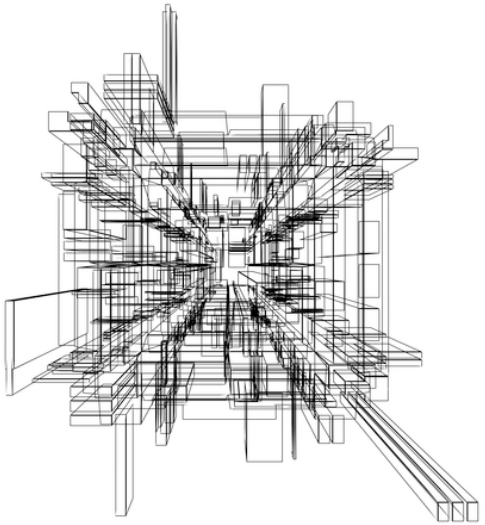
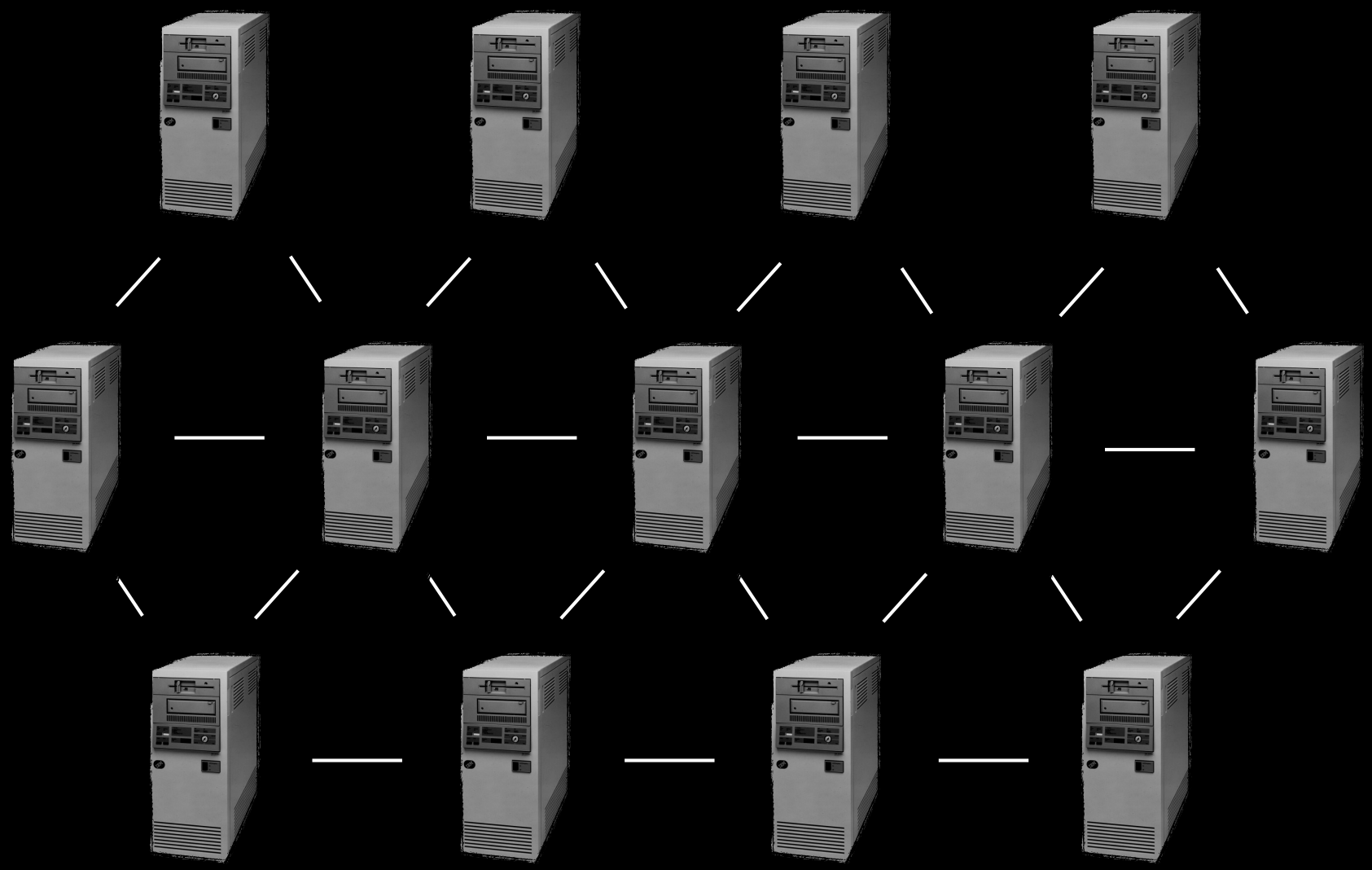
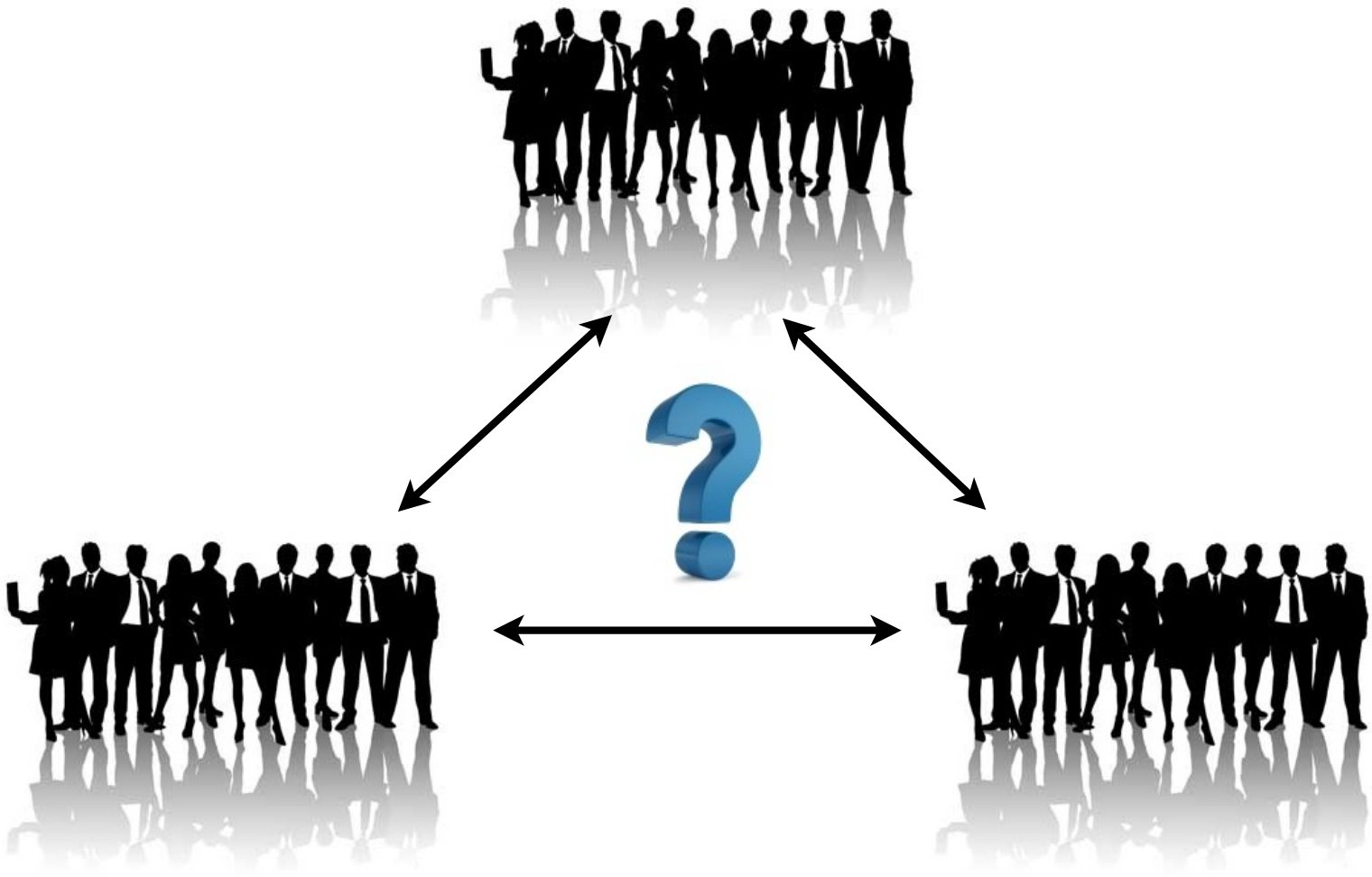


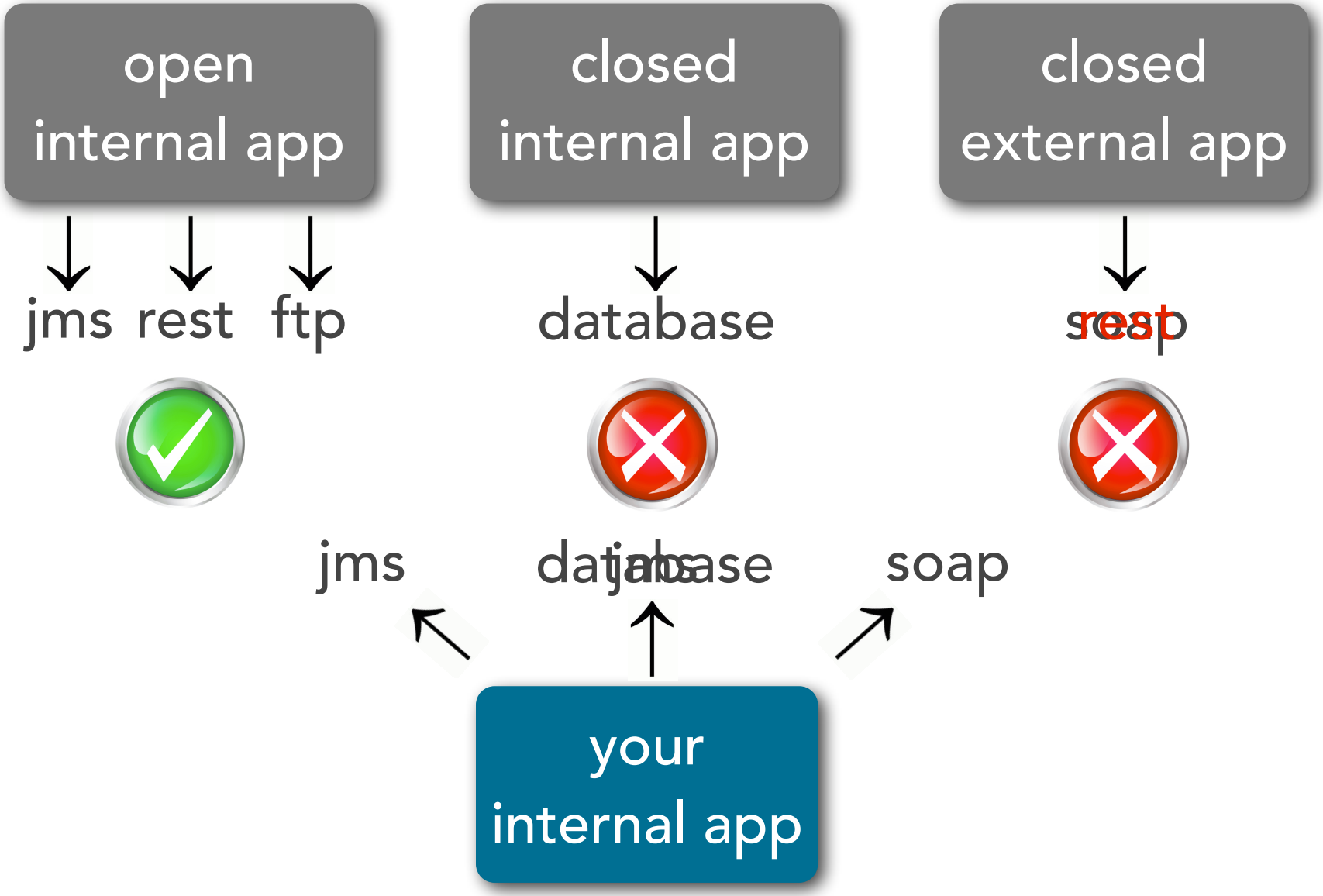
Integration Architecture Fundamentals




HP MediaSmart Server



challenges


open
internal app
closed
internal app
closed
external app
jms
rest
ftp
database
soap
rest
jms
database
soap
your
internal app

