Università della Svizzera italiana

Faculty of Informatics

RESTunconf

Munich, January 2011

Atomic Transactions for the REST of us

Cesare Pautasso
Faculty of Informatics
University of Lugano, Switzerland

c.pautasso@ieee.org
http://www.pautasso.info
@pautasso

Acknowledgements

- This is joint work with Guy Pardon, Atomikos

http://www.atomikos.com

Does REST need transactions?

- If you find yourself in need of a distributed transaction protocol, then how can you possibly say that your architecture is based on REST? I simply cannot see how you can get from one situation (of using RESTful application state on the client and hypermedia to determine all state transitions) to the next situation of needing distributed agreement of transaction semantics wherein the client has to tell the server how to manage its own resources.
- ...for now I consider "rest transaction" to be an oxymoron.

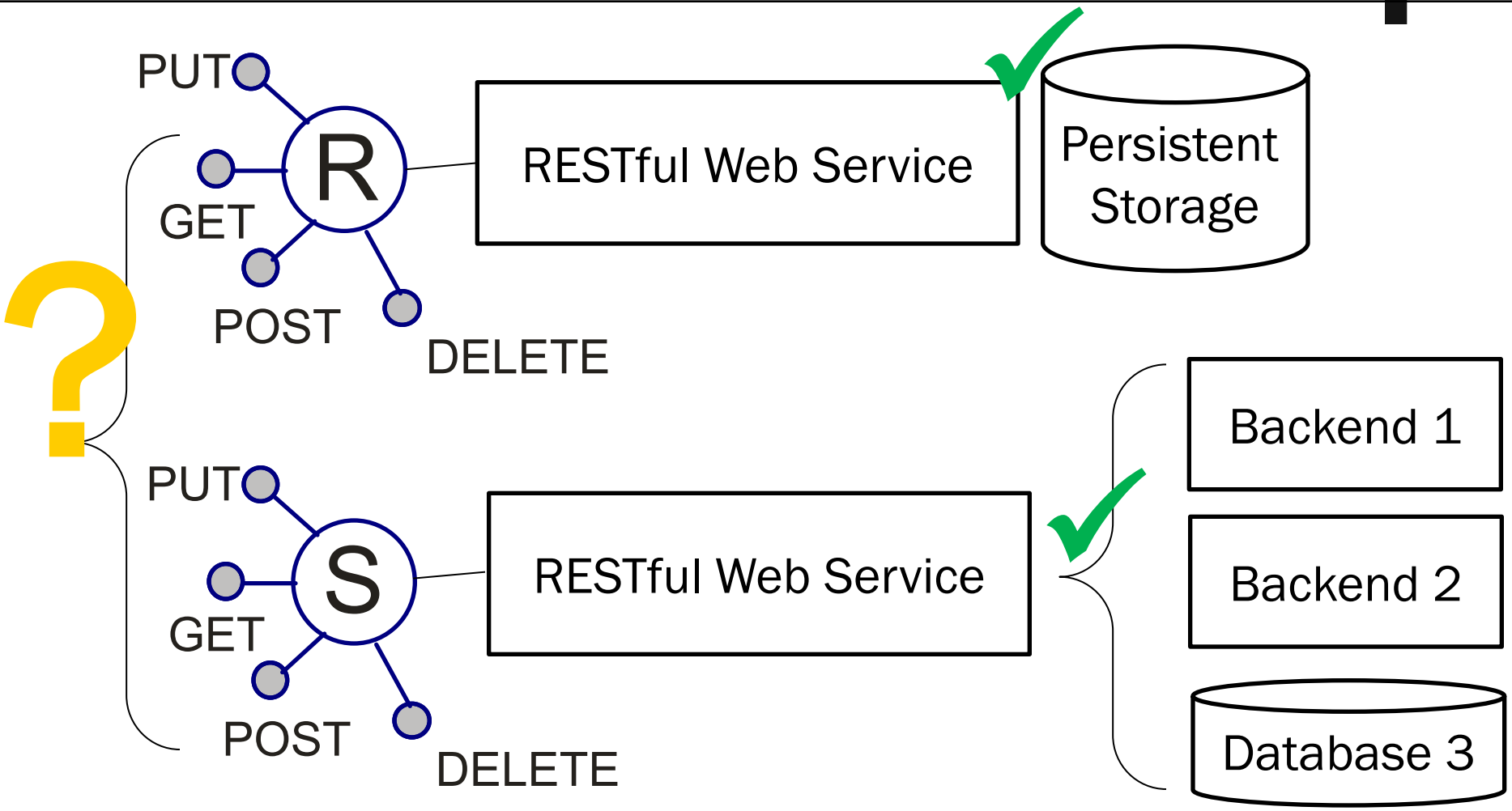
Roy Fielding, REST discuss, June 9th, 2009

Does REST need transactions?

