A Formal Definition of RESTful Semantic Web Services

ws://rest
2010

VNiVERSiDAD
D SALAMANCA

Antonio Garrote Hernández
María N. Moreno García

Outline

- Motivation
- Resources and Triple Spaces
- Resources and Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

Motivation

- Web development is becoming increasingly API-centric:
  - Rich Internet Applications
  - Mobile platforms
  - Mash-ups, OAuth, Microformats

Motivation

- Open issues in current web development:
  - Description of data: graph of social objects
  - Data interoperability
  - ...

Motivation

- Semantic Web technologies have the potential to solve these issues:
  - Open world semantics, monotonic reasoning
  - Description vocabulary (OWL, RDFS)
  - Data model (RDF, RDFa), query (SPARQL)...

Problem

- Semantic web adoption in regular web development is almost non existent
    - A pragmatic approach for levering semantic technologies is required
    - RESTful web services can serve as the foundation for such an approach

Proposal

- A model for RESTful semantic distributed computation
- A formal definition of the model

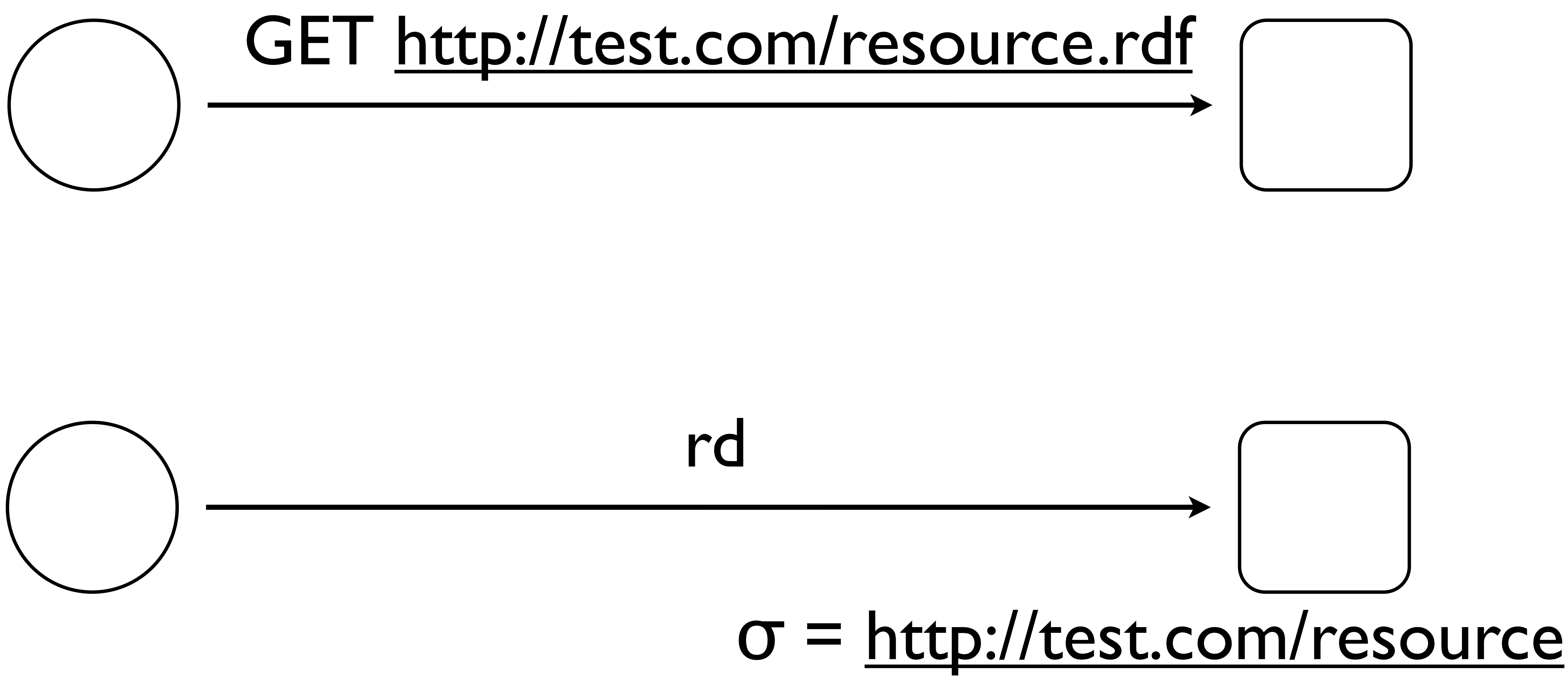
Outline

- Motivation
- Resources as Triple Spaces
- Resources and Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

