Would you architect the control API of the next-gen Mars rover to be RESTful instead of an RPC?

Asked 16 years, 2 months ago Modified 1 year, 5 months ago

Viewed 711 times

6

Forgive me if this verges on being a "discussion" question, but I really would appreciate a yes/no answer, with an appropriate explanation.

Suppose you have to design and implement a control API for a robot, say the next generation Mars Rover. Do you architect this API according to RESTful principles, or do you use a classic RPC, such as XMLRPC?

I ask this because I have to do something similar, though the "robot" is a collection of virtual machines. I'm being urged by one rather persuasive engineer, a well known REST advocate, to make the API RESTful. I've never used REST principles, and I'm struggling to see how they fit in designing low-level inter-process APIs. REST seems infused with the theme of interacting with a modifiable data repository, usually many hops away. What I'm trying to do feels more like closely controlling a robot. I can see how one could argue that the robot is, in the abstract, just a data repository -- "PUT left turn", "PUT travel 100 meters", "GET outside temperature". But this seems to be

a rather contrived model. I certainly will receive no benefit from caching or a proxy ("Hello, JPL? This is the Akamai co-lo in Canberra. We're taking over the Rover now, ok?")

So, is a RESTful architecture useful here? Is it still superior to RPC even when the interaction is so narrowly focused?

rest ipc rpc robotics

Share

Improve this question

Follow

edited Jul 1, 2023 at 9:28

Brian Tompsett - 汤莱恩

5,875 ● 72 ● 61 ● 133

asked Oct 9, 2008 at 2:41

Kevin Little

12.9k ● 6 ● 42 ● 48

XMLRPC is designed exactly for this. None of the REST verbs make sense in this context, all your doing is calling remote procedure calls. Additionally, other REST benefits (such as implied catching) don't make any sense here either. – FlySwat Oct 9, 2008 at 2:58

2 Answers

Sorted by: Highest score (default)

7

I think REST would make more sense than traditional RPC. Even the Micorosft Robotics Studio runtime application model uses REST.



The robot can consist of different resources that are identified by URI, including one for each sensor and actuator or composite abstractions thereof.

REST puts emphasis on guaranteeing what the side effects of certain methods are and also it facilitates caching, both of which can be useful for something like controlling and monitoring a distant robot. And just because you can use REST it **doesn't** have to be the HTTP protocol.

A SAFE and IDEMPOTENT method like GET, though, is good for tracking the state of the robot and polling its sensor data. You can use something like the Last-Modified header to retrieve cached sensor data that doesn't change often (e.g., humidity or light levels).

For long distances you can use relay proxies for caching.

For commands that move the robot, something like POST would be used where every such message will change the robot (e.g., turn right). A status code can be returned indicating whether the command was immediately executed or queued for processing.

The absolute state of any resources can be set using something like PUT where multiple messages will not change things any more than just a single message (e.g., point to north pole or dim front lights to 10% brightness). This allows for reliable messaging in case of probablity of messages getting lost en route.

A new resource may be created via a POST-like operation as well, e.g., a data-collection routine and a set of parameters. The POST request can send back a CREATED result with a URI for the new resource, which can be used to DELETE when no longer needed.

A group of robots may also speak to each other using the same REST based protocol and can enjoy the same benefits.

Granted, for something simple like one person controlling a single isolated local robot, a REST API may be overkill. But for multi-user and/or unreliable-communications-channels and/or Web-scale-networking, REST is something to consider.

Share Improve this answer Follow

edited Oct 9, 2008 at 3:25

answered Oct 9, 2008 at 2:57

Mark Cidade

99.8k • 33 • 229 • 237

1

REST principles ensure that your application scales well, and plays well with intermediaries across the internet, (proxies, caching, etc). If your "virtual machine" network is large scale then a RESTful architecture could be advantageous. If you are building a small-scale network, then REST would not be as compelling.

Share Improve this answer

Follow

answered Oct 9, 2008 at 3:01

Chris Noe

37.1k ● 22 ● 74 ● 92