- The typical conversation thread, real or virtual, about transactions over HTTP goes something like this (elided for brevity):
  - **"You don't want transactions over HTTP"**
  - But I need to organize number of steps into a single unit I can deal with easily.
  - **"OK, but you *don't need* transactions over HTTP"**
  - But I need the ability to back out changes in multiple locations safely and consistently.
  - **"OK, but you *can't do* transactions over HTTP!"**
  - Really?
- And here the topic usually dies or descends into a heated debate.

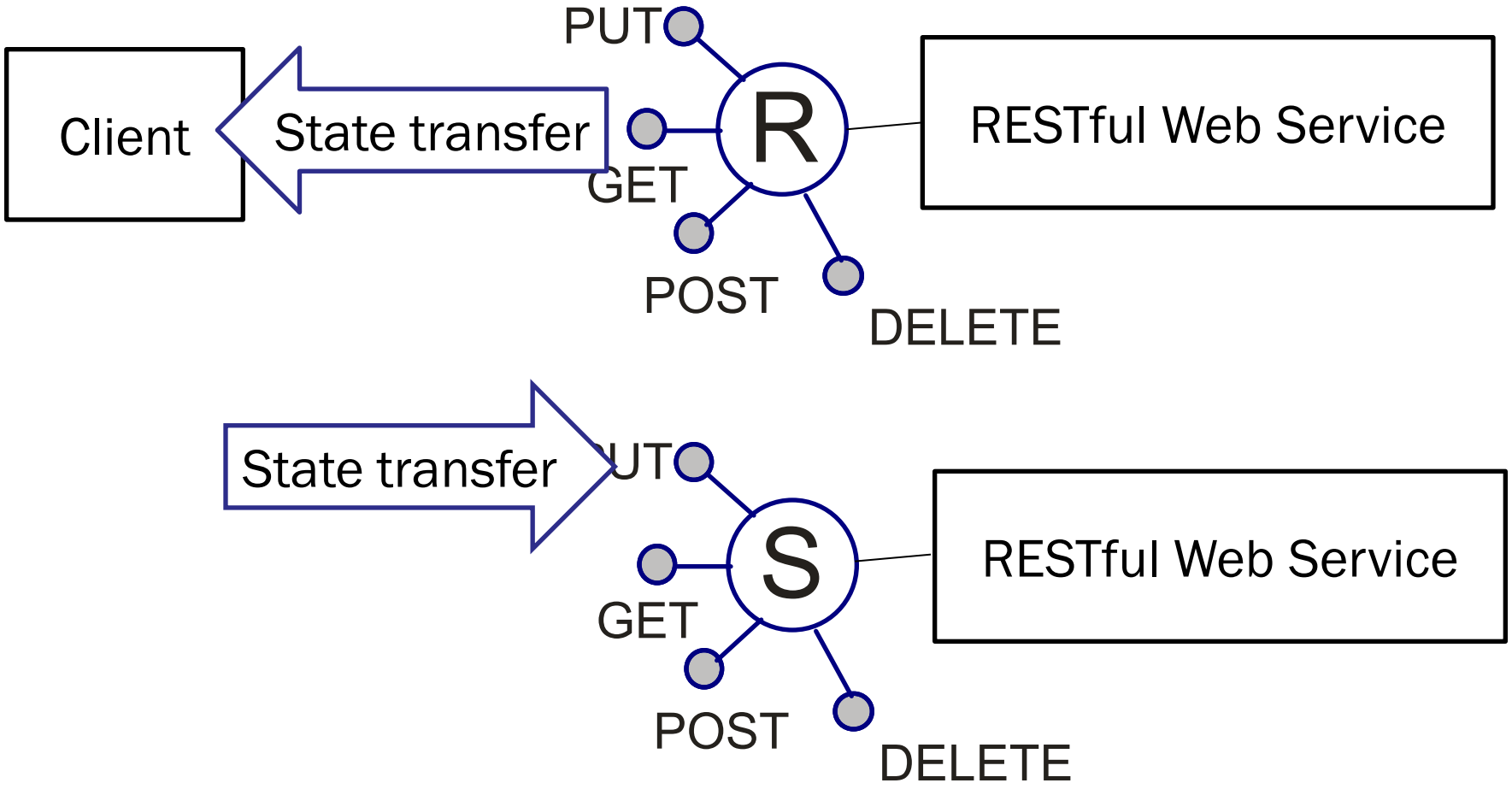
Mike Amudsen,
http://amundsen.com/blog/archives/1024