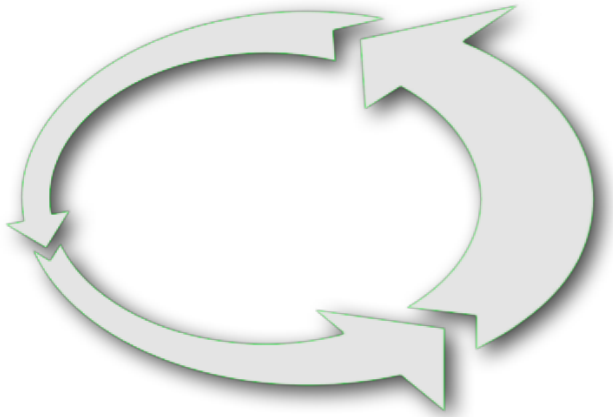
challenges

how do you deal with slower method invocations?

challenges

what do you do if a remote system is unavailable?

challenges



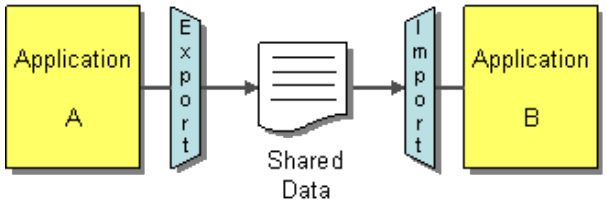
how do you deal with remote contract changes?

challenges

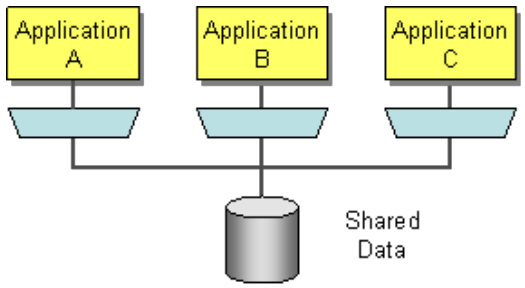
how do you make remote connections secure?

integration styles

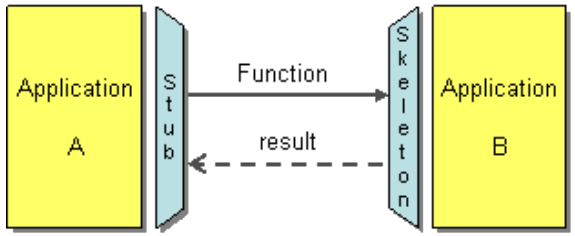
From Enterprise Integration Patterns by Hohpe and Woolf



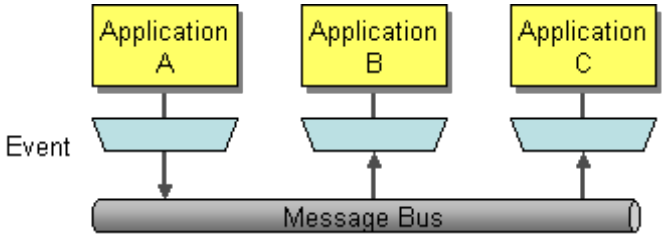
file transfer



shared database

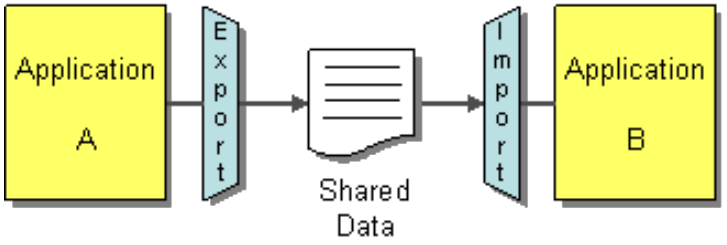


remote procedure invocation



messaging

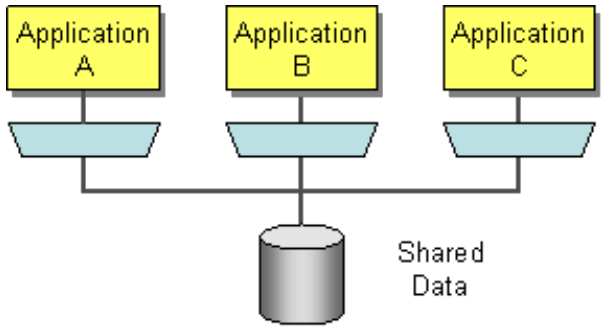
file transfer



universal integration style, integration simplicity, system decoupling and system abstraction

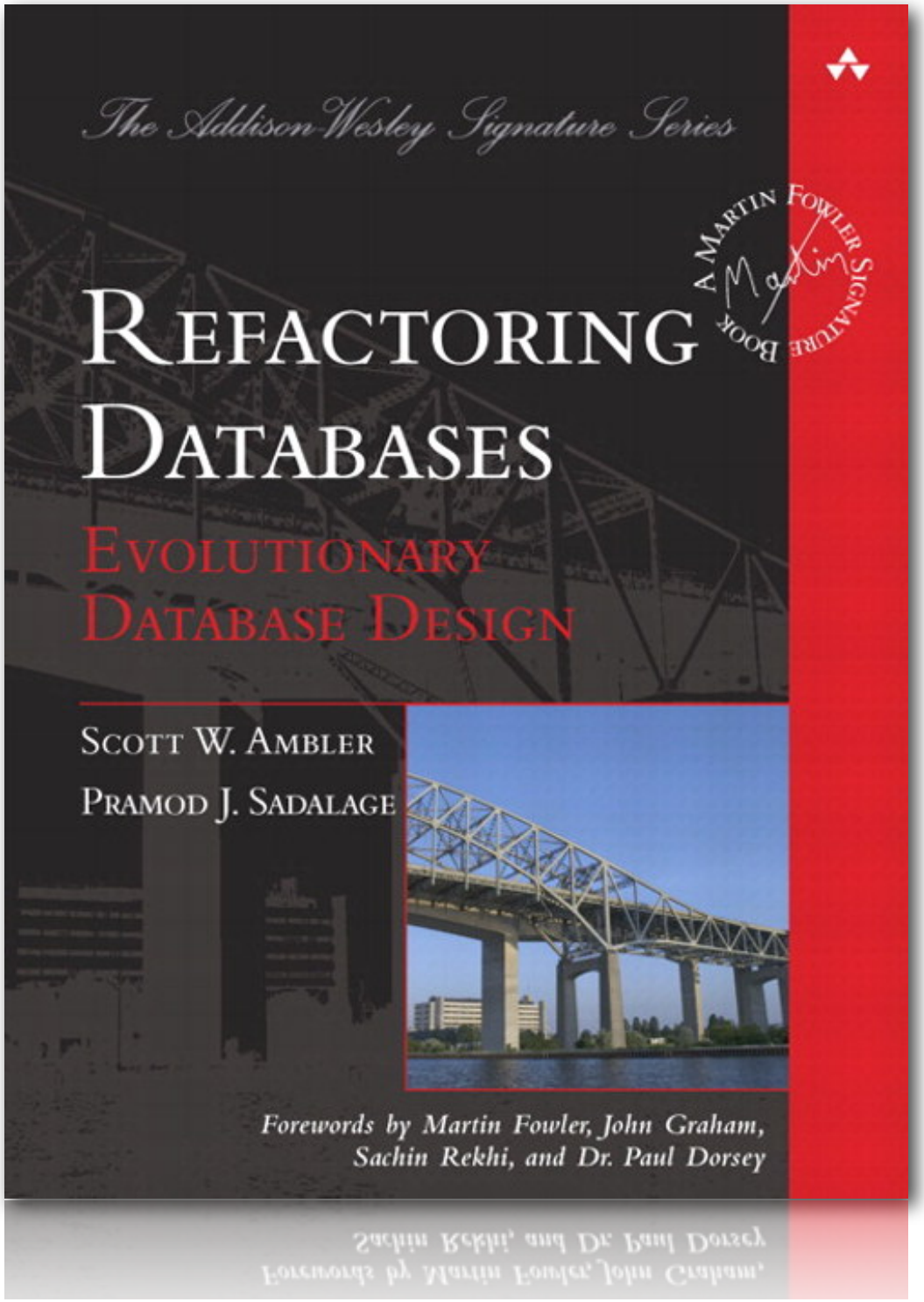
file-based processing is expensive, error processing, timeliness of data synchronization, data-only transfer

shared database

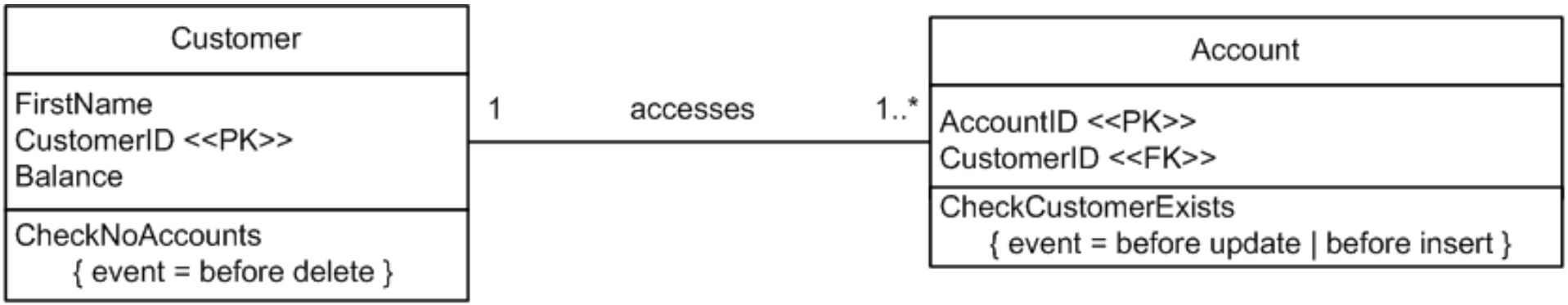


near-universal integration via SQL, system abstraction, system decoupling

cannot use persistence caching (ORM), performance bottleneck issues, schema change issues, data ownership issues

