Server Discovery

Testing client behaviour

The client must be tested to follow a set of specifications in order to have the same behaviour compared to other drivers. Depending on the language this behaviour can be shown in events or return values. This report is following a specification from Mongodb website and is shown below.

Abstract

This spec defines how a MongoDB client discovers and monitors one or more servers. It covers monitoring a single server, a set of mongoses, or a replica set. How does the client determine what type of servers they are? How does it keep this information up to date? How does the client find an entire replica set from a seed list, and how does it respond to a stepdown, election, reconfiguration, or network error?

All drivers must answer these questions the same. Or, where platforms' limitations require differences among drivers, there must be as few answers as possible and each must be clearly explained in this spec. Even in cases where several answers seem equally good, drivers must agree on one way to do it.

MongoDB users and driver authors benefit from having one way to discover and monitor servers. Users can substantially understand their driver's behavior without inspecting its code or asking its author. Driver authors can avoid subtle mistakes when they take advantage of a design that has been well-considered, reviewed, and tested.

The server discovery and monitoring method is specified in four sections. First, a client is configured. Second, it begins monitoring by calling ismaster on all servers. (Multi-threaded and asynchronous monitoring is described first, then single-threaded monitoring.) Third, as ismaster calls are received the client parses them, and fourth, it updates its view of the topology.

Finally, this spec describes how drivers update their topology view in response to errors, and includes generous implementation notes for driver authors.

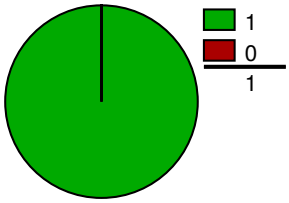
This spec does not describe how a client chooses a server for an operation; that is the domain of the Server Selection Spec. But there is a section describing the interaction between monitoring and server selection.

There is no discussion of driver architecture and data structures, nor is there any specification of a user-facing API. This spec is only concerned with the algorithm for monitoring the server topology.

```
use lib 't';
use Test-support;
use MongoDB;
use MongoDB::Client;
use MongoDB::Server;
my MongoDB::Test-support $ts .= new;
my Int $p1 = $ts.server-control.get-port-number('s1');
my Int $p2 = $ts.server-control.get-port-number('s2');
my MongoDB::Client $client .= new(:uri<mongodb://localhost:34567/>);
ok $client.defined, 'T0';
```

✔ **T0:** Returned client object is defined, even when the uri is not pointing to an existing mongod server.

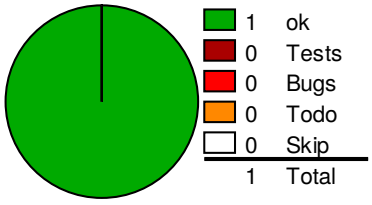
Normal tests



Bug issues

Todo tests

Summary



Generated using SemiXML, SxmlLib::Testing::*, XML, ©Google prettify