Resources as Triple Spaces

- Generative Communication (Linda), TS computing [Gelernter, Fensel]
- **Resource**: Triple encoded graph stored in shared memory known as **triple space**
- Triple space operations:
  - Data manipulation
  - coordination primitives

Resources and Triple Spaces



Resources and Triple Spaces

- Extended triple space operations:
  - atomic *swap* operation modeling PUT requests
  - *notify* operation as an additional coordination primitive

Resources and Triple Spaces

- Formalization:

$$
\begin{aligned}
P &::= 0 \mid T \mid P|P \mid !P \mid if\, T\,?\,P.P \mid x ::= T \\
T &::= rd(\theta_i, p) \mid in(\theta_i, p) \mid out(\theta_i, v) \mid swap(\theta_i, p, v) \mid \\
&\quad\; rdb(\theta_i, p) \mid inb(\theta_i, p) \mid notify(\theta_i, \rho, v)
\end{aligned}
$$

$$
\begin{aligned}
\theta &::= \{\text{ triple spaces }\} \\
\rho &::= \{in, out\} \\
\mu &::= \{\text{URIs}\} \\
\lambda &::= \{\text{literals}\} \\
p &::= \{\text{patterns}\} \\
v &::= \{\text{values}\} = \{\mu\} \cup \{\lambda\} \cup < p, v > \cup < p, \theta_i >
\end{aligned}
$$

Resources and Triple Spaces

- Formalization:

$$(1)\quad \frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

$$(2)\quad \frac{P \rightarrow P'}{Q \rightarrow Q'}\, if\, P \equiv Q \; and \; P' \equiv Q'$$

$$(3)\quad !P.Q \rightarrow Q \mid P$$

$$(4)\quad \frac{rd(\theta_i,p).P}{rd(\theta_i,p).P \xrightarrow{<p,\theta_i>} P}$$

$$(5)\quad \frac{in(\theta_i,p).P}{rd(\theta_i,p).P \xrightarrow{<p,\theta_i>} P, \theta_i = \theta_i - <p,\theta_i>}$$

$$(6)\quad \frac{out(\theta_i,v).P}{out(\theta_i,v).P \xrightarrow{\overline{v}} P, \theta_i = \theta_i \cup v}$$

$$(7)\quad \frac{swap(\theta_i,p,v).P}{swap(\theta_i,p,v).P \xrightarrow{<p,\theta_i>,\overline{v}} P, \theta_i = \theta_i - <p,\theta_i> \cup v}$$

$$(8)\quad \frac{out(\theta_i,v).Q}{notify(\theta_i,out,p).P|out(\theta_i,v).Q \xrightarrow{\overline{v},<p,v>} P|Q}$$

$$(9)\quad \frac{in(\theta_i,p).Q}{notify(\theta_i,in,q).P|in(\theta_i,p).Q \xrightarrow{<p,v>,<q,<p,v>>} P|Q}$$

$$(10)\quad \frac{if\ T\ P.Q}{if\ T\ P.Q \xrightarrow{\overline{0}} Q}, \quad \frac{if\ T\ P.Q}{if\ T\ P.Q \xrightarrow{\overline{v}} P}$$

Resources and Triple Spaces

- TS useful for describing RESTful web resources as data manipulation
- Problems:
  - Creation, destruction of triple spaces (POST, DELETE operations)
  - Creation of new names
  - Blocking triple space operations