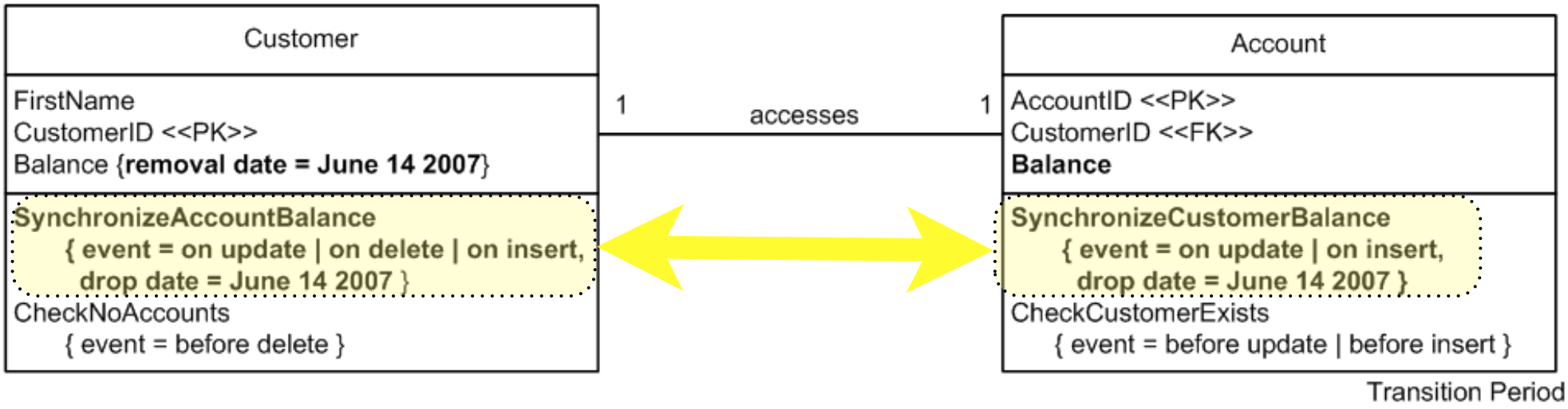

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage
Forewords by Martin Fowler, John Graham,
Sachin Rekhi, and Dr. Paul Dorsey

move column refactoring



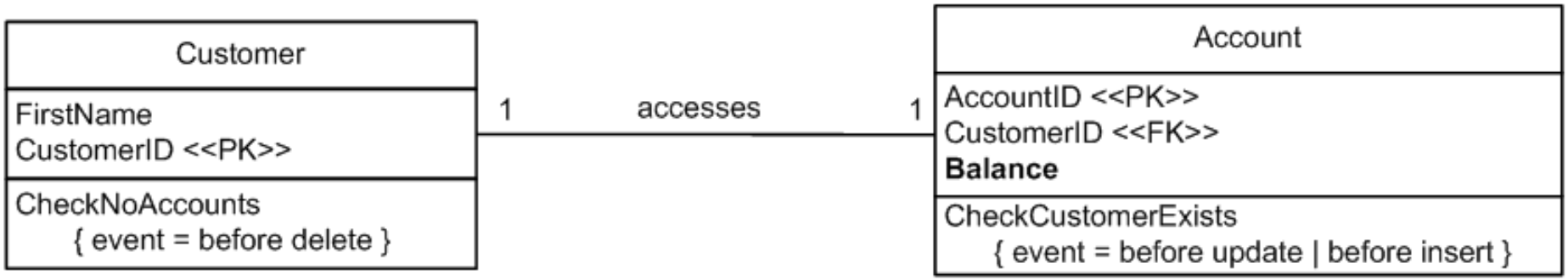
Original Schema

transition period



move column refactoring

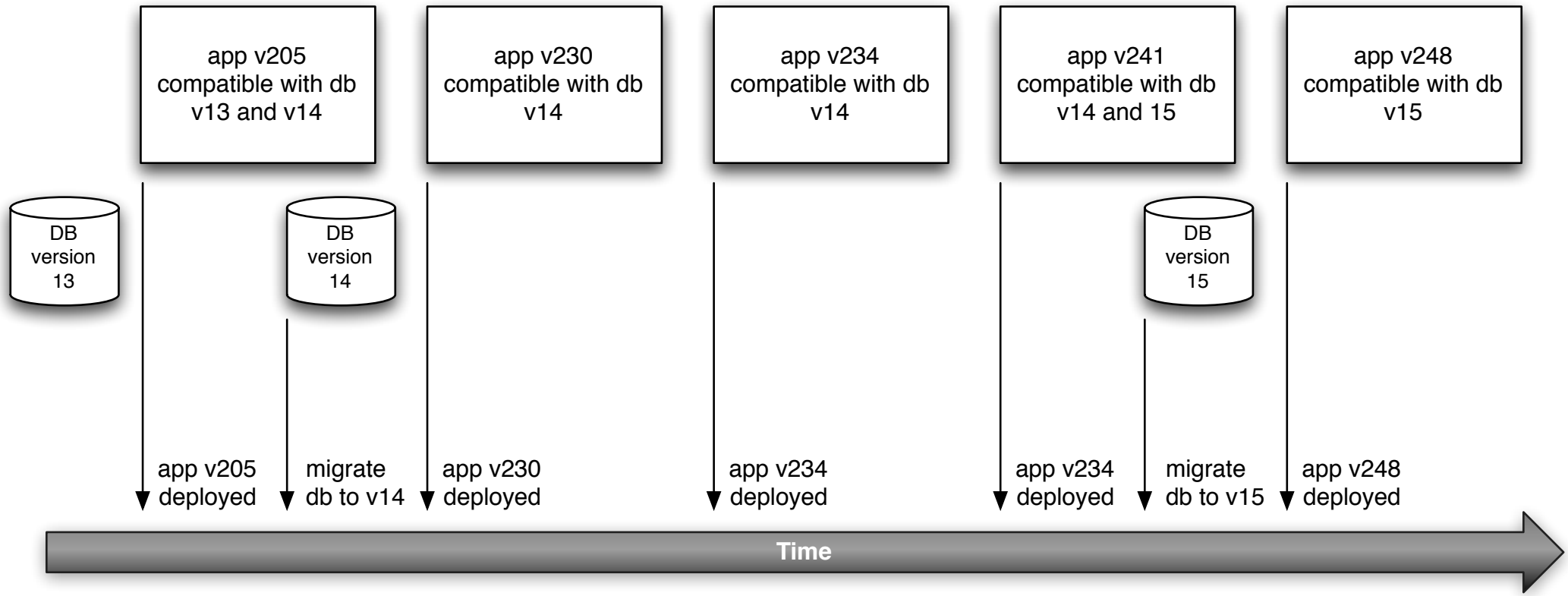
end schema



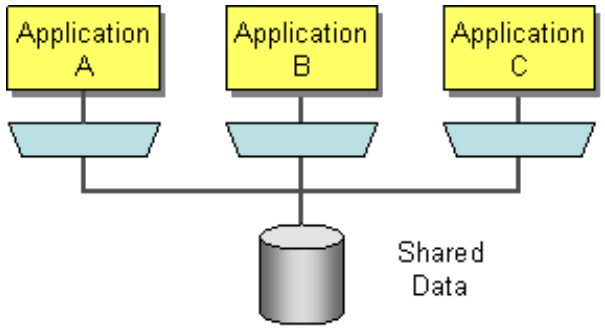
Resulting Schema

move column refactoring

decouple db updates



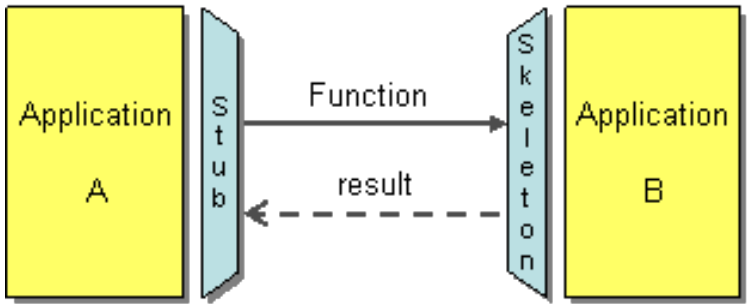
shared database



near-universal integration via SQL, system abstraction, system decoupling

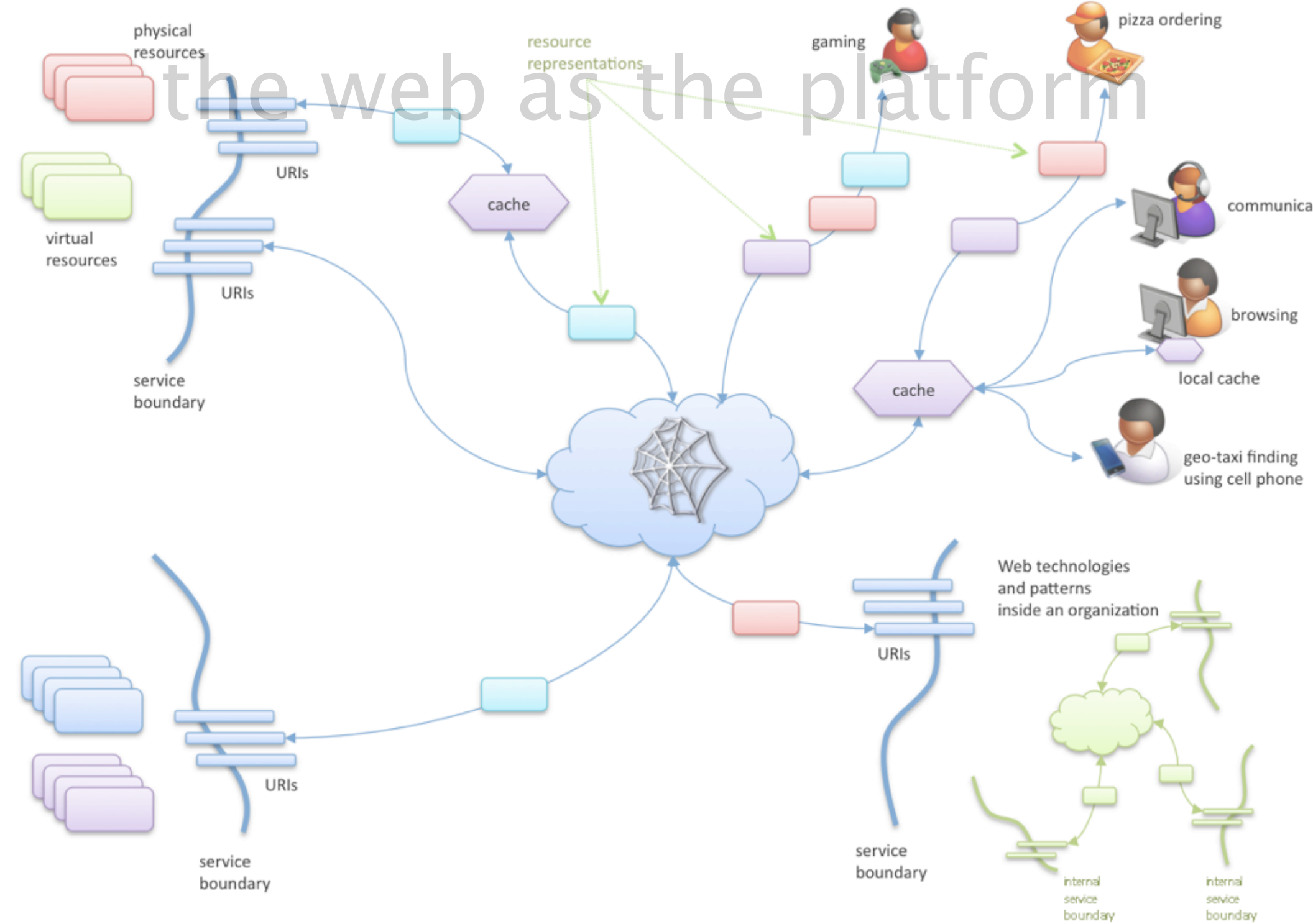
cannot use persistence caching (ORM), performance bottleneck issues, schema change issues, data ownership issues