Context



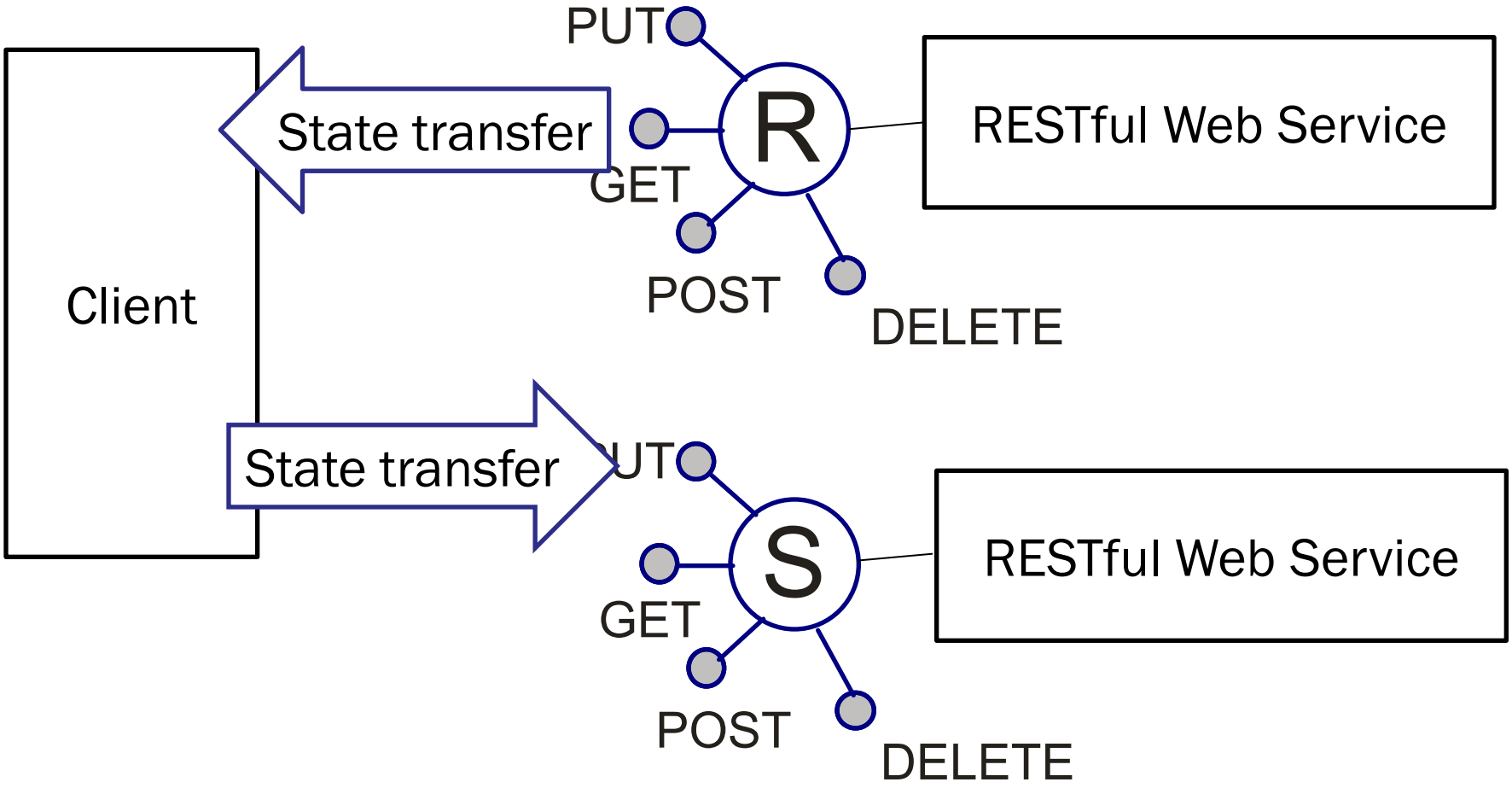
Adapted from Stefan Tilkov, Using REST for SOA, QCon SFO 2010

The problem



Thanks to the idempotency of GET/PUT,
each individual state transfer is reliable and atomic

The problem



How to we make both interactions atomic?