Outline

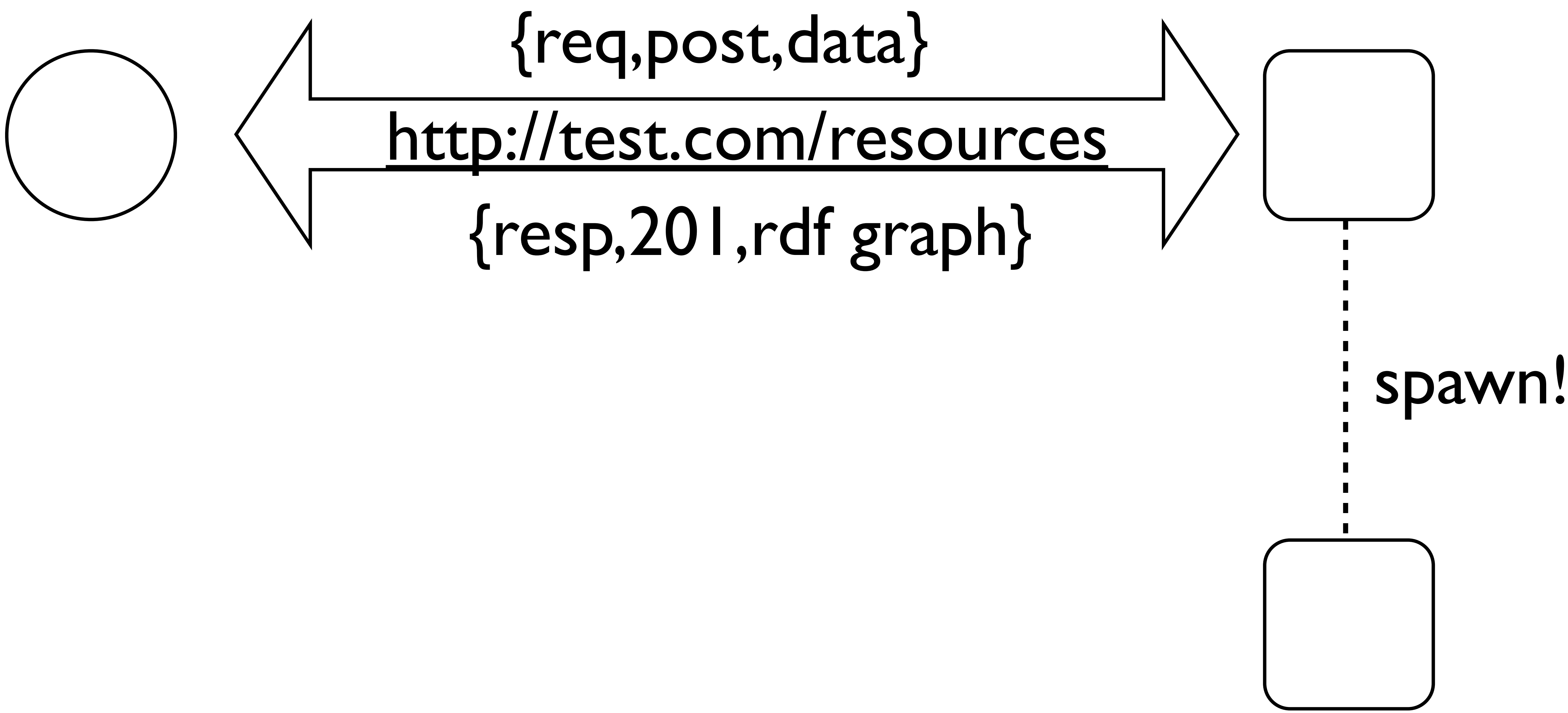
- Motivation
- Resources as Triple Spaces
- Resources as Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

Resources and Processes

- Pi-Calculus [Milner]
- HTTP request can be modeled as a message sent from a process (client) to another process (resource)
- URI can be modeled as a channel between processes
- The URIs inside the RDF graph returned as a response grants the process access to new communication channels

Resources and Processes

Resources and Processes



Resources and Processes

- Formalization:

$$
\begin{array}{rcl}
P & ::= & 0 \mid T \mid M \mid P|P \mid !P \mid if\, T\,?\,P.P \mid x ::= T \mid \\
 & & new\, \mu\, in\, P \\
M & ::= & \overline{req(\mu)}[m,p,v] \mid [m,p,v]req(\mu) \mid \overline{resp(\mu)}[c,v] \mid \\
 & & [c,v]resp(\mu) \\
 & & \\
m & ::= & \{get, post, put, delete\} \\
c & ::= & \{200, 201, 404, 401\}
\end{array}
$$

