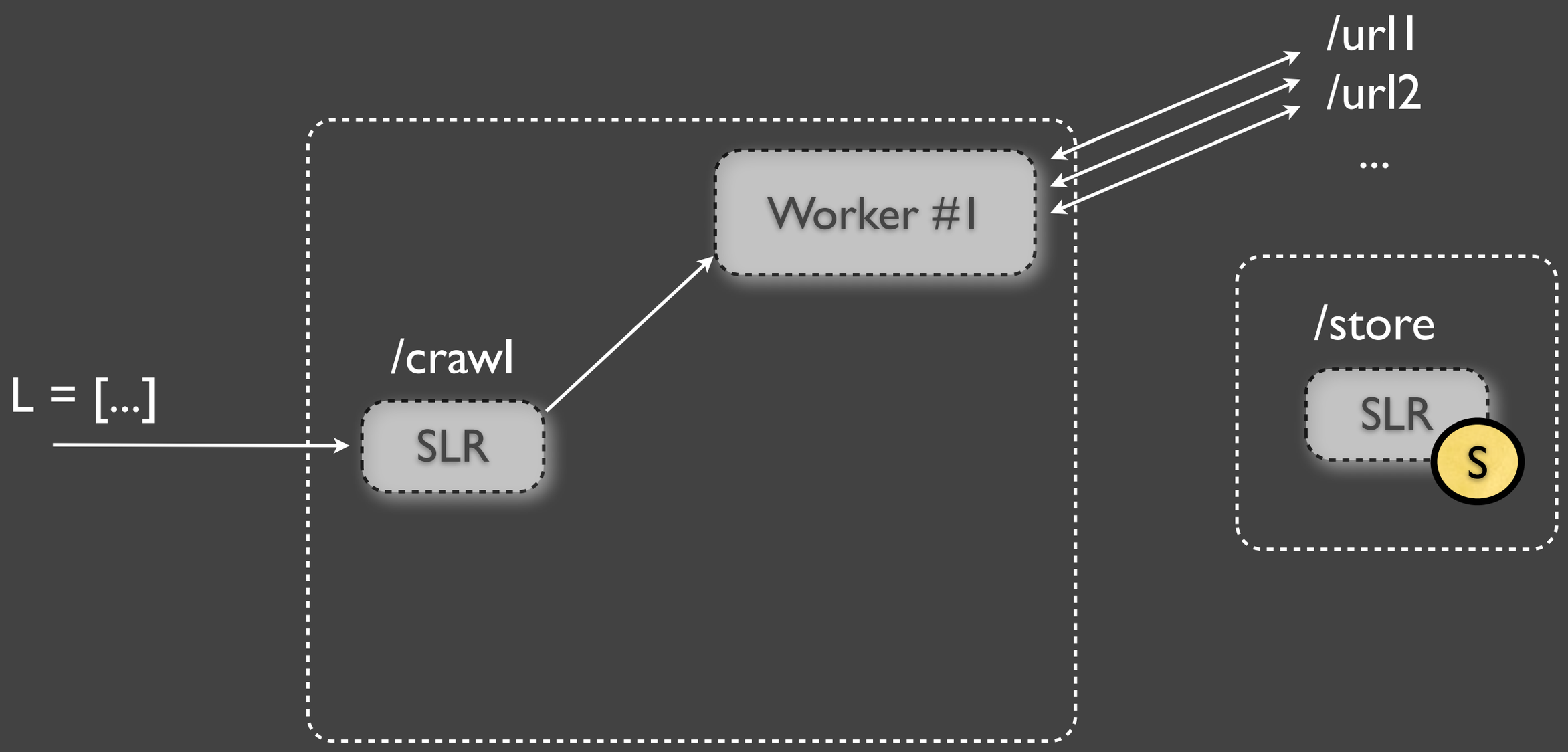


Crawler Runtime

Università della Svizzera Italiana
Faculty of Informatics

```
res '/crawl' on PUT {
    // ...
}
```

