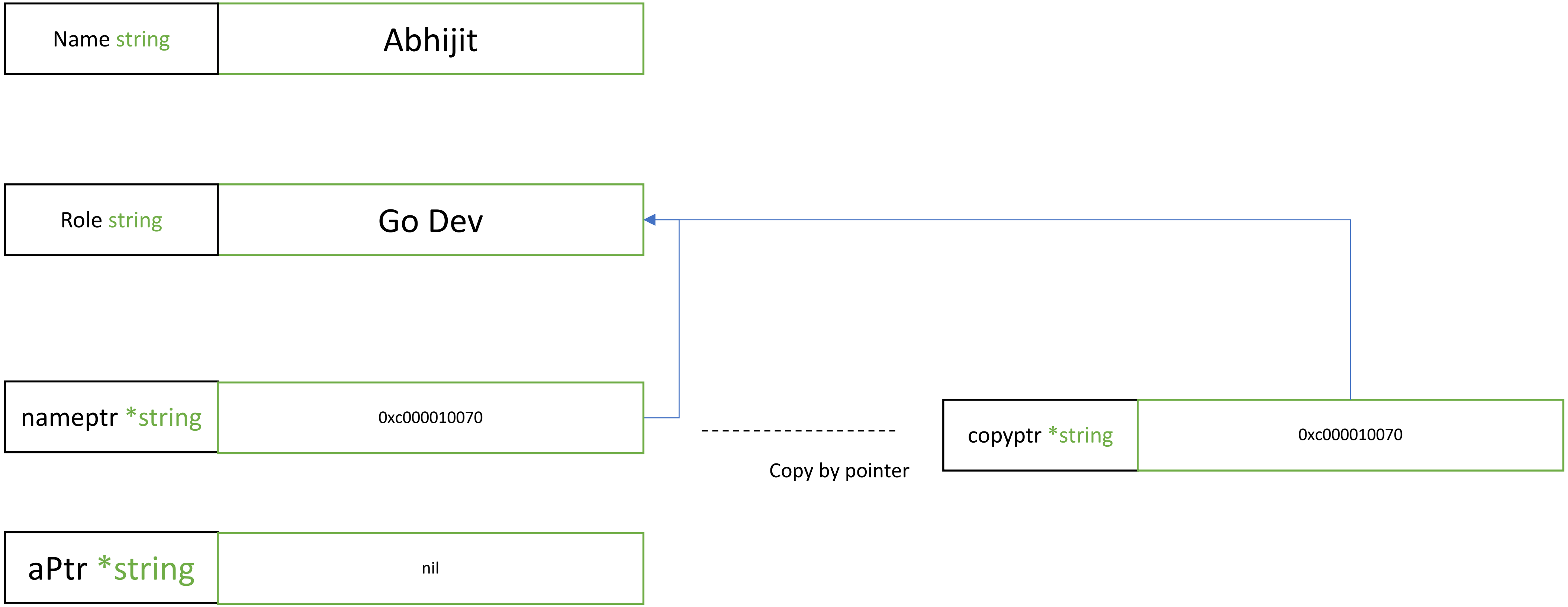


Name string

Abhijit

Role string

Go Dev

nameptr *string

0xc000010070

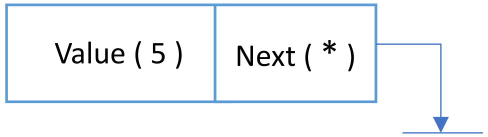
Copy by pointer

copyptr *string

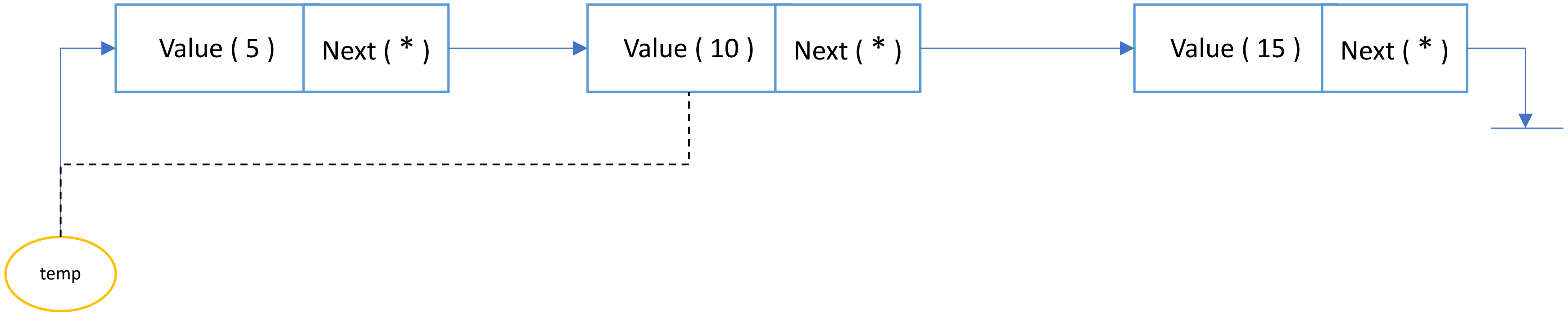
0xc000010070

aPtr *string

nil



Value (5) Next (*)