Resources and Processes

- Formalization:

$$(11) \quad \frac{P \xrightarrow{\overline{req(\mu)[m,p,v]}} P', Q \xrightarrow{[m,p,v]req(\mu)} Q'}{P|Q \rightarrow P'|Q'}$$

$$(12) \quad \frac{P \xrightarrow{\overline{resp(\mu)[c,v]}} P', Q \xrightarrow{[c,v]resp(\mu)} Q'}{P|Q \rightarrow P'|Q'}$$

$$(13) \quad \frac{\overline{req(\mu)[m,p,v]}.P}{\overline{req(\mu)[m,p,v]}.P \xrightarrow{*} [c,v]resp(\mu).Q}$$

$$(14) \quad \frac{[m,p,v]req(\mu).P}{[m,p,v]req(\mu).P \xrightarrow{*} \overline{resp(\mu)[c,v]}.Q}$$

Resources and Processes

uri1 ::= http://test.com/resources
uri2 ::= http://test.com/resources/test

$$Ag ::= \overline{req(uri1)[post,0,Data]}.$$
$$[201, \{Uri2,type,resource\}]resp(uri1).$$
$$\overline{req(Uri2),[get,*,0]}.\ [200,Data]resp(Uri2)$$

$$Res ::= new\ uri2\ in\ ([post,0,Data]req(uri1).Res2(uri2,Data)!.$$
$$\overline{resp(uri1)[401,uri2]}).Res$$

$$Res2(uri2, D) ::=[post,*,0]req(uri2).\overline{resp(uri2)[200,D]}.Res2$$

Resources and Processes

- Message passing, channels and processes useful for modeling dynamic aspects of HTTP computations
- Problems:
  - Modeling the state of the resource is less intuitive

Outline

- Motivation
- Resources as Triple Spaces
- Resources as Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

RESTful semantic resources

- Triple spaces and Pi-calculus are complementary formalisms
- Combination of both formalisms for describing RESTful distributed computation

