REST Design Patterns for Robust Asynchronous Notification

Using simple observe/notify to build
a robust and reusable design pattern
for asynchronous notifications

Problems

- Observe is not a well managed relationship
  - The list of observers is hidden server state
  - Client can't be certain if it is still in the list
  - Conditional Observe is difficult to manage
- Events have life cycle beyond one notification
  - Alerts are generated, acknowledged, and eventually cleared
  - Use cases for asynchronous Event delivery, polling, and batch Event processing

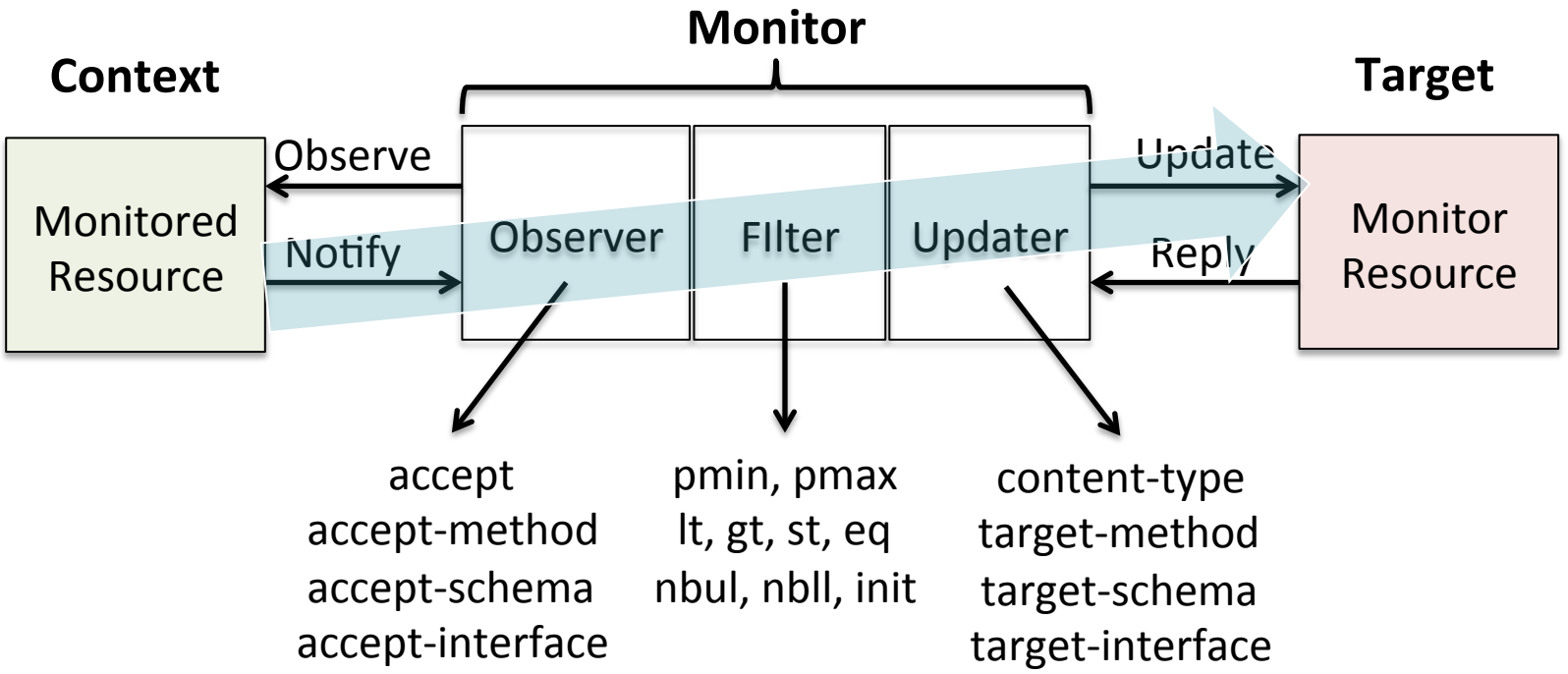
Design Patterns

- Monitor
  - Create a managed Observe relationship using a REST resource with a defined link relation and parameter set
- Events
  - REST resource to represent an Event instance
  - Maintain Event instances in an observable collection

Monitor

- Use the IANA registered "monitor" link relation
  - *Description: Refers to a resource that can be used to monitor changes in an HTTP resource* (RFC5989)
  - Similar to "boundto" (dynlink) but defines a unidirectional state update from context to target.
- A Monitor may use Observe on the server to obtain state changes of the context resource
- A Monitor may implement conditional notification using filter parameters (dynlink) as well as defining transfer methods and formats
- A Monitor may support multiple source and target protocols based on URI scheme (mqtt, coap, http)
- Monitor parameters may be encoded as link attributes or as properties of a monitor configuration resource

Monitor


Monitor
Context
Target
Observe
Update
Monitored Resource
Notify
Observer
FIlter
Updater
Reply
Monitor Resource
accept
accept-method
accept-schema
accept-interface
pmin, pmax
lt, gt, st, eq
nbul, nbll, init
content-type
target-method
target-schema
target-interface

