MultiDasher

MultiDasher Chain Handling

Joining chains and providing a chain registry

Keir Finlow-Bates

23 October 2018

One of the pressing problems with deploying MultiChain blockchains is that a chain administrator has to admit further blockchain participants, even if the chain parameters are set to allow anyone to connect. This is because an address cannot be used until it has received a transaction (although even an empty transaction, such as a transfer of a zero amount of an asset will do). The admission action from the administrator is often manual. This paper provides a specification for MultiDasher to allow the chain joining activity to be automated.

Furthermore, the specification allows the automated approval of new participants to be contingent on the participant providing a set of information, for example: email address, name, location or other data required by the administrator before approval. In the future this can be extended to include "respond to this email to activate your account"-type functionality.

A second problem is that finding a specific chain requires knowledge of a chain peer location. In the case of permissioned blockchains the participants are usually readily identifiable and the problem can be overcome by, for example, publishing connection details on a website or through an email. However, in a public-yet-permissioned blockchain such as those enable by MultiDasher, a method for registering blockchains on a public "bulletin-board" is desirable.

The following paper provides further insight into these problems and proposes a design solution for MultiDasher.

1 Introduction

1.1 Automating chain connections

MultiChain chains contain the following setting in the blockchain launch parameters, set to `false` by default (it cannot be changed after the chain is instantiated):

```
anyone-can-connect = false
```

With this setting, someone hoping to join the blockchain has to explicitly be granted permission to connect by a chain administrator. If the `anyone-can-connect` parameter is set to `true`, although participants cannot actively participate on the blockchain, because they are unable to send valid transactions, they are able to read the publicly available contents of the blockchain.

By default, MultiChain blockchains contain a publicly readable data stream called `root`. Our solution at MultiDasher is for the chain admin to publish a sign-up message on the chain in the root stream. A participant's MultiDasher then searches the blockchain for the latest valid sign-up message, and extracts it from the `root` stream.

The sign-up message contains enough information for the participant's MultiDasher to generate a form, which the participant can fill in and POST back to the chain admin. Either the admin's MultiDasher can validate the form response and automatically admit the participant by sending a zero value transaction, or the request can be placed in the admin's inbox for manual approval.

The first section of this document provides a specification for the sign-up messages, and example implementations for Drupal developers and administrators.

1.2 A public chain registry

Although MultiDasher users are free to launch private permissioned blockchains as they please, the second part of this document provides a description of the mechanism in MultiDasher for announcing chains on the *Frogchain* blockchain that is listed in every standard installation of MultiDasher.

New chains have the option of announcing a chain creation message on *Frogchain*, and MultiDasher can provide users with the option of signing up to *Frogchain* and searching it for other chains. Announcement messages can subsequently be canceled by the original announcer, or by *Frogchain* administrators if they detect that the chain is no longer active. Note that the announcement messages are never deleted (the tamper-proof feature of blockchains), so even if a message is subsequently canceled the only effect is that the MultiDasher software will no longer list it in the active chains view. Blockchain users are still welcome to search the `registry` stream of *Frogchain* in order to find all previously announced chains themselves.

2 Chain Connections

Chain connections handled by MultiDasher servers require the chain to be open for anyone to connect, that is, they require the parameter setting `anyone-can-connect = true` in the `params.dat` file. This is the default setting for MultiDasher generated blockchains.

2.1 Sign-up messages specification

The first MultiDasher instance creating a chain becomes the admin for that chain, and will write a sign-up message to the `root` stream of the chain shortly after its creation. This registers the admin's address as the only legitimate entity to subsequently submit further edits to the sign-up message, should its content need changing – for example if the chain-administrating MultiDasher instance is moved to a different domain, or if it is decided later on that more or less sign-up data is required.

The MultiChain API command for registering a sign-up form and location (URL) is by writing a sign-up object to the `root` stream of the chain, because when connecting to a new chain, MultiDasher searches for the first sign-up object in the `root` stream, retrieves the publisher address, and then searches for the last sign-up object published by the same publisher address, and uses it to construct a sign-up form.

So the MultiChain API command for publishing the sign-up message is:

```
publish root sign-up '<form object>'
```

The `form object` can have many structures, but the `chainAddress` field, the `resourceUrl` and the `mandatory` array are mandatory, as they are used to determine the address of the chain, receive the blockchain address of participants signing up to the chain, and ensure that mandatory fields are filled in. Here is an example:

```
{
  "json":{
    "chainAddress":
        "<participants put their address here>",
    "resourceUrl":"<POST location>",
    "email":"Enter your email address here",
    "firstName":"Enter first name here",
    "lastName":"Enter last name here",
    "mandatory":["chainAddress",
                 "email"]
  }
}
```

The `mandatory` array is used to determine what data any participant signing up must provide. Of course there is no way of automatically checking that what they typed in is actually true.