remote procedure

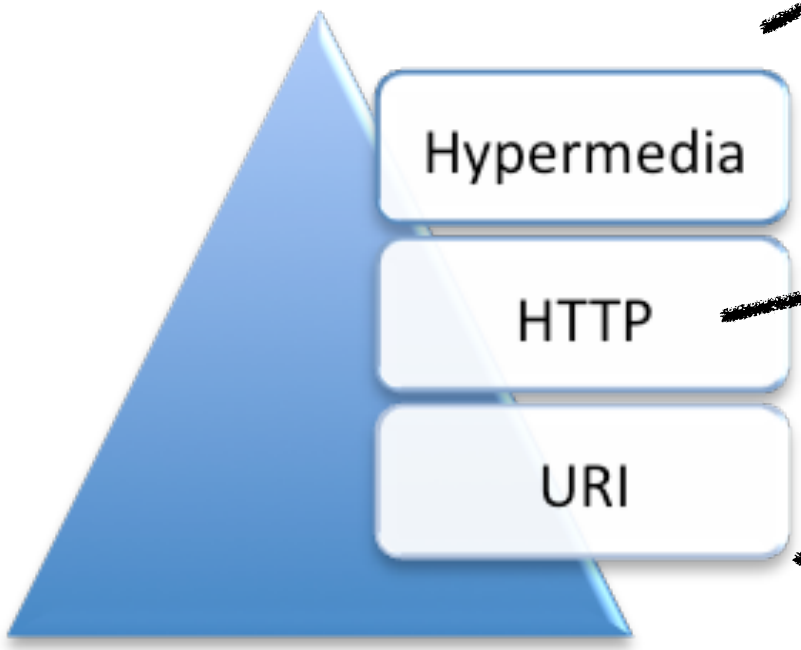


data encapsulation and ownership, external systems integration via web services, mature frameworks and tools

tight system coupling due to dependency on service availability and location knowledge, poor asynchronous communications


the web as the platform
physical resources
URIs
virtual resources
URIs
service boundary
resource representations
cache
gaming
pizza ordering
communica
browsing
local cache
cache
geo-taxi finding using cell phone
URIs
service boundary
URIs
service boundary
Web technologies and patterns inside an organization
internal service boundary
internal service boundary

Richardson's restful levels of maturity



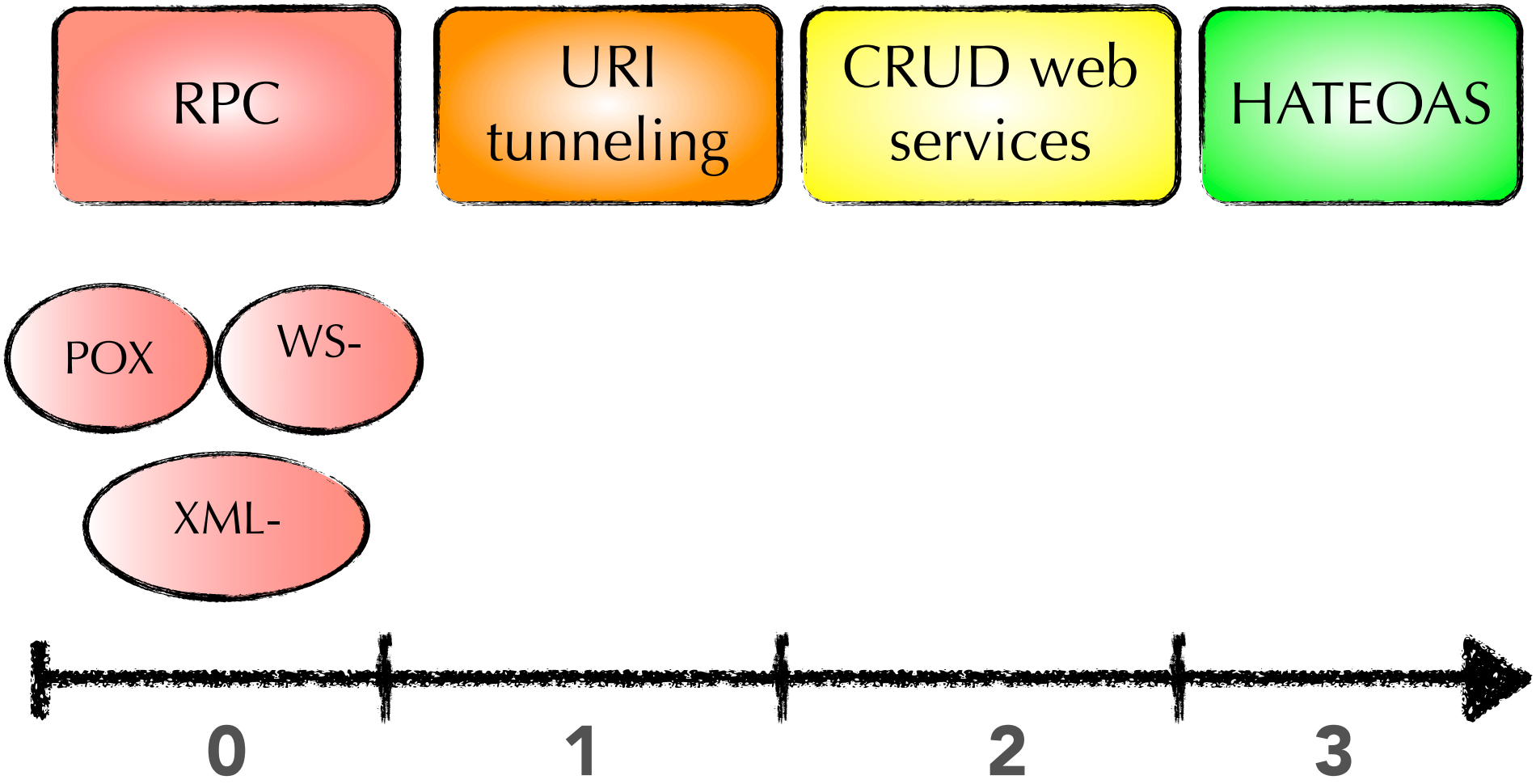
level 3:
HATEOAS

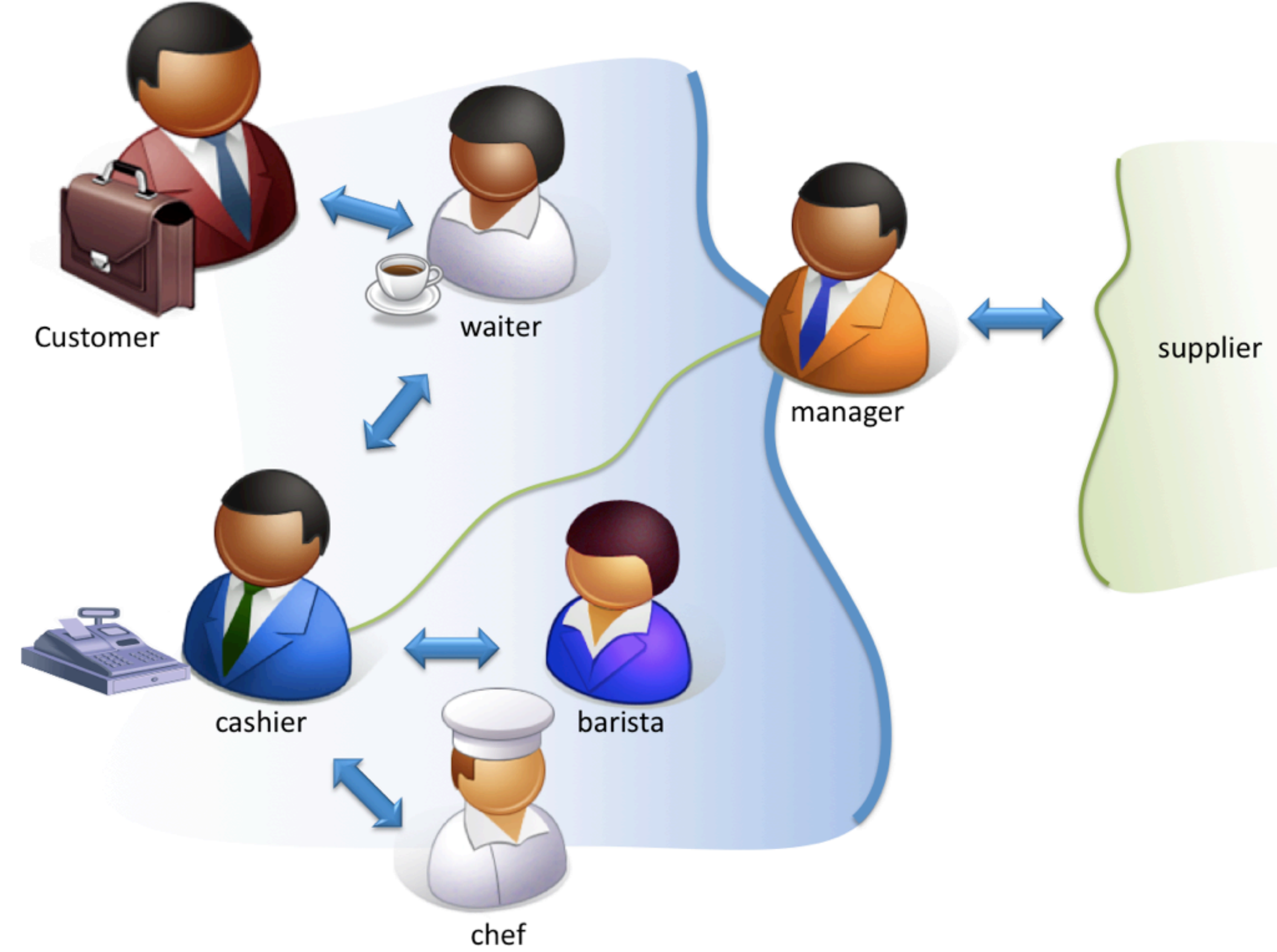
level 2:
numerous URI
addressable
resources