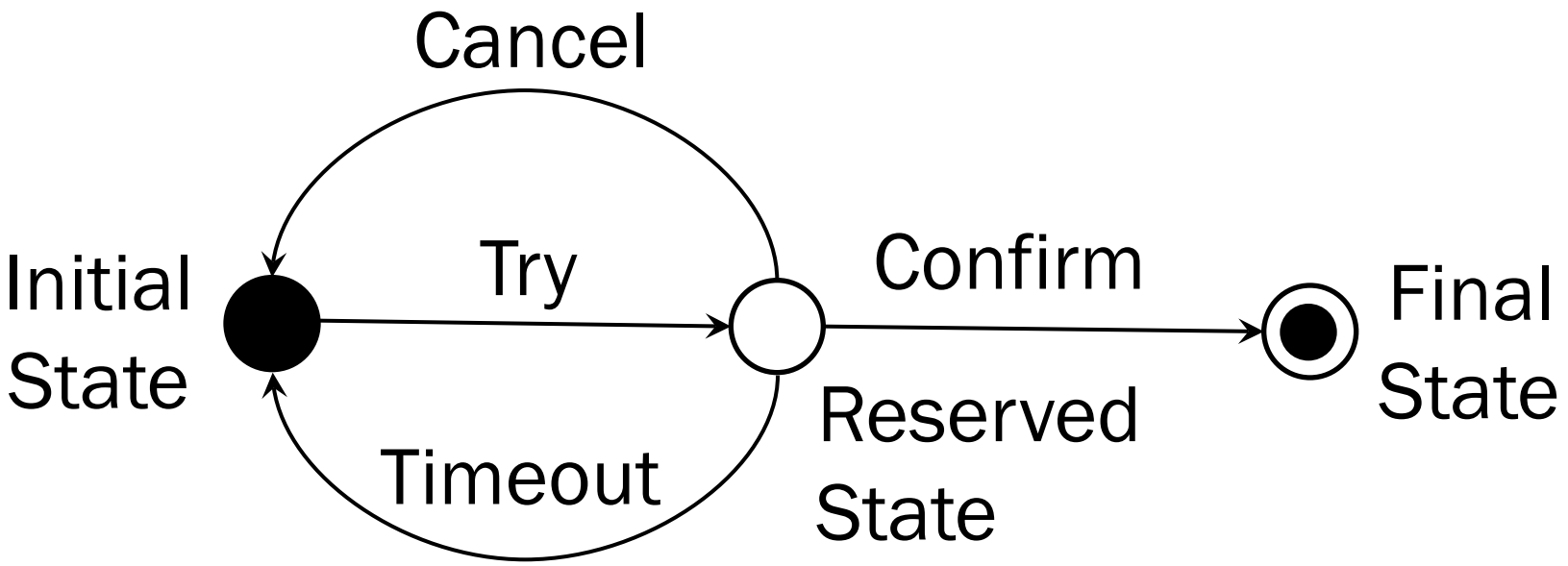
Constraints

- Interoperability:
  - No changes/extensions to HTTP
    - No additional verbs
    - No special/custom headers
- Loose Coupling:
  - REST shifts all the “work” to the client
  - RESTful Web services *should remain unaware* they are participating in a transaction
- Simplicity:
  - Transactions will not be adopted in practice unless they can be made simple enough

Assumption: Try-Confirm/Cancel

Università della Svizzera italiana

- Resource state transitions follow the TCC pattern



- Before they are made permanent state transitions go through an intermediate "reserved" state which either will be confirmed or canceled by a client within a given time
- Hint: Cancel/Confirm are idempotent

Example: Flight Booking Resource

Università della Svizzera italiana

Try

- POST /booking
- 302 Location: /booking/X

Reserve the flight

URI of the reserved state

Confirm

- PUT /booking/X
- 200

Pay and confirm the flight

Cancel

- DELETE /booking/X
- 200

Cancel the reservation

Protocol

1. A client interacts with multiple RESTful Web services. Interactions may lead to state transitions (the intermediate state is identified by a URI known to the client)
2. Once the client has completed all interactions, it uses the URIs identifying the intermediate states to confirm the state transitions (and thus commit the transaction)

Note: If the client stops before step 2, the state transitions will eventually be undone by the services themselves (after a timeout).
As an optimization, the client can use the same URIs to cancel the state transitions (and thus explicitly rollback the transaction).

Simple Example

1. POST www.swiss.ch/booking
   302 Location: /booking/1

   POST www.ezyj.com/booking
   302 Location:/booking/X

2. PUT www.swiss.ch/booking/1
   200

   PUT www.ezyj.com/booking/X
   200

What if something fails?