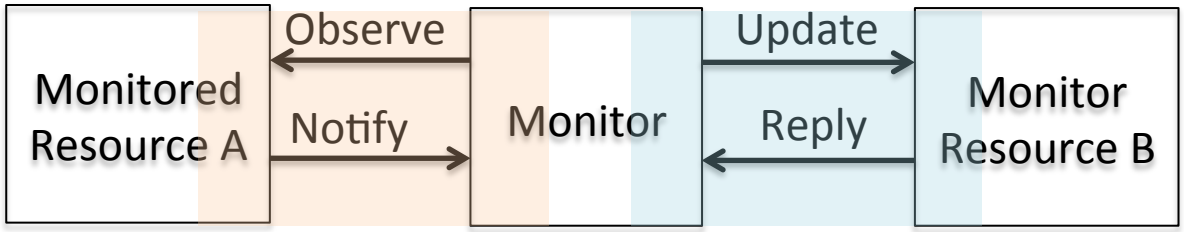
Monitor Patterns

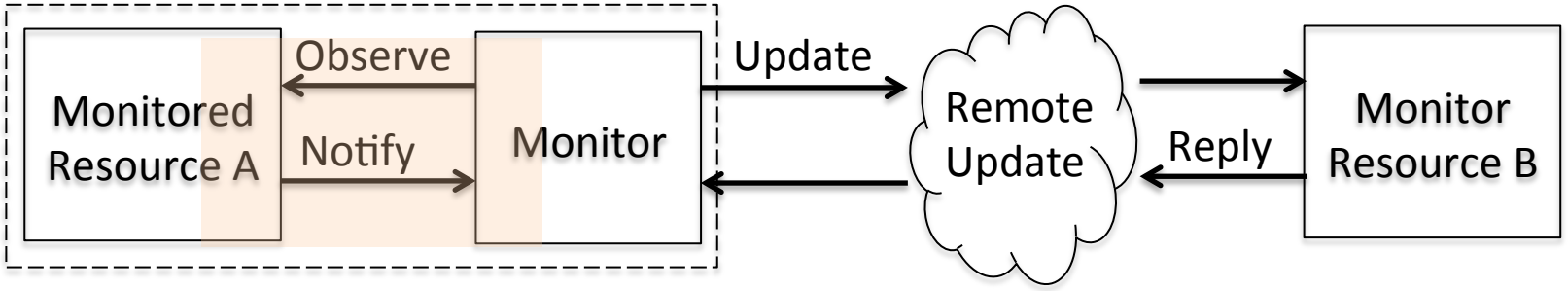
A Pushes State To B

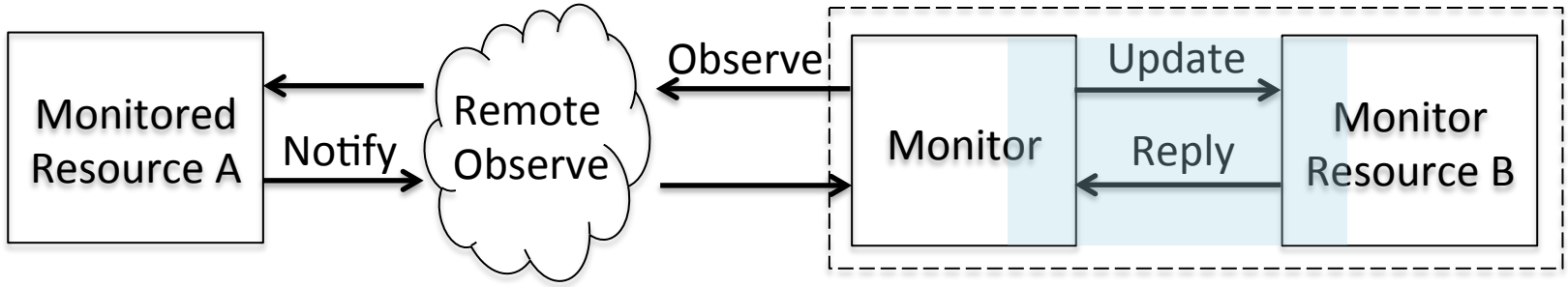
`<B>;rel=monitor`

B Observes State From A

`<>;anchor=A;rel=monitor`







Monitor Link Examples

Update a monitor resource when context is updated

```
  "rel": "monitor",
  "href": "monitor"
}
```

Update the context when a remote resource is updated

```
{
  "anchor": "coap://0m2m.net:5683/example/test",
  "rel": "monitor",
  "href": ""
}
```

Monitor Link Examples

Subscribe to an MQTT topic and update a resource

```
{
  "anchor": "mqtt://0m2m.net/example/topic",
  "rel": "monitor",
  "href": "updated-on-mqtt-notify"
}
```

Publish updates on a resource to an MQTT topic

```
{
  "anchor": "publish-updates-to-mqtt",
  "rel": "monitor",
  "href": "mqtt://0m2m.net/example/topic"
}
```

Events

- State changes that require more than simple notification may be handled as Events
- Events may have a life cycle, like log records, alerts, etc.
- A monitor may add state change notifications to a collection of Event instances using CREATE
- The Event collection is Observable and transmits newly created Event instances as notifications

Monitor Link to Event Collection

```
Create new event instances when events occur
{
  "anchor": "/example/resource/event-emitter",
  "rel": "monitor",
  "href": "events",
  "target-method": "create"
}

Push event notifications to a MQTT topic
{
  "anchor": "events",
  "rel": "monitor",
  "href": "mqtt://0m2m.net/example/topic"
{
```