level 1:
many URIs
single HTTP verb

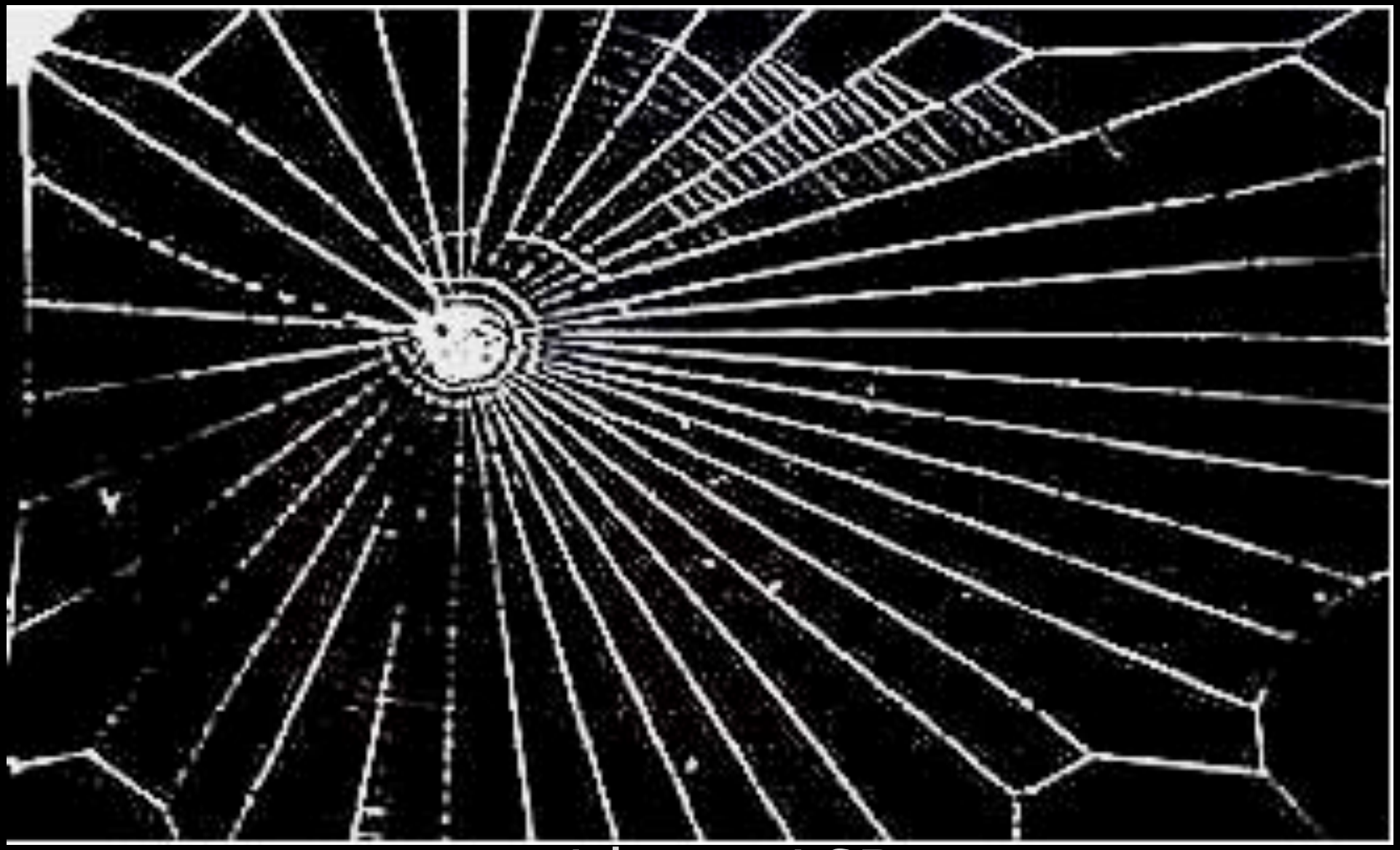
level 0:
single URI
single HTTP verb

evolution




Customer
waiter
manager
supplier
cashier
barista
chef

RPC Flavors



spider on LSD

WS-* / Plain Old XML

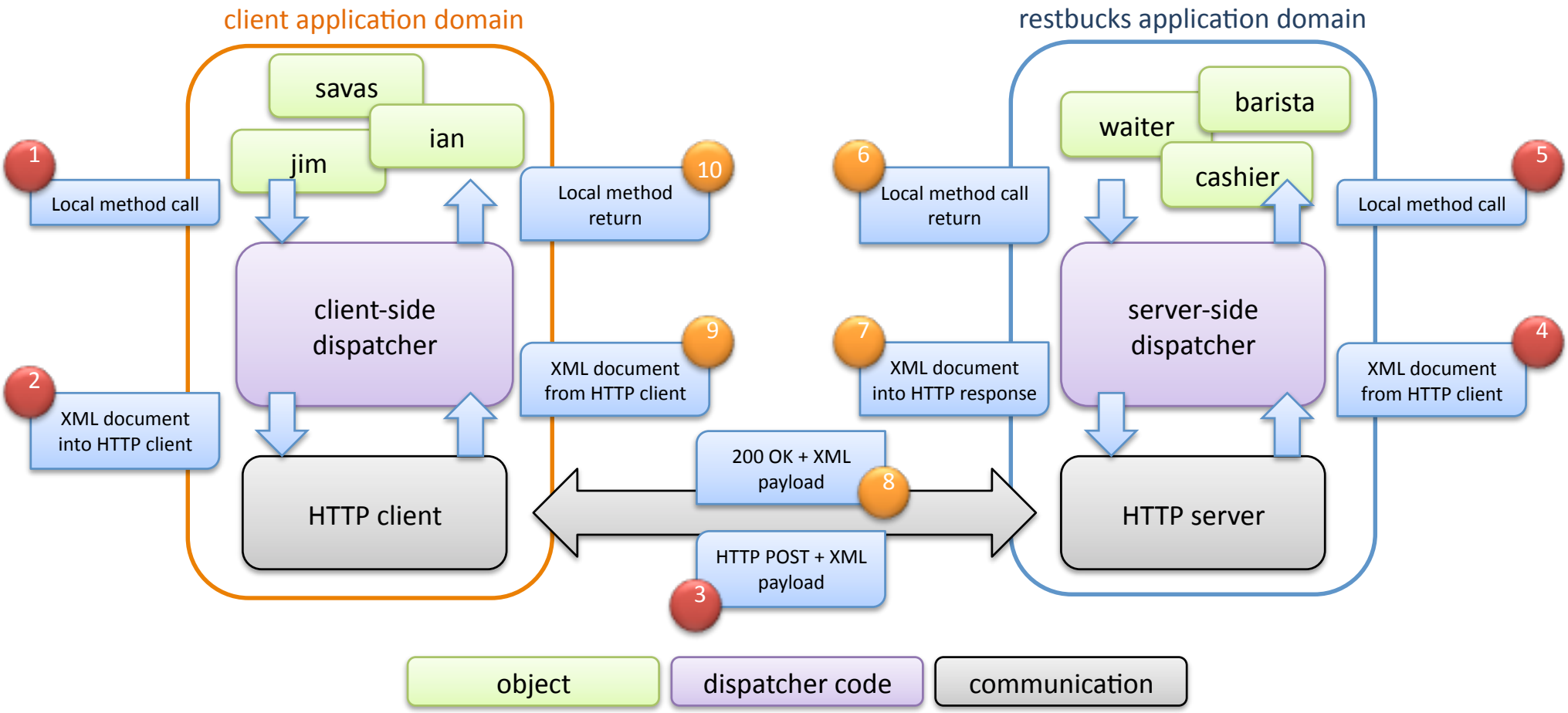
uses HTTP POST to transfer XML documents

