# Distributed Design Pattern

So here we have a few options to go through, and one of the options we can take a look at would be ambassador design pattern and it is very similar to ‘Sidecar design pattern’ but not quite the same. One of the differences is that from Sidecar pattern is that the ambassador will be responsible for all connections to the application, and thus it is required for the application to be capable of reaching outside.

So what is it that the ambassador enables us to do. It makes far more options in regards to networking available to us without having to make it apart of the applications code base. In essence some of these are the functions added, for resilience we have circuit breaking, monitoring, routing and the ability to update our network configurations. So while these may not be too complicated to add or make for a modern application, but take an older legacy application then it might be near impossible to add all the functionality you would require.

Another thing to keep in mind is that we might want to manage these features, as the network and security needs goes up. As having different frameworks for doing the same thing depending on how an application is written would introduce unnecessary difficulty as the management team wouldn’t be familiar enough with the applications code base for it to go as smoothly and securely.

So how do we solve these issues, with management and allowing for legacy applications among other things to still function securely. In essence we are going to put the application behind a proxy that will handle the monitoring, routing and even security features, while being deployed in the same environment as the application host. Now with these features we have a more manageable and secure application. Some of the data we can get out of these ambassadors are latency, resources, metrics. As we have taken these features out of the application itself we gain the ability to manage them separately. This allows for teams to manage these separate features without impacting the application itself. We can also have more specialized teams working on security, networking etc, as these features are now the same everywhere we have chosen to use this design pattern.

So while it’s good to have all these features, there’s no design pattern that is just consisting of benefits and no drawbacks. We will be adding some latency in overhead as we do introduce another jump into the process, so how latency reliant is your application? Does it require super fast responses? Maybe you should use a client side library in that case.

Another thing to keep in mind is whether it would be better to use an application specific integration, as it might allow for a deeper integration and allow for less overhead. Is it even suitable for your application to be using a sidecar of this type or even at all, as it does increase

the overhead and if your application doesn’t benefit enough from it, maybe we should avoid it all together.
Those were some of the concerns I had when looking into and reading about this design pattern. I hope that some of the information that I included here have shed some light and shown what the ambassador design pattern means and how it functions.

One relatively common reason to be using this design pattern is when we want to access a caching service, Memcached, Redis etc. So instead of configuring all of that in the application itself, we tell the application to listen to localhost, and leave the finding of the backend for the ambassador. As it enables the developer to leave the connection as localhost, instead of having to potentially work with another team to figure out how to connect it all up. Thus we can have the application think it reached our caching server, right when it sends it connection out to the ambassador.The ambassador then relays the connection to a server in its configuration that is healthy and running properly. By doing this we are segregating the responsibilities of each container we are running and making sure each container is doing what it’s best at.

I hope this has been somewhat informative and I think that we can see that there are some useful bits of information contained in here for anyone that will be reading this.