Università della Svizzera italiana

1. P0S… …ch/booking
   …king/1
   P… …n/b…
   302 Location:/booking/…

Whatever happens, these state transitions are temporary.

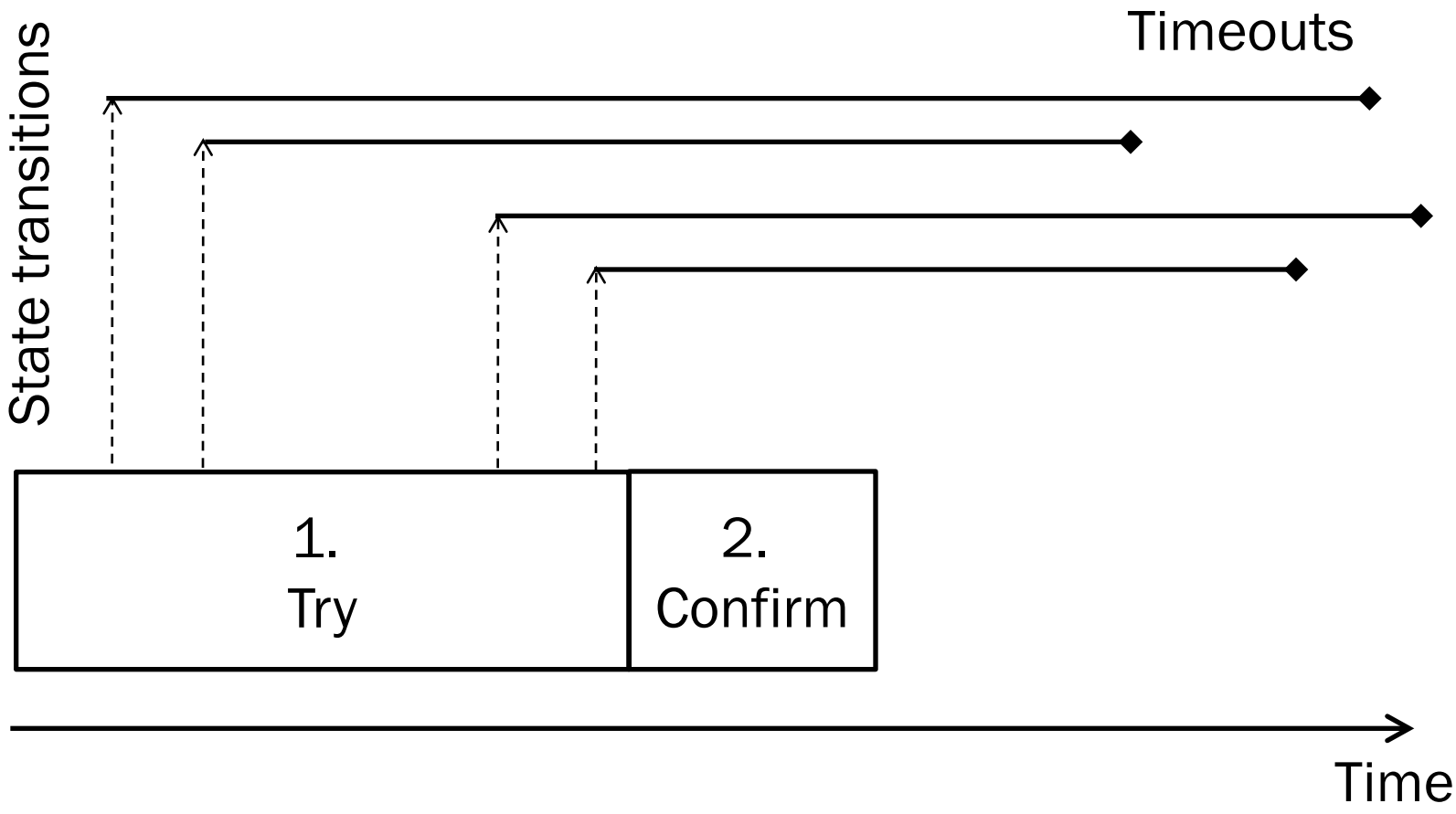
If something fails, stop before moving to phase 2

2. PUT … …/booking/1
   P… …o…
   200

Only idempotent methods are allowed in the confirmation phase.