all application semantics reside inside XML payload

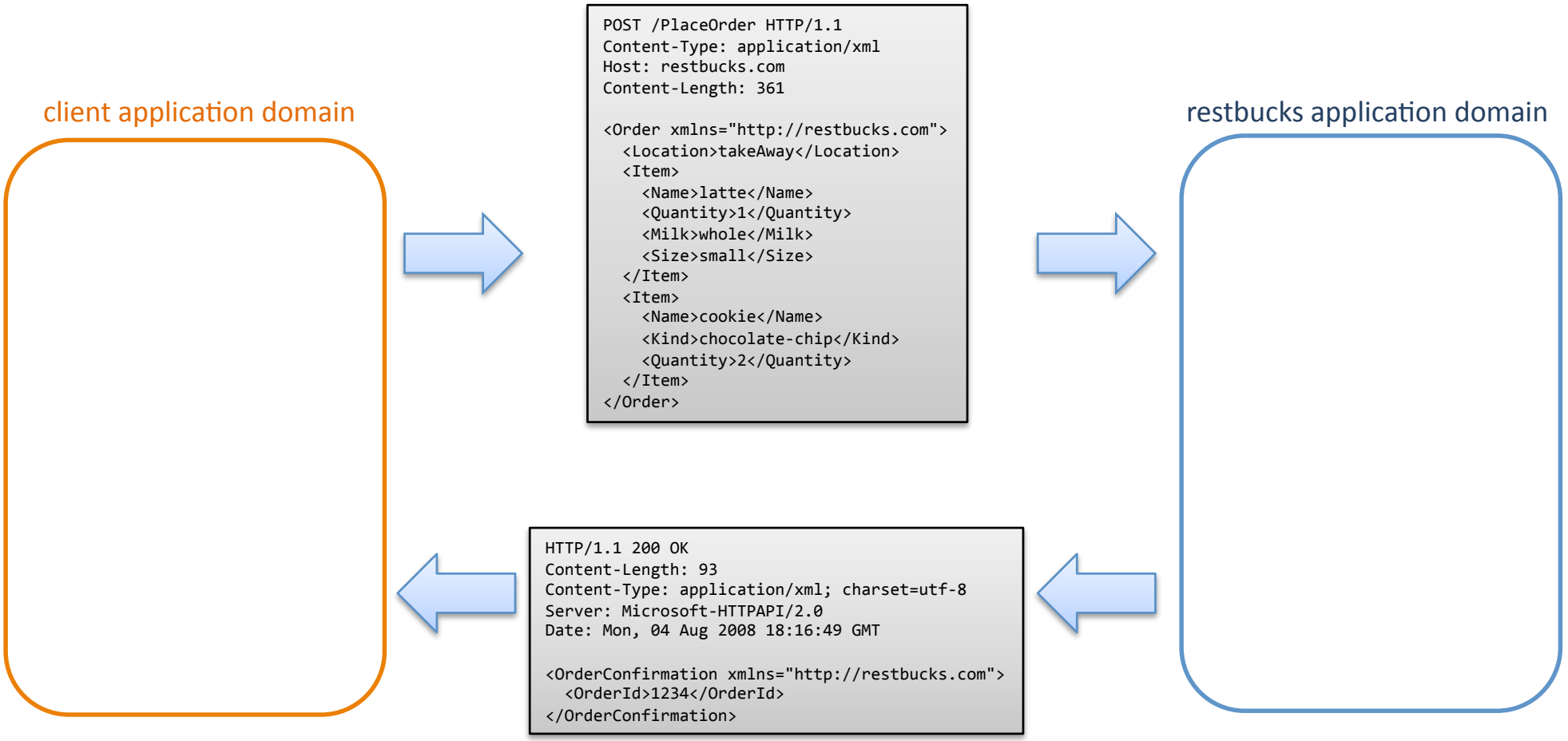
HTTP meta-data ignored

POX is a synchronous, firewall friendly transport protocol for convenience

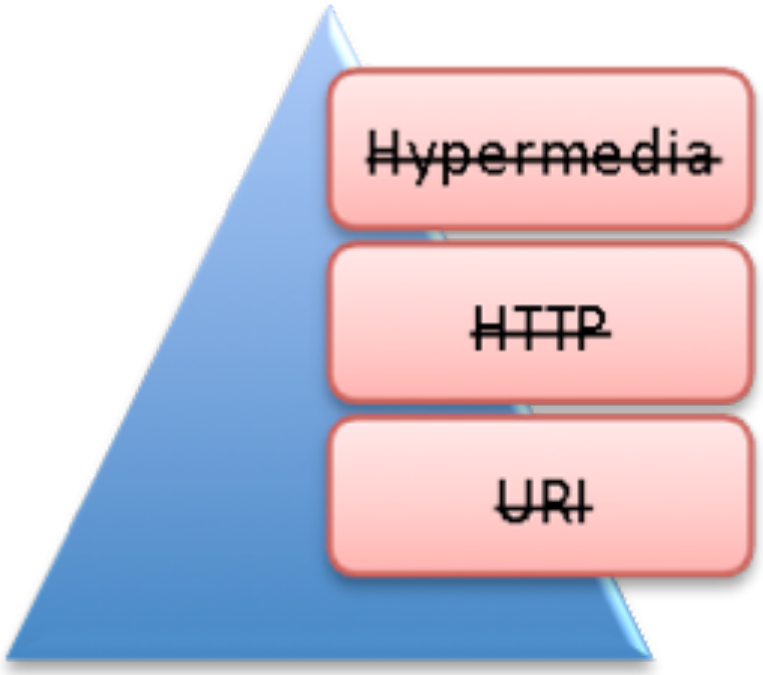
typical RPC interaction



RPC wire-level protocol



RPC architectural style



level 0

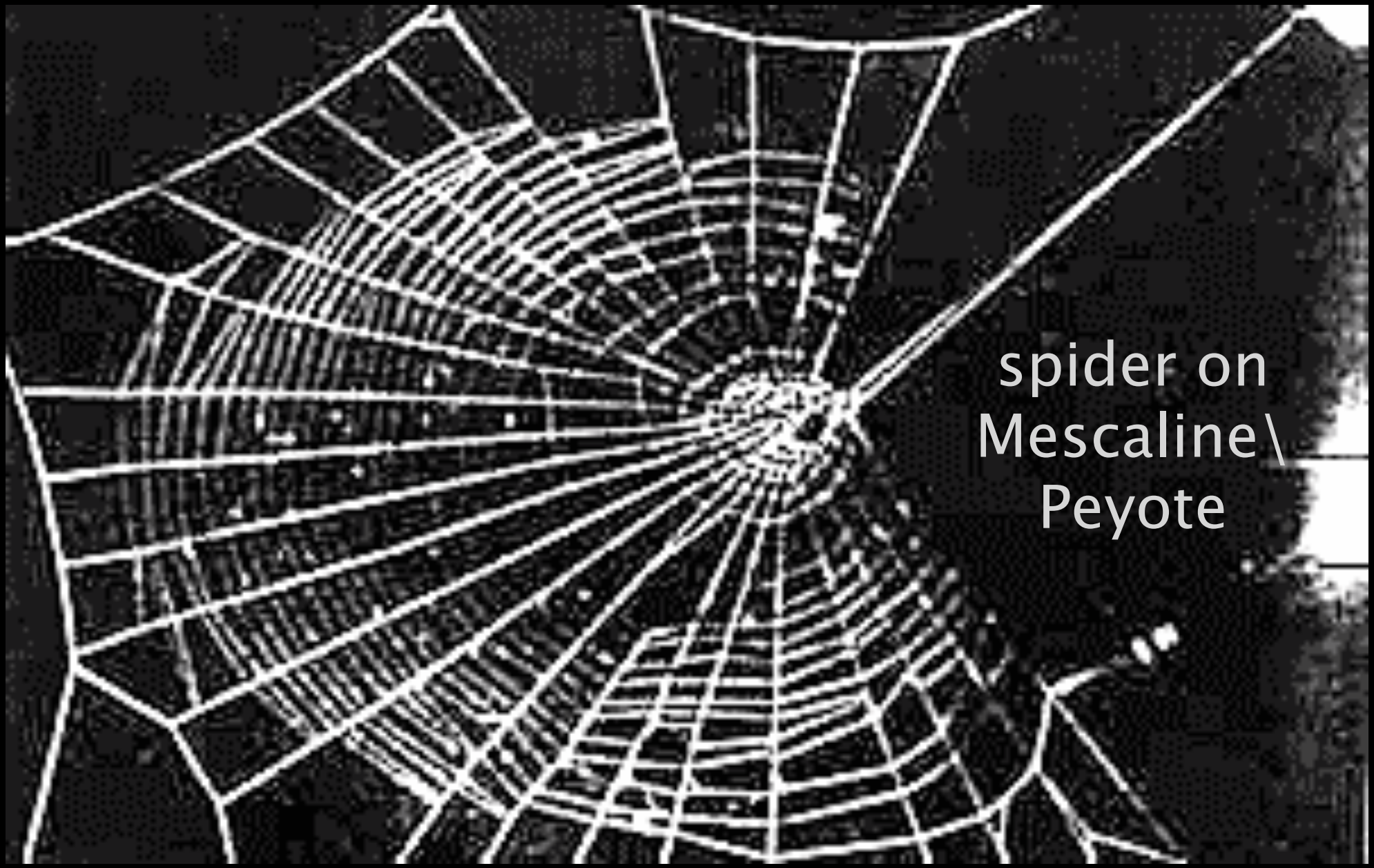
lightweight
almost universally
interoperable

not especially robust

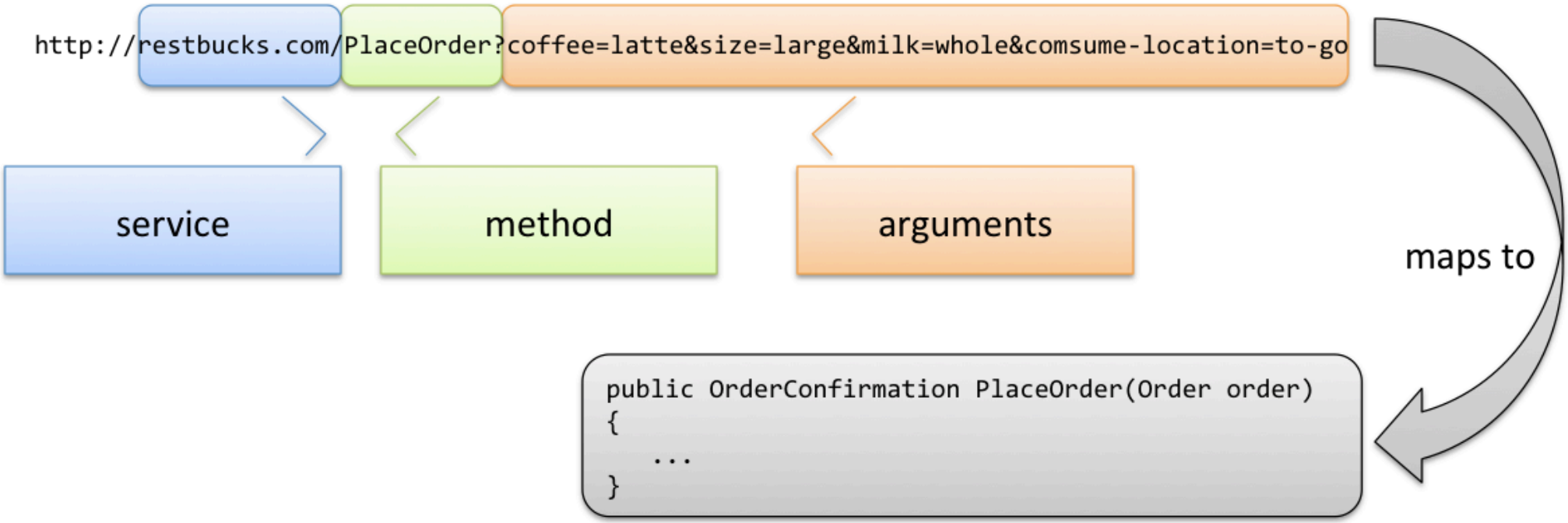
ignores the Web as a platform

limited to simpler integration problems

URI Tunneling

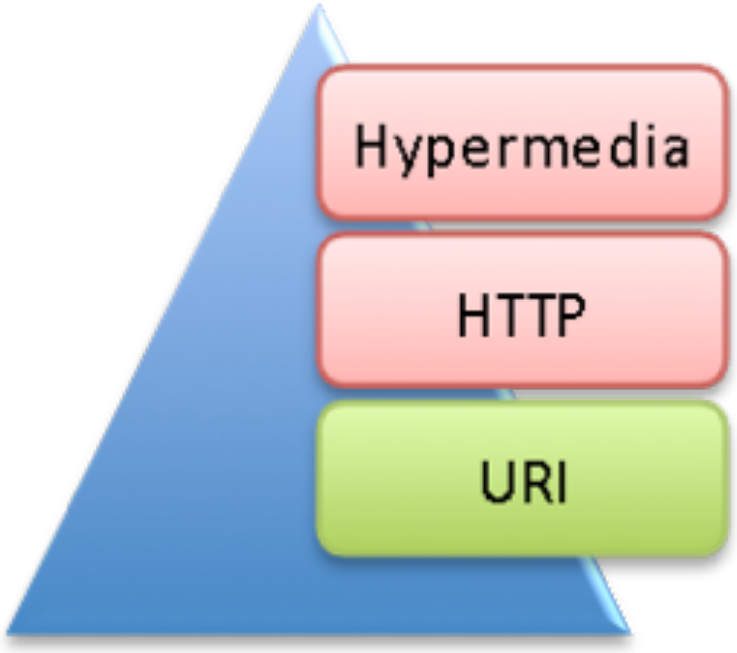


URI tunneling



uses URIs as a means of transferring information across system boundaries by encoding the information within the URI itself

URI tunneling?



uses only URIs

not sophisticated

can be web friendly

URIs encode operations rather than identify resources

`http://restbucks.com/PlaceOrder?1234`

GET placing order violates idempotence

CRUD web services

spider on marijuana

modeling orders as resources

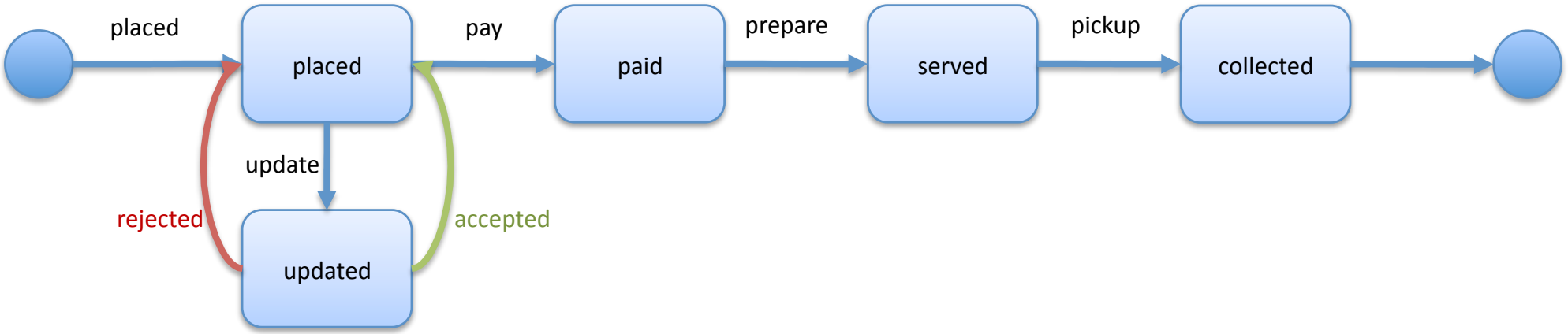
orders are created when a customer makes a purchase

orders are frequently read, particularly when their preparation status is inquired

