Instruction

In Linux, make sure the current directory is the container of go.mod.
Start the app by command:

```
go run app/master/master.go
```

Or start by the binary file bin/master, which can be compiled by run build.sh script

Configuration

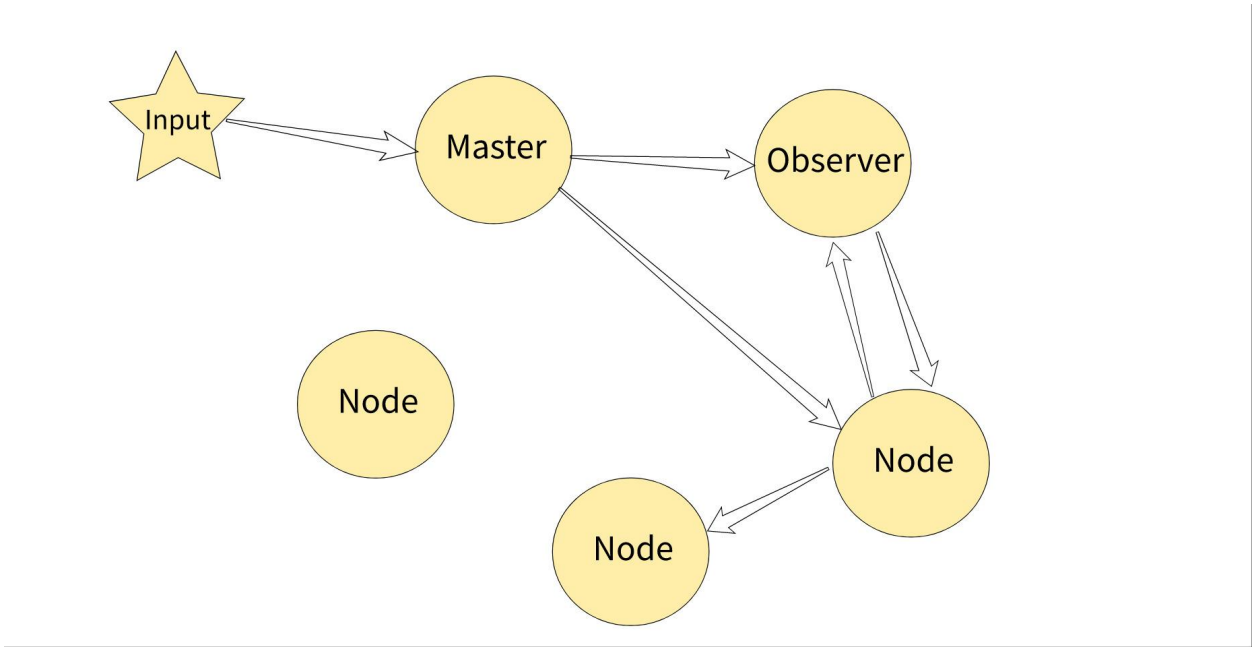
Master program read the default configuration in config.json. This configuration can be change by append the start command with the other configuration path such as

```
go run app/master/master.go config2.json
```

Name	Type	Default Value	Meaning
IsProduction	boolean	false	Running mode, this attribute have no impact yet
UseBin	boolean	false	Master program will start other program by binary file or by source file
InputFile	string	"input.ini"	Path of input file
UseLog	boolean	false	Log the communication between nodes, it can help debug operation
PrintInput	boolean	true	Print the input command to console
MasterId	int32	823412	The default id of master program
MasterPort	int32	9000	The default port of master program, make sure this port is available
ObserverId	int32	823413	The default id of observer program
ObserverPort	int32	9001	The default port of master program, make sure this port is available

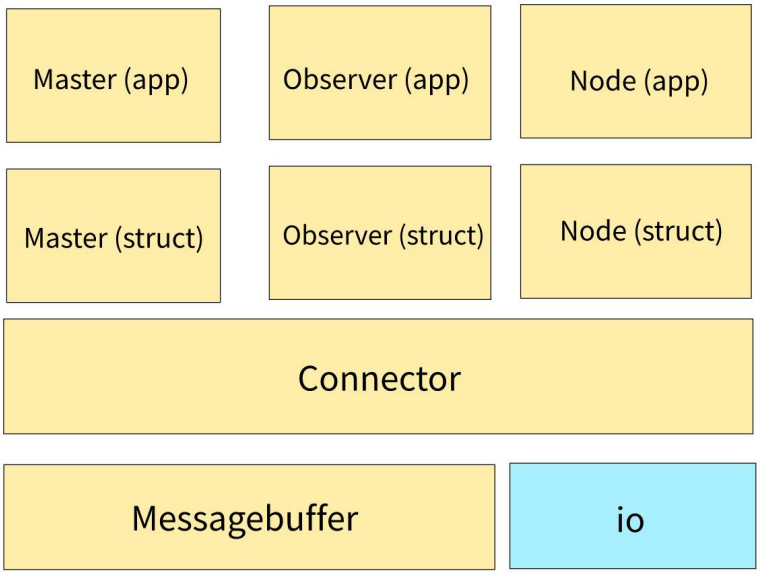
Configuration description

Design



This system is designed following the project specification. It is just a emulator system to visualize the Chandy-Lamport global snapshot algorithm. It is different from the real Chandy-Lamport working flow, and away further from the real system.
In this system, the master, the observer, the nodes are different os processes. For simplify, the system should be started in the order of (master, observer, nodes). If the system is started in different orders, it cannot work correctly.
Master program receives user input and send the command to others (nodes or observer) for visulize purpose.
Observer can send snapshot commands.
Node can receive snapshot commands from observer and send/receiver command from master.

Implemetation



This application is designed and implemented by bottom up approach.
Firstly, the Messagebuffer is implemented to serialize/deserialize primitive data including int32, int64 and string.
Then, the Connector uses that Messagebuffer and golang builtin package to send and receive data over the network (network on the same host and different port).
After that, master, observer, nodes use the Connector to communicate each other. The project business, Chandy-Lamport algorithm, is implemented here.
Finally, master application, observer application, node application is created as the entry of processes.