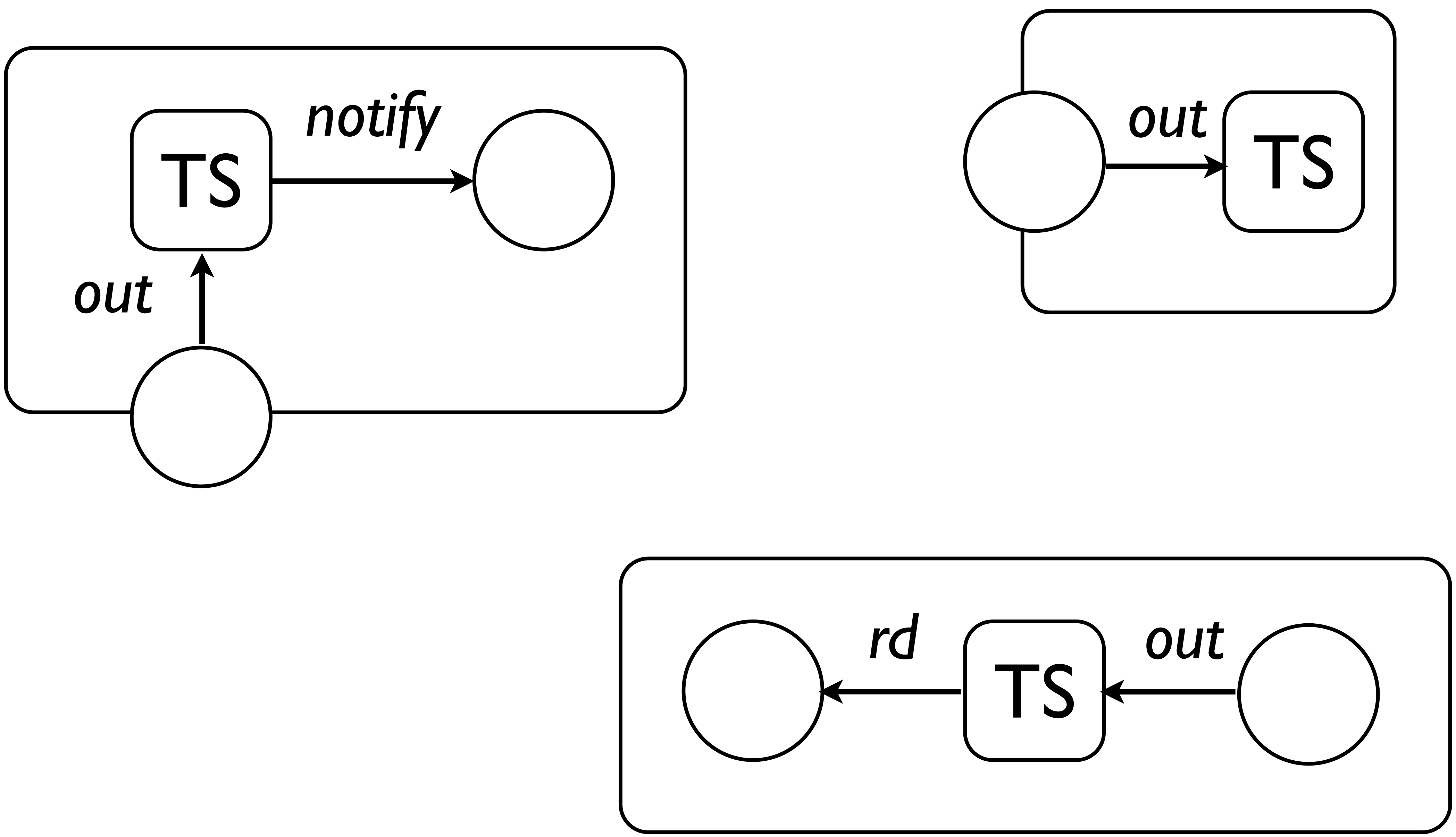
RESTful semantic resources

- Computation takes place in certain "computational places"
  - Nodes executing a web services API
  - A web browser
  - A mobile phone

RESTful semantic resources

- Inside each “computational place” a set of processes are executed and a certain number of triple spaces are shared
- Communication between processes inside a computational place is tuple space based