under certain conditions, orders may be updated

if an order is still pending, a customer can cancel (or delete) it

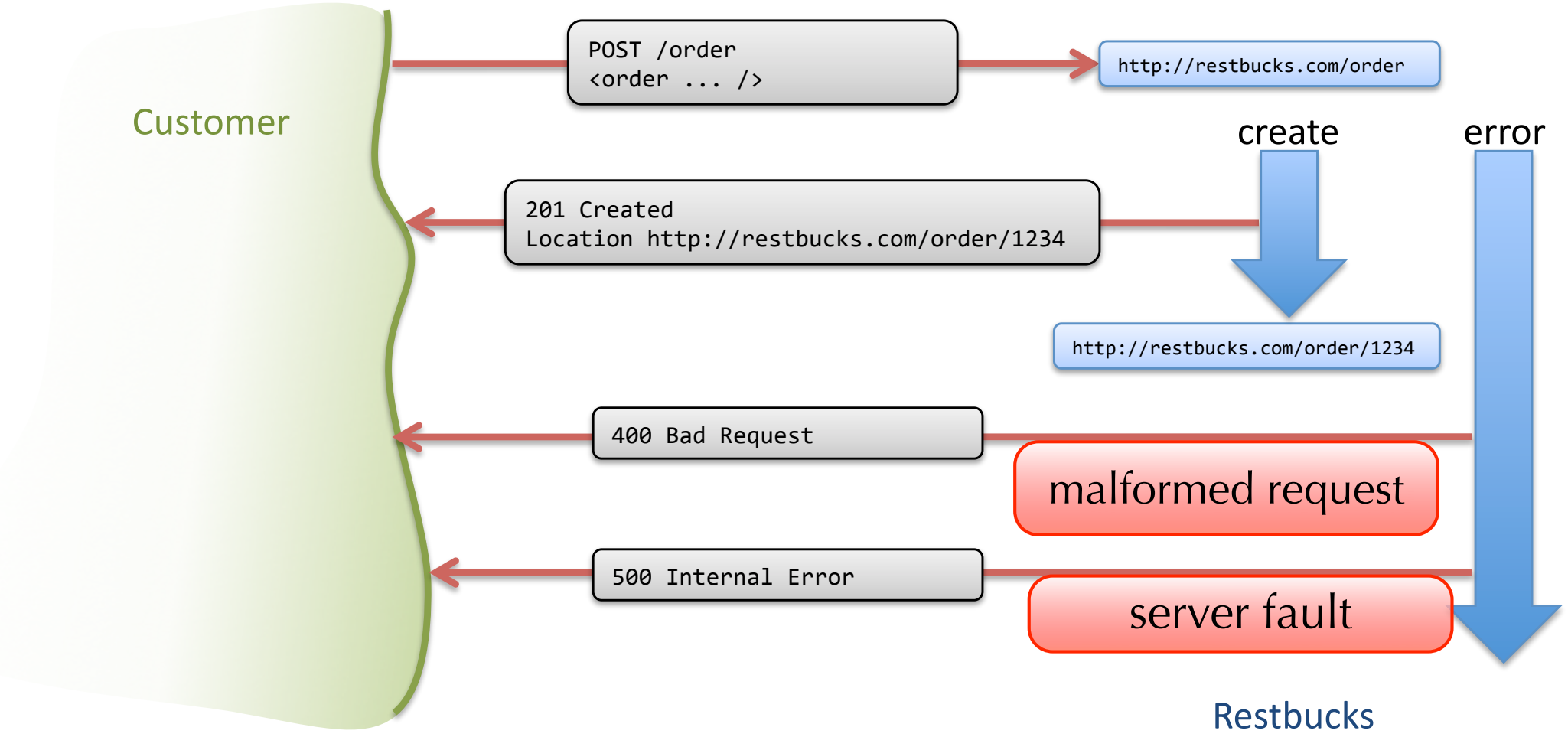
possible HTTP requests for Restbucks



ordering service contract

verb	URI or template	use
POST	**`/order`**	Create a new order and upon success receive a `Location` header specifying the new order's URI
GET	**`/order/{orderId}`**	Request the current state of the order specified by the URI
PUT	**`/order/{orderId}`**	Update an order at the given URI with new information providing the full representation
DELETE	**`/order/{orderId}`**	Logically remove the order identified by the given URI.

creating an order via POST

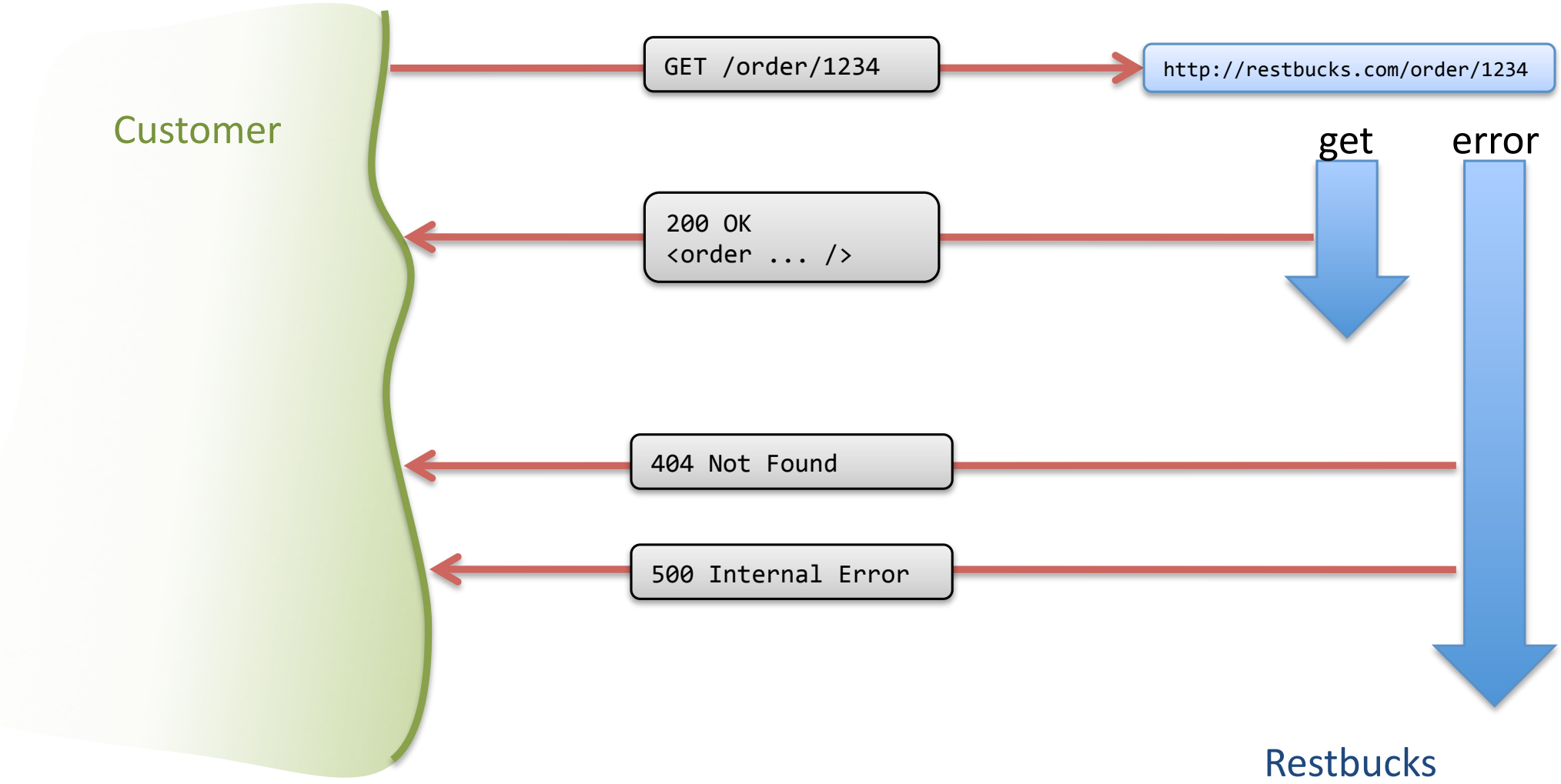


coffee order via POST

```
POST /order HTTP/1.1
Host: restbucks.com
Content-Type:application/xml
Content-Length: 239

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
    <item>
      <name>latte</name>
      <quantity>1</quantity>
      <milk>whole</milk>
      <size>small</size>
    </item>
  </items>
</order>
```

reading via GET

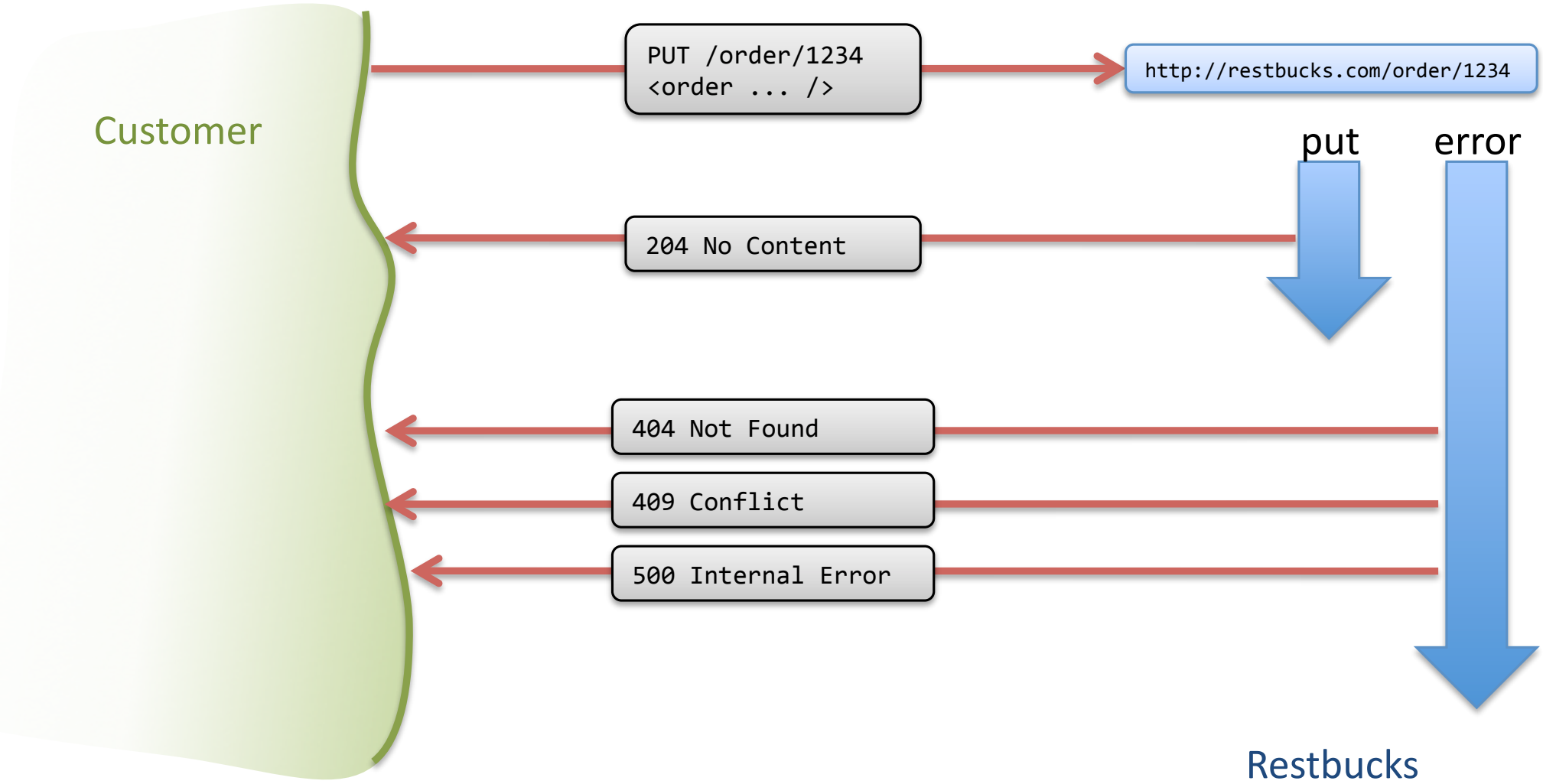


GET implications

```
HTTP/1.1 200 OK
Content-Length: 265
Content-Type: application/xml
Date: Wed, 19 Nov 2008 21:58:21 GMT

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
    <item>
      <name>latte</name>
      <quantity>1</quantity>
      <milk>whole</milk>
      <size>small</size>
    </item>
  </items>
  <status>served</status>
</order>
```