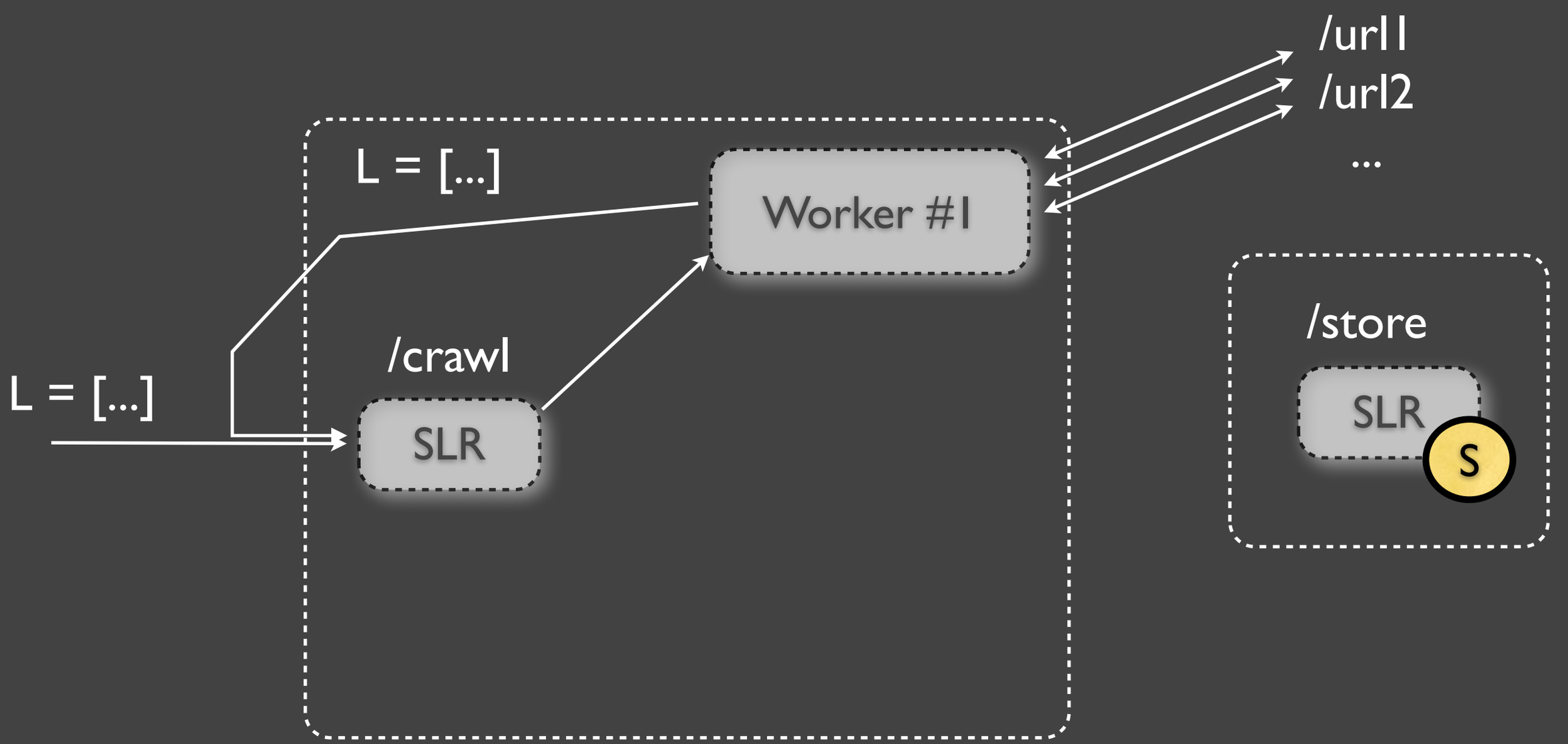
```
res '/store' on PUT {
    // ...
}
```



Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

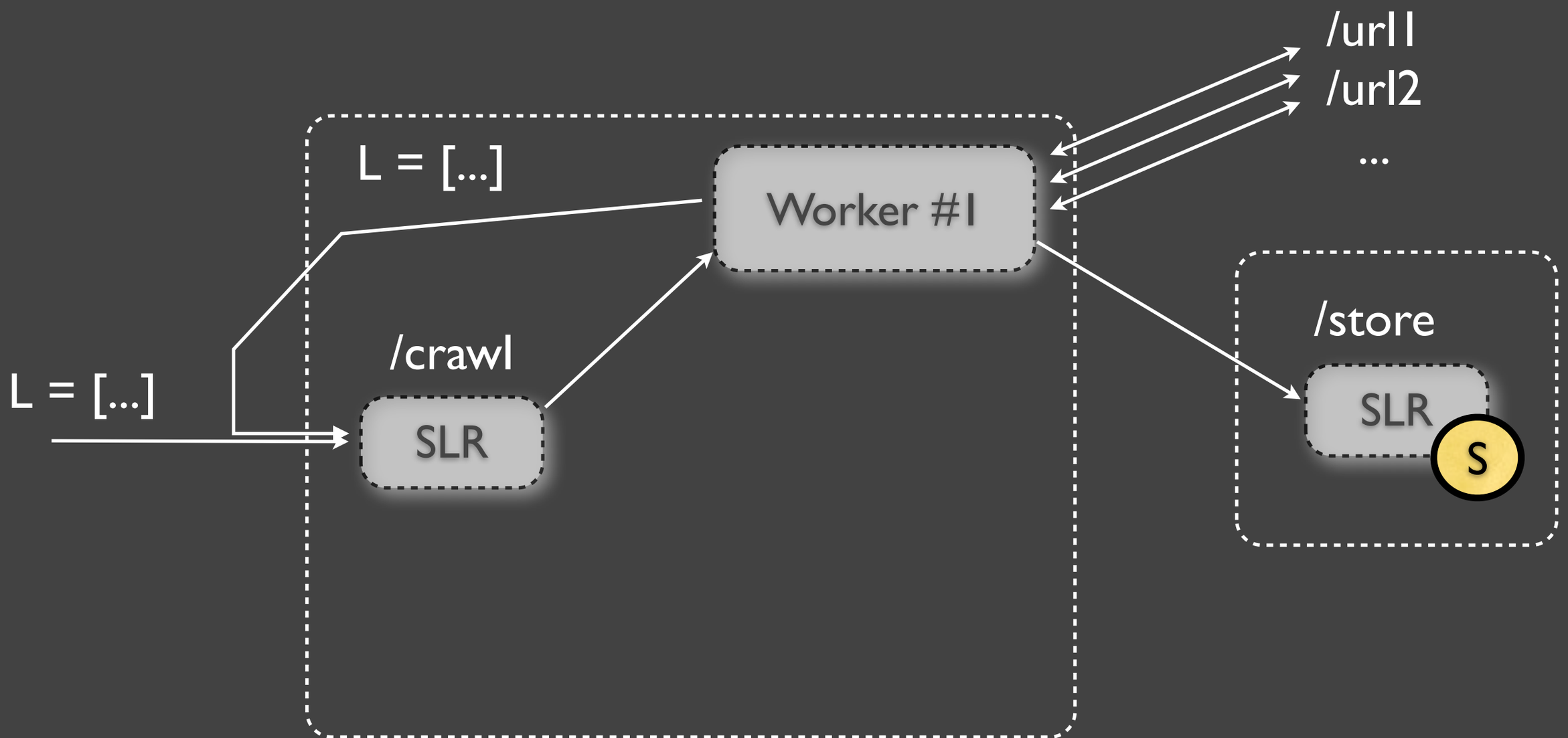
```
res '/store' on PUT {
    // ...
}
```



Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

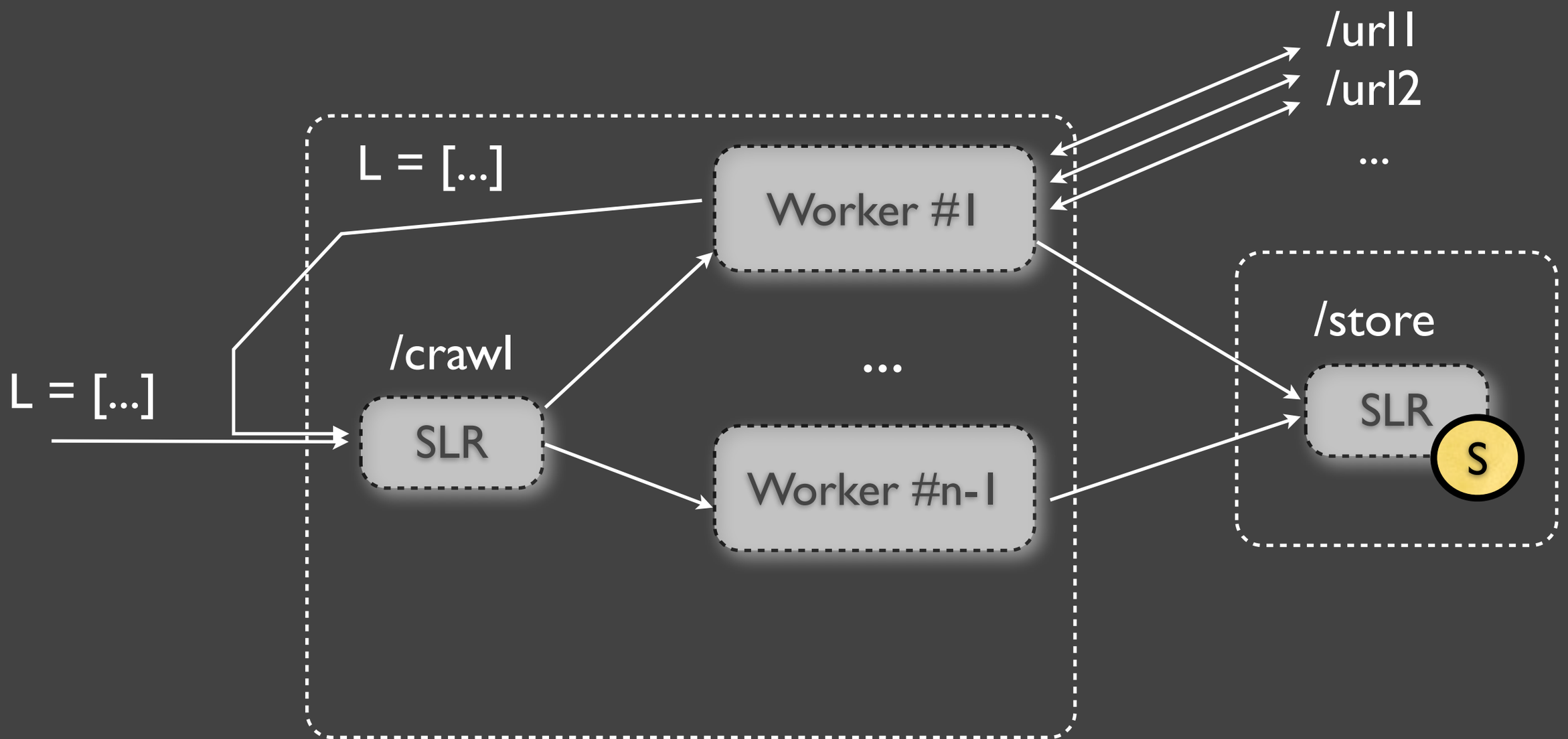
```
res '/store' on PUT {
    // ...
}
```



Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

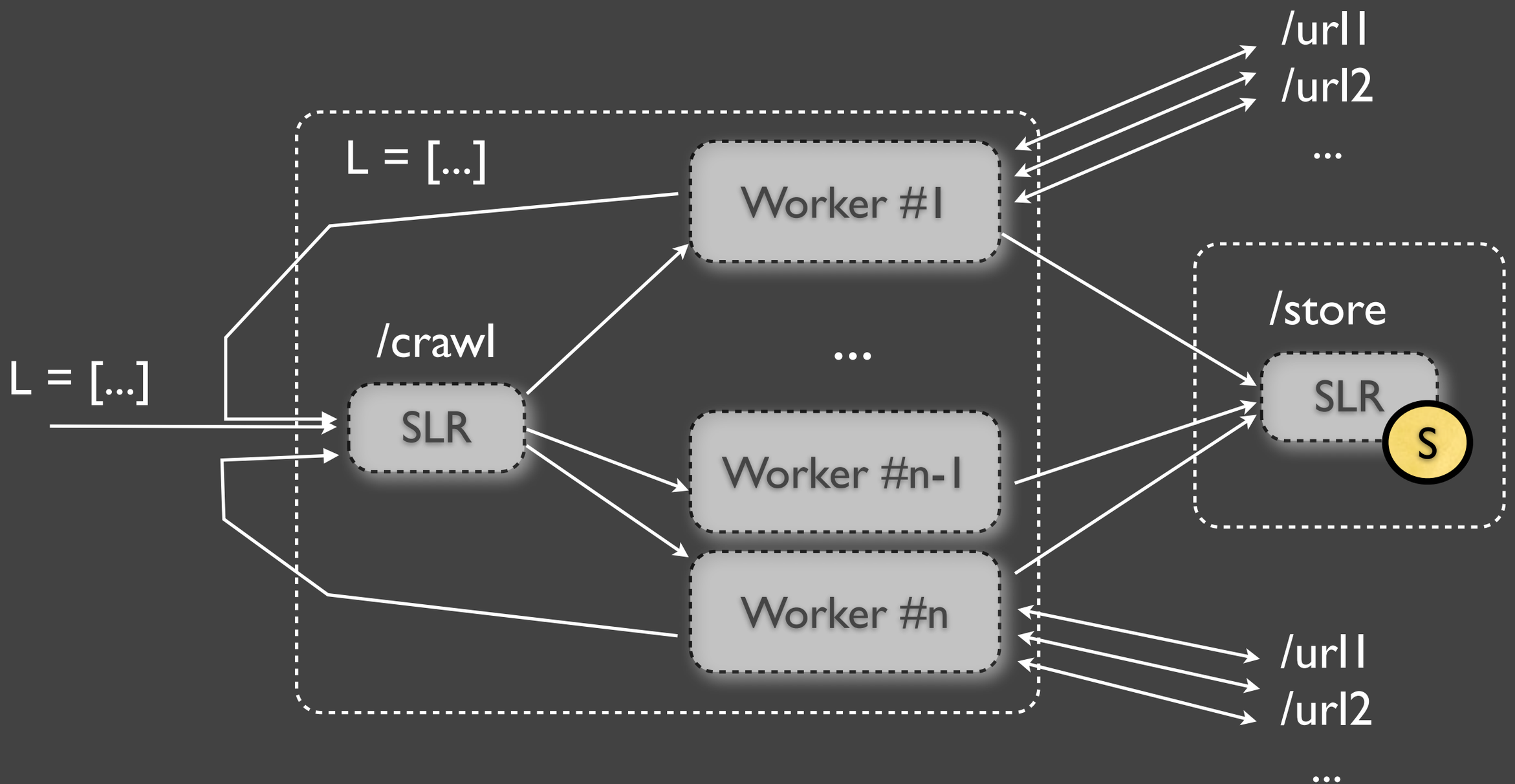
```
res '/store' on PUT {
    // ...
}
```



