

Some abuse of UML here!

This demonstrates Seagull service architecture.

Services calls each other by name and the SGRouter resolves closest service to deliver to. SGRouter instances can have configured addresses to other instances but are also allowed more dynamic discovery where such is possible.

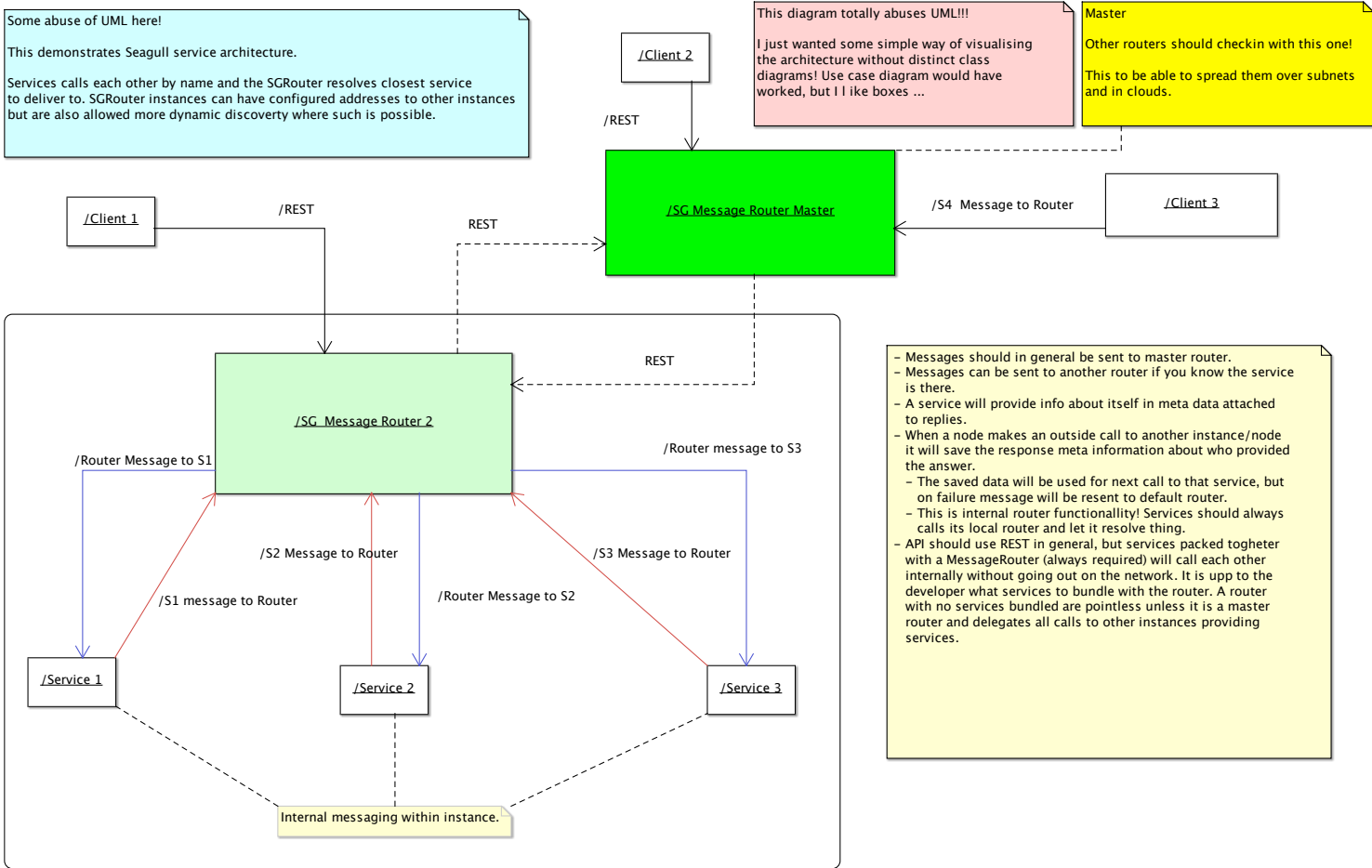
This diagram totally abuses UML!!!

I just wanted some simple way of visualising the architecture without distinct class diagrams! Use case diagram would have worked, but I like boxes ...

Master

Other routers should checkin with this one!

This to be able to spread them over subnets and in clouds.



- Messages should in general be sent to master router.
- Messages can be sent to another router if you know the service is there.
- A service will provide info about itself in meta data attached to replies.
- When a node makes an outside call to another instance/node it will save the response meta information about who provided the answer.
- The saved data will be used for next call to that service, but on failure message will be resent to default router.
- This is internal router functionality! Services should always call its local router and let it resolve thing.
- API should use REST in general, but services packed together with a MessageRouter (always required) will call each other internally without going out on the network. It is up to the developer what services to bundle with the router. A router with no services bundled are pointless unless it is a master router and delegates all calls to other instances providing services.