updating with PUT



handling conflicts

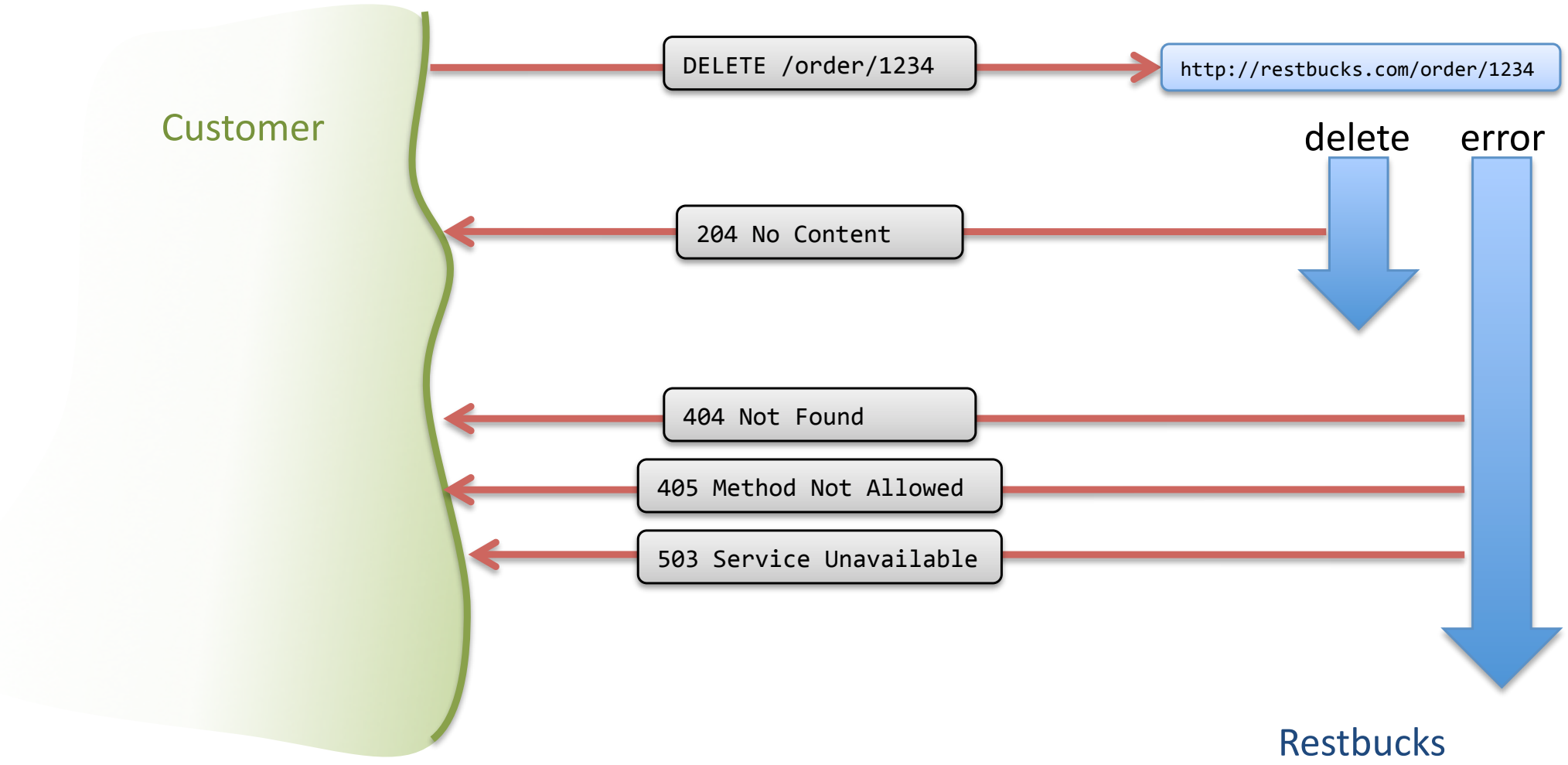
```
HTTP/1.1 409 Conflict
Date: Sun, 21 Dec 2008 16:43:07 GMT
Content-Length:271

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
     <item>
      <milk>whole</milk>
      <name>cappuccino</name>
      <quantity>1</quantity>
      <size>large</size>
    </item>
  </items>
  <status>served</status>
</order>
```

PUT, GET, & DELETE are idempotent

allows simple retries for failure states

DELETE


Customer
DELETE /order/1234
http://restbucks.com/order/1234
delete
error
204 No Content
404 Not Found
405 Method Not Allowed
503 Service Unavailable
Restbucks

aligning resource state

inevitable state conflicts in distributed applications

entity tags (ETags) — opaque string token that server associates with a resource to uniquely identify the state of the resource over its lifetime

using ETags

```
GET /order/1234 HTTP/1.1
Host: restbucks.com

HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 275
ETag: "72232bd0daafa12f7e2d1561c81cd082"

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
    <item>
      <milk>skim</milk>
      <name>cappuccino</name>
      <quantity>1</quantity>
      <size>large</size>
    </item>
  </items>
  <status>pending</preparing>
</order>
```

1st customer PUTs order

```
PUT /order/1234 HTTP/1.1
Host: restbucks.com
If-Match: "72232bd0daafa12f7e2d1561c81cd082"

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
    <item>
      <milk>whole</milk>
      <name>cappuccino</name>
      <quantity>1</quantity>
      <size>large</size>
    </item>
  </items>
  <status>pending</preparing>
</order>

HTTP/1.1 204 No Content
ETag: "6e87391fdb5ab218c9f445d61ee781c1"
```

2nd customer PUTs

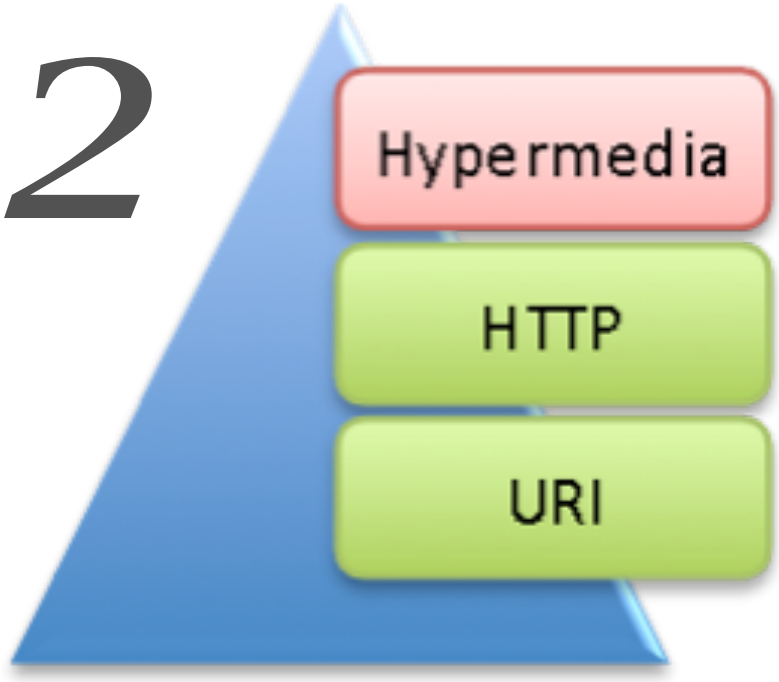
```
PUT /order/1234 HTTP/1.1
Host: restbucks.com
If-Match: "72232bd0daafa12f7e2d1561c81cd082"

<order xmlns="http://schemas.restbucks.com/order">
  <location>takeAway</location>
  <items>
    <item>
      <milk>skim</milk>
      <name>cappuccino</name>
      <quantity>2</quantity>
      <size>large</size>
    </item>
  </items>
  <status>pending</preparing>
</order>

HTTP/1.1 412 Precondition Failed
```

CRUD services

2



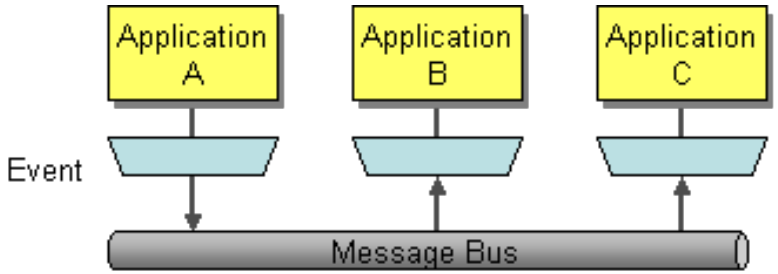
embraces both HTTP & URIs

viable, robust, and easily implemented solution for some problem domains

CRUD is best for...CRUD

shared, tightly coupled, understanding of resource life cycles

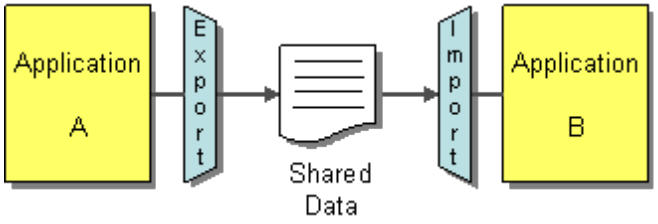
messaging



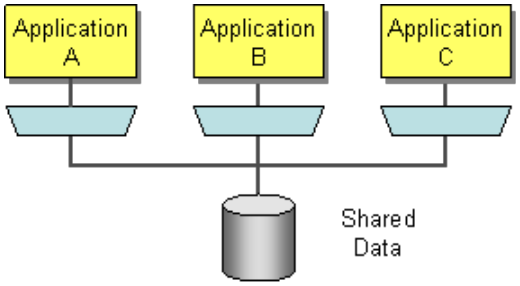
asynchronous and reliable messaging, highly decoupled systems, excellent scalability capabilities, monitoring

external integration beyond firewall, implementation and testing complexity, cross platform standards still evolving

which is the best integration style?

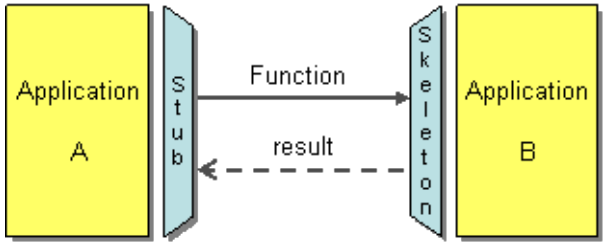


file transfer

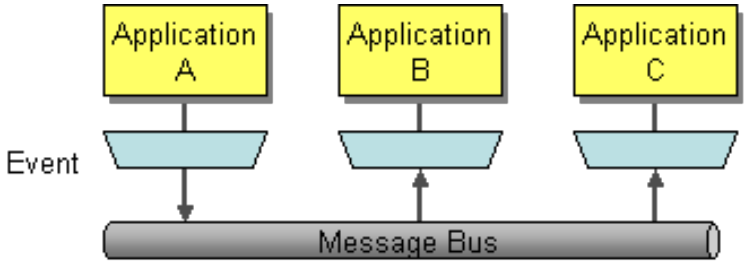


shared database

?

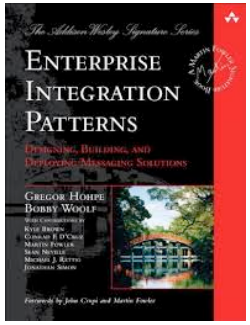


remote procedure invocation



messaging

for more information



Enterprise Integration Patterns
Gregor Hohpe and Bobby Woolf
http://www.enterpriseintegrationpatterns.com/
http://www.amazon.com/dp/0321200683

Java Message Service, 2nd Edition
Mark Richards, O'Reilly, 2009
http://oreilly.com/catalog/9780596522049/index.html

?’S

Mark Richards
Independent Consultant
Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

Published Books:
Java Message Service, 2nd Edition
97 Things Every Software Architect Should Know
Java Transaction Design Strategies

Neal Ford
Director / Software Architect /
Meme Wrangler

ThoughtWorks®

2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com