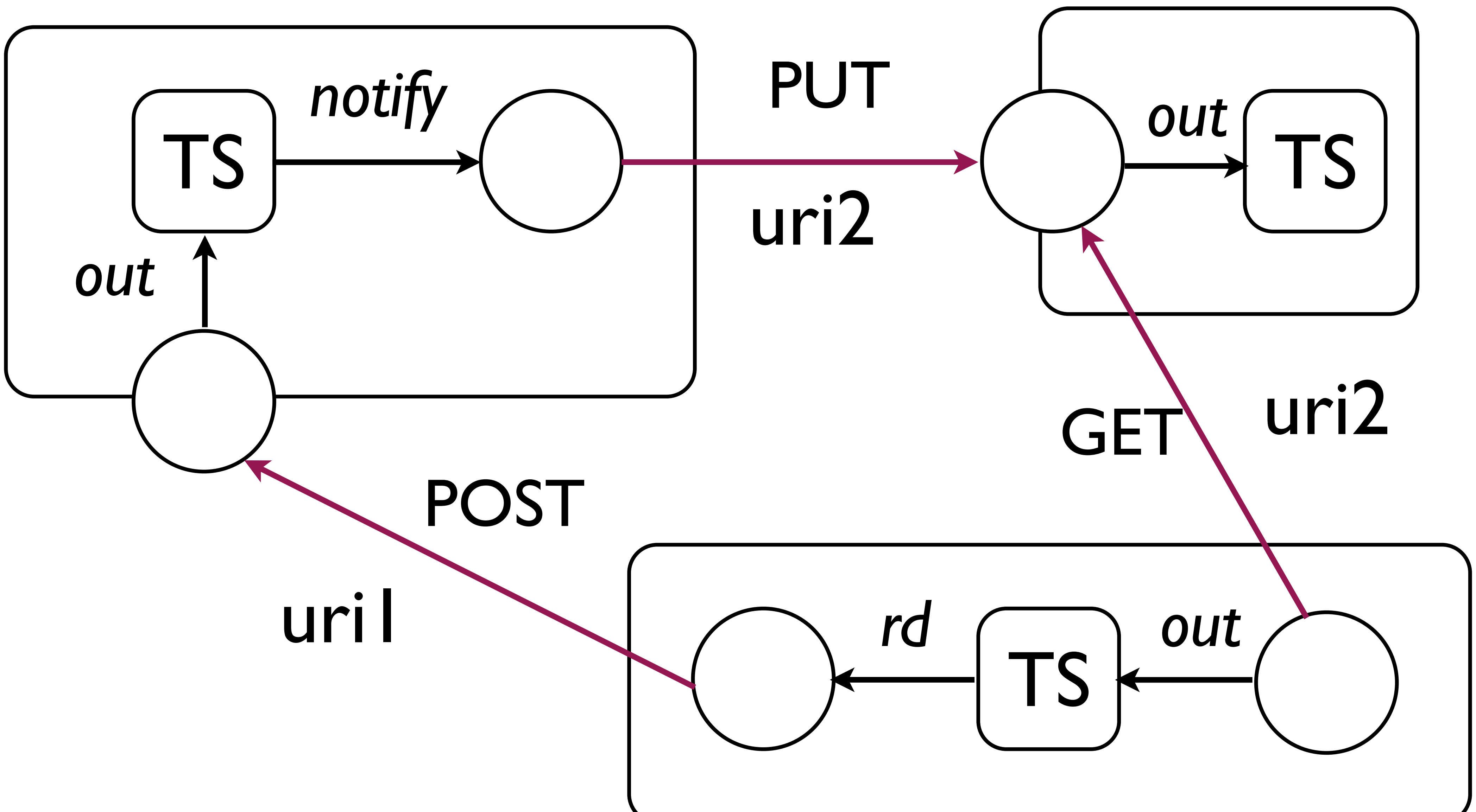
RESTful semantic resources



RESTful semantic resources

- Certain processes in “computational places” have an associated URI and process incoming messages according to REST semantics
- Communication between “computational places” is message passing based
- URIs can be transfered between triple spaces in different “computational places”

RESTful semantic resources



RESTful semantic resources

- RESTful semantic web resource:
  - Process being executed in a computational place
  - Associated URI
  - Receives HTTP messages through URI
  - Manipulates TS according to REST semantics

RESTful semantic resources

- Formalization:

$$
\begin{aligned}
R_{REST}(\theta,\mu) \quad ::= \quad & [m,v,p]req(\mu).\, if\, m = get\,?\, R_{get}(\theta,\mu). \\
& if\, m = post\,?\, R_{post}(\theta,\mu). \\
& if\, m = put\,?\, R_{put}(\theta,\mu). \\
& if\, m = delete\,?\, R_{delete}(\theta,\mu). \\
& \overline{resp(\mu)}[406,0].R_{REST}(\theta,\mu)
\end{aligned}
$$

$$
R_{get}(\theta,\mu) \quad ::= \quad x ::= rd(\theta,p).\overline{resp(\mu)}[200,x].R_{REST}
$$

$$
\begin{aligned}
R_{post}(\theta,\mu) \quad ::= \quad & new\,\nu\, in\, out(\theta, < p,\nu >).!R(\theta,\nu). \\
& \overline{resp(\mu)}[201, < p,\nu >].R_{REST}
\end{aligned}
$$