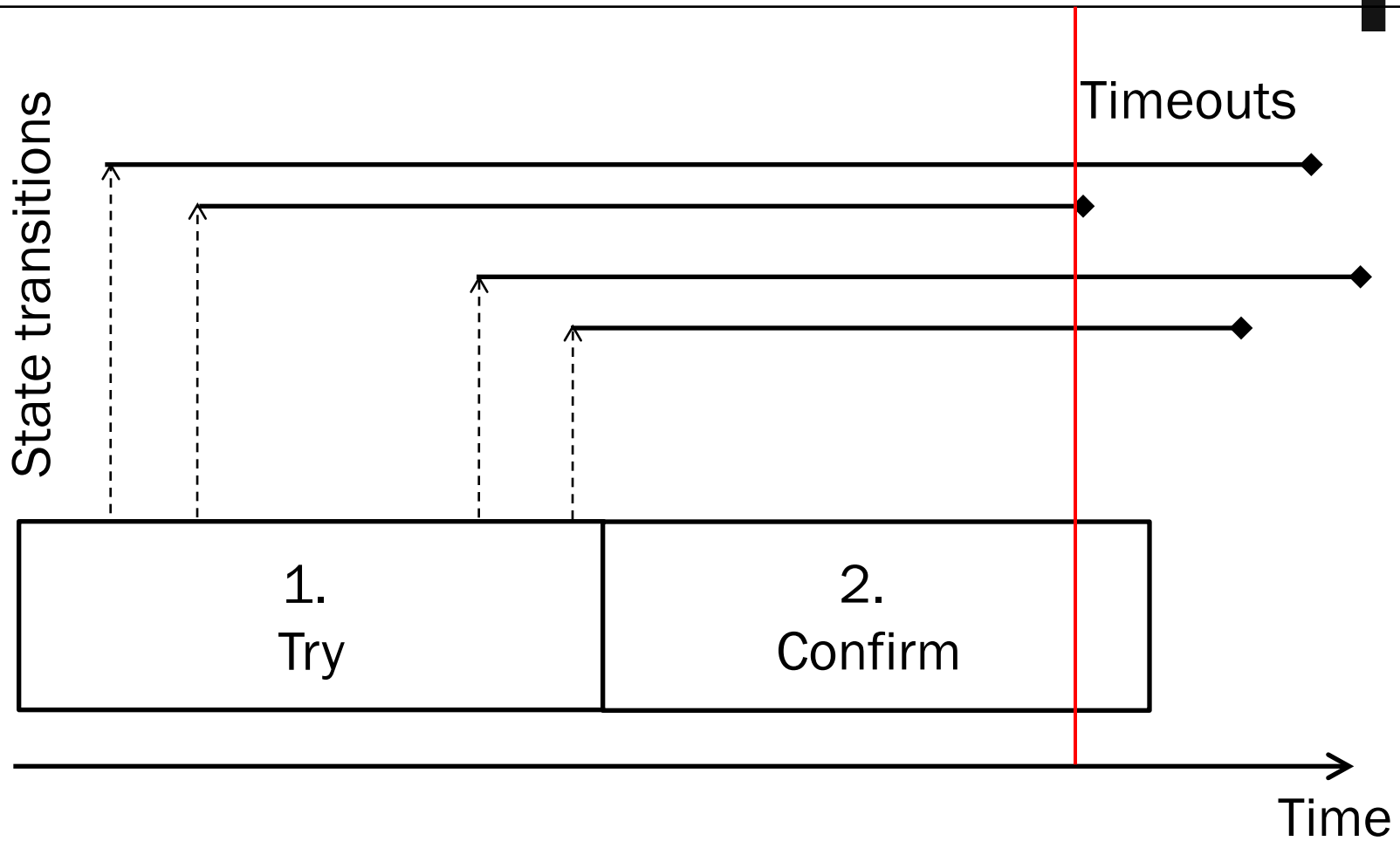
If something fails, retry as many times as necessary

A matter of timing



Agreement is reached if the confirmation phase ends before the resources undo the state transitions because of the timeouts