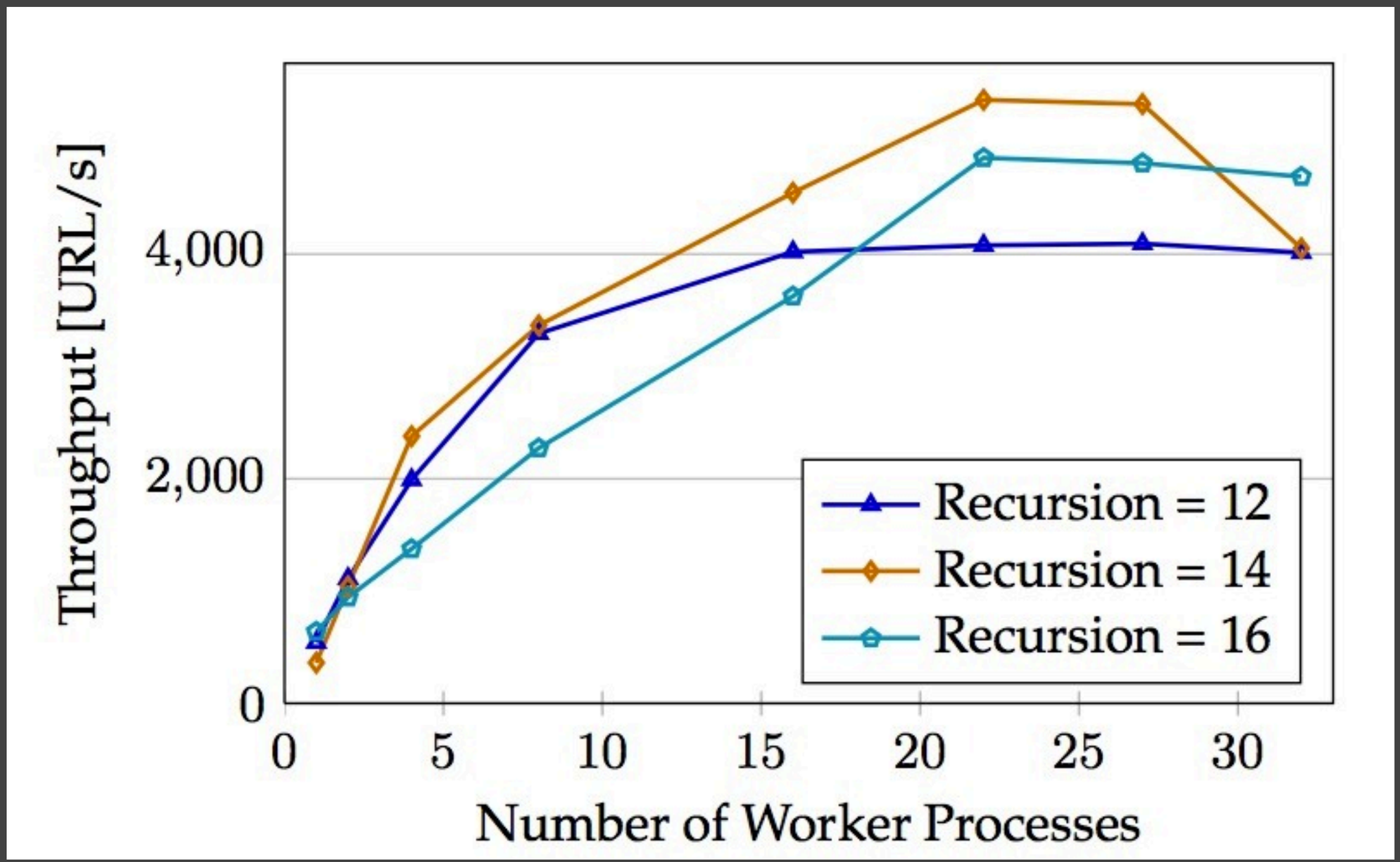
Crawler Runtime

```
res '/crawl' on PUT {
    // ...
}
```