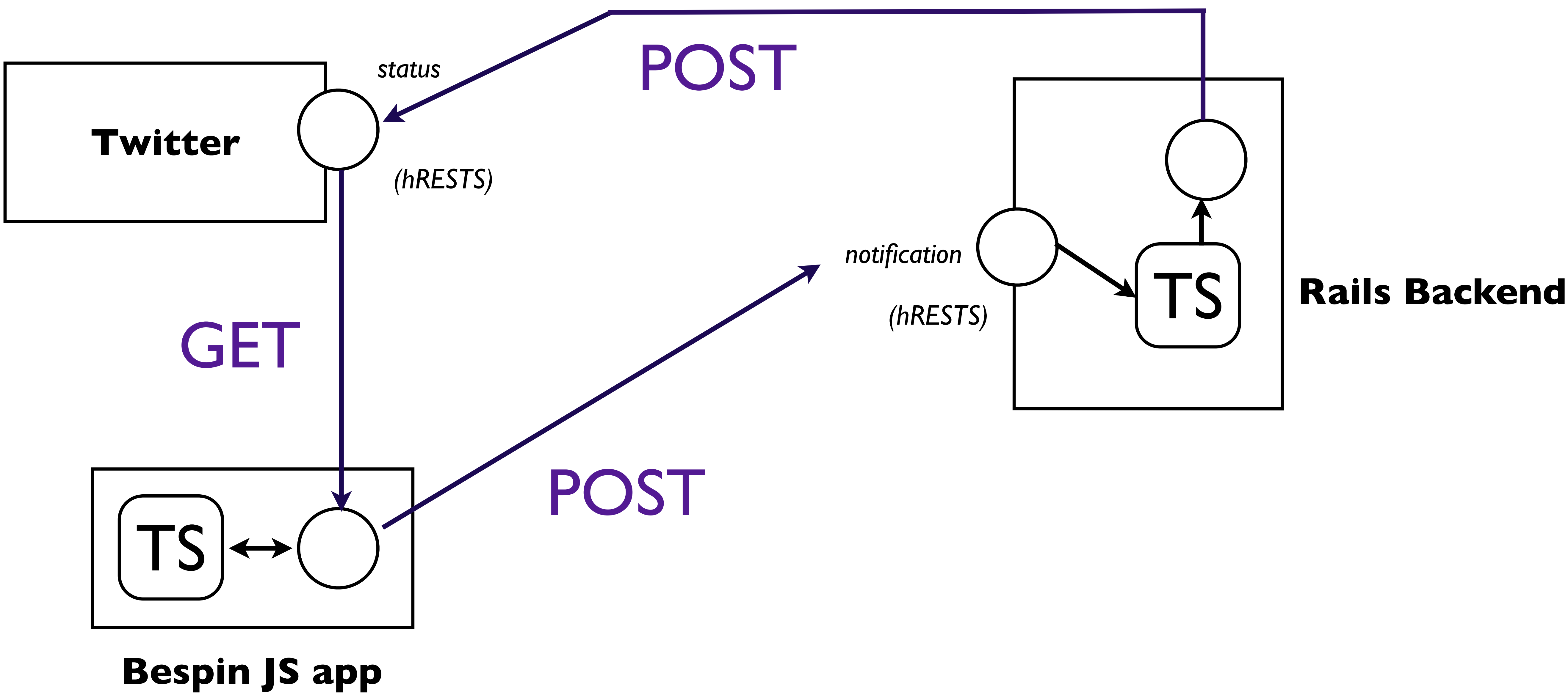
$$
R_{put}(\theta,\mu) \quad ::= \quad swap(\theta,p,v).\overline{resp(\mu)}[200,v].R_{REST}
$$

$$
R_{delete}(\theta,\mu) \quad ::= \quad in(\theta,p_\mu).\overline{resp(\mu)}[200,0].0
$$

Outline

- Motivation
- Resources and Triple Spaces
- Resources and Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

Semantic Bespin



http://github.com/antoniogarrote/semantic_rest

Outline

- Motivation
- Resources and Triple Spaces
- Resources and Processes
- RESTful Semantic Resources
- Example
- Conclusions and Future Work

Conclusions

- Clear definition of data related and process related aspects of RESTful computations
- Introduces the notion of computational place as an aggregation of resources or processes with RESTful interfaces
- Modeling decoupled from actual implementation

Conclusions

- Benefits:
  - Composition of services trivially modeled as a sequence of messages in the calculus
  - It is possible to model complex interaction scenarios triggering blocking TS operations (notify, rdb, outb) as a side effect of a HTTP message

Conclusions

- Benefits:
  - Use of semantic metadata offers an uniform model for data shared among resources
  - Shared operations for querying and manipulating resources
  - Incremental description of resources

Future work

- Blocking operations
  - blocking communication primitives useful for coordination between agents and resources
  - avoid polling
  - restricted to triple space operations
  - extension to the HTTP interface

Future work

- Type system
  - Types can be assigned to resources based on the ontology primitives used in the description of the resource
  - OWL, RDFS, RDF entailment regimes
  - Importance for the discovery of resources

Future work

- Implementation
  - Experimental implementation with blocking operations
  - hRESTS, RabbitMQ, OpenSesame, Erlang OTP
  - http://github.com/antoniogarrote/Plaza