A matter of timing

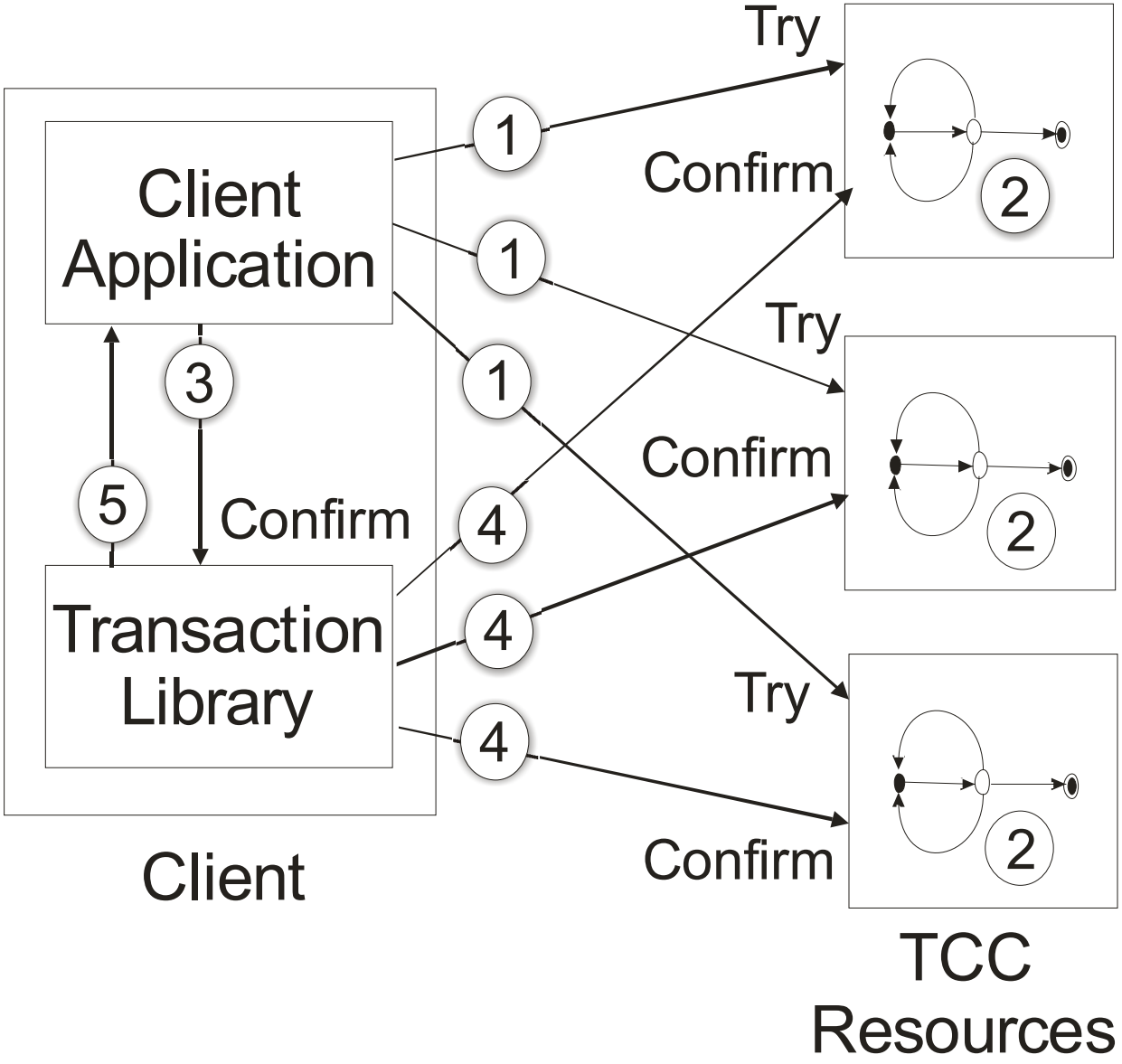


If the confirmation runs longer than the earliest timeout we cannot guarantee agreement