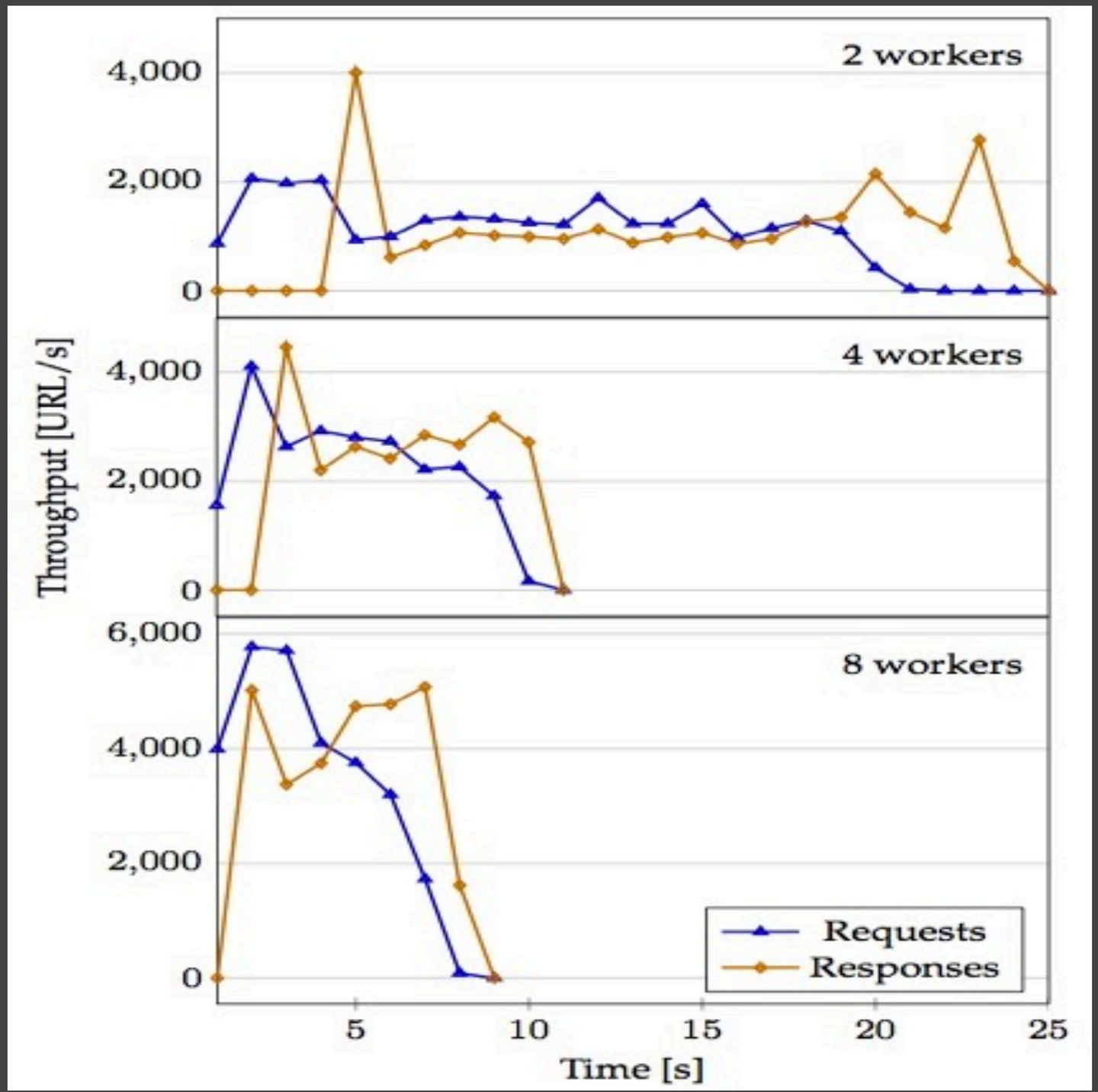
```
res '/store' on PUT {
    // ...
}
```



Crawler Performance (24cores)



Crawler Performance



Conclusion

What is S ?

Conclusion

What is S ?

Simple language for composing RESTful services
Scalable parallel runtime system
JavaScript compatible

Conclusion

What is S ?

Simple language for composing RESTful services
Scalable parallel runtime system
JavaScript compatible

Safe parallelism is possible for scripting stateful Web services if they are RESTful

Conclusion

Future work

JVM (Rhino Engine)
REST + Streaming services
Distributed deployment (Cloud)

Thank You

http://sosoa.inf.usi.ch/S

daniele.bonetta@usi.ch

Daniele Bonetta
Achille Peternier, Cesare Pautasso, Walter Binder
Faculty of Informatics
University of Lugano - USI
Switzerland