In this chain the participants must provide an email address in order to sign up. Their chain address should be automatically provided to the admin MultiDasher by their MultiDasher.

2.2 Sign-up by participant

The participant connects to the chain, and retrieves the latest valid sign-up message. The admin may have entered anything into the `chainAddress` field (typically they will put `<chainname>@<address>:<port>`, but the participant's MultiDasher must automatically put the participant's chain address in there.

The resource URL indicates the location for a web server resource on the admin's server that can receive a posted form in the format that the sign-up message specifies.

2.3 Sign-up messages update

If the sign-up message needs to be updated, for example because the chain administrator node has moved, simply submit another sign-up message to the chain with the data corrected. The sign-up message must be re-submitted by the same publisher that published the original sign-up message, or participant MultiDasher instances will reject it as invalid (this is to prevent anyone else from hijacking your chain). The updated sign-up form can be published with the same `publish root sign-up '<form object>'` command.

For example, here is a re-submission of the above sign-up message to make the first name compulsary, and to add a country of origin field:

```
{
  "json":{
    "chainAddress":
        "<participants put their address here>",
    "resourceUrl":"<POST location>",
    "email":"Enter your email address here",
    "firstName":"Enter first name here",
    "lastName":"Enter last name here",
    "country":"Enter country of residence",
    "mandatory":["chainAddress",
                 "email",
                 "country"]
  }
}
```

2.4 Sign-up messages cancel

We do not see a need to cancel MultiDasher MultiChain sign-up messages.

2.5 Sign-up messages transfer

It may become necessary to grant the right to update transfer messages to another address.[TODO: add specification here]

2.6 Admin response

The admin MultiDasher is norally configured to response to a POST request with either `200 OK` if the posted data is validated, or with `403 Forbidden` if the form was incorrect (for example, if a mandatory field was left blank).

The admin MultiDasher then, if configured, will automatically execute a transaction to the provided address to activate it. For example, the *Frogchain* chain automatically sends 10 vote assets and 10 reputation assets to anyone signing up.

Finally, the participant MultiDasher receives the POST response, and either shows a success or a failure to sign up page.

3 Chain Registry

The *Frogchain* blockchain also has a public stream called `root`, which can be used by anyone to announce new blockchains.

3.1 Announce chain

The MultiChain API command for publishing a blockchain announcement on the registry stream (if you are active on `Frogchain` is as follows:

```
publish registry <chain-key> '<register object>'
```

The `chain-key` is simply the chainname, address and port, in the following format:

```
<chainname>@<address>:<port>
```

This key uniquely identifies the chain and where it is. The `<address:port>` can be in domain name format, or an IPv4 or IPv6 address.

The format of a JSON register object is:

```
{
  "json":{
    "chainname":"<chain-name>",
    "peer":"<address:port>",
    "fallback":[
      "<fallback-address:port>",
      "<fallback-address:port>"
    ],
    "description":"<max 90 characters>",
    "active": true
  }
}
```

All the fields are optional other than `active`. The chain-key and publisher are subsequently used to identify the unique new chain, and the publisher address should be used to identify the validity of future updates or deactivation (i.e. deregistration) of the chain.

3.2 Update chain

If the peer address or fallback addresses of the chain change, an update can be submitted to the registry. The update should only be considered valid if published by the original announcement publisher chain address.

```
publish registry <chain-key> '<update object>'
```

The format of a JSON update object is the same as the register object, with fields updated with the new data. Note that the process in MultiDasher is to retrieve the first item with key `chain-key` to get the origitnal publisher data from the registry stream, and then the last item with key `chain-key` published by the original publisher to get the most recent version.

3.3 Deregister chain

A chain can be removed from the registry (well, not really, as it's a tamper-proof ledger) by the original publishing address sending a delete message to the registry stream:

```
publish registry <chain-key> '<delete object>'
```

The format of a JSON delete object is:

```
{
  "json":{
    "active": false,
    "reason": "reason for removing chain"
  }
}
```

The `reason` field is optional.

4 What Next?

The specification above should provide functionality for automated chain admission, or administrator notification, along with a chain registry for MultiDasher.

If you're managed to read this far we hope you agree with us and are perhaps thinking about joining in. Why not start out by visiting the MultiDasher website and the Github repository, and install the software to try it out. The MultiDasher Wiki in particular has a lot of resources to get you started.

Once you've done that, please feel free to reach out to us, raise issues, or start improving the code with us.

5 About Chainfrog Oy

Chainfrog Oy was founded in 2016, and invents and researches blockchain technology. Visit our website or contact us at info@chainfrog.com for more information.