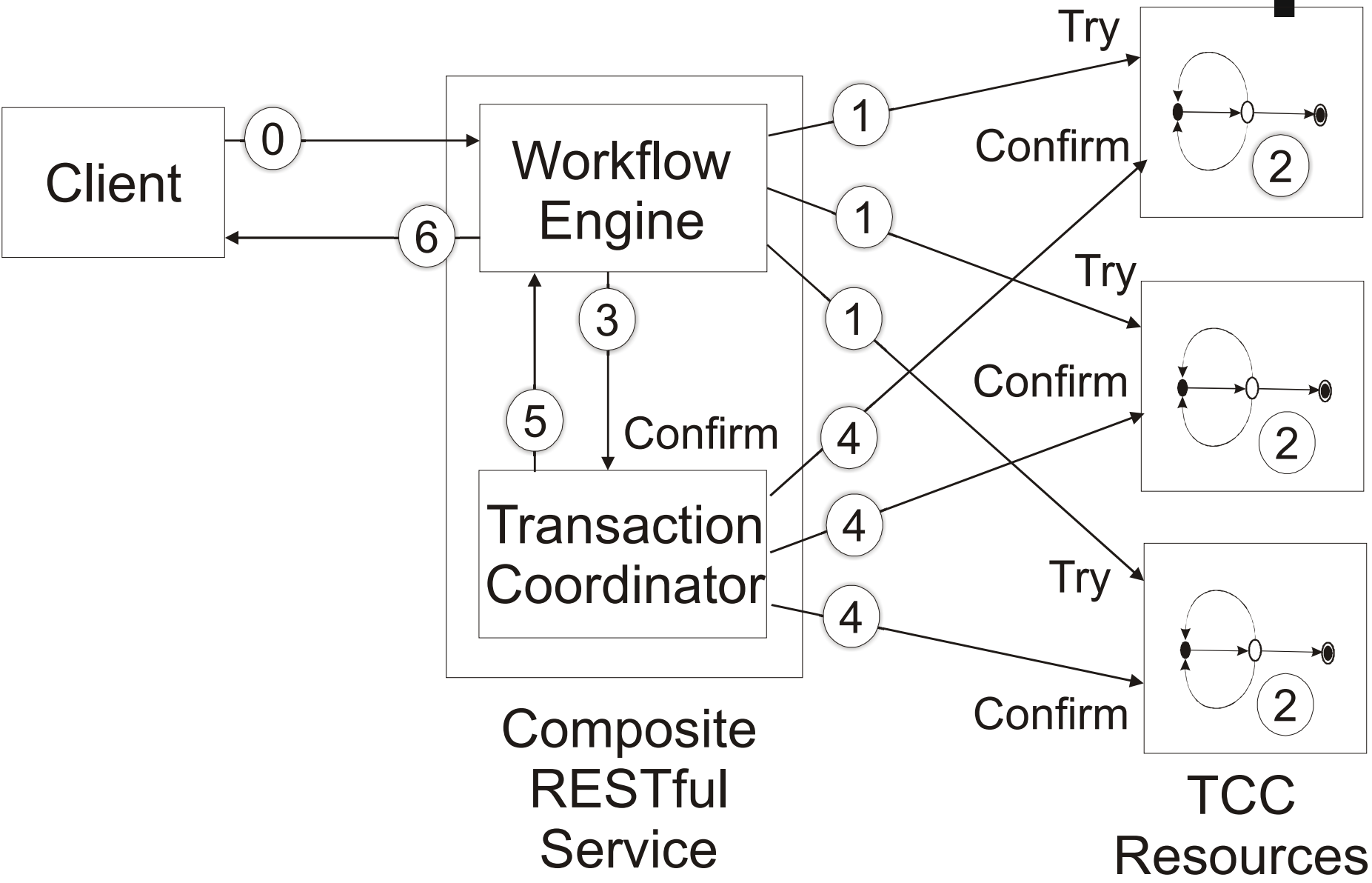
Timeouts and heuristics

- Bad News: As with every distributed agreement protocol, it is impossible to avoid heuristics
- Good News:
  - thanks to the REST uniform interface we can always do a GET on the URI of the reserved resource to see how much time we have left before it cancels
  - Avoid starting phase 2 if there is not enough time left to confirm with every service
- More complex preparation: if the resource allows it, extend the reservation time (also idempotent) before starting phase 2.
- In any case, use a lightweight transaction coordinator to log everything for recovery and human diagnosis of heuristics

Architecture (Client-side Transaction)



Architecture (Service Composition)



Conclusion

- The protocol guarantees **atomicity** in the event of failures among multiple interactions with RESTful Web services that comply with the TCC pattern.
- (We are not interested in isolation)
- No HTTP extension is required
- Fits very nicely with the REST uniform interface and the idempotency of the PUT/DELETE methods
- Hypermedia can be easily built in to guide the discovery of the cancellation/confirmation URIs (e.g., with HTTP Link Headers)

References

- G. Pardon, C. Pautasso, **Towards Distributed Atomic Transactions over RESTful Services**, in: REST: from Research to Practice, Springer (to appear)
- Cesare Pautasso, RESTful Web Service Composition with JOpera, Proc. Of the International Conference on Software Composition (SC 2009), Zurich, Switzerland, July 2009.

WS-REST 2011

ws://rest.2011

Second International Workshop on RESTful Design

http://ws-rest.org/2011

- @WWW2011, Hyderabad, India
- 28 March 2011
- Paper Submission is open (deadline: 31 Jan 2011)


ECOWS`11
Università della Svizzera italiana
9th IEEE European Conference on Web Services
(ECOWS 2011)
Lugano, Switzerland
September 14-16, 2011
http://ecows2011.inf.usi.ch
http://twitter.com/ecows2011
Università della Svizzera italiana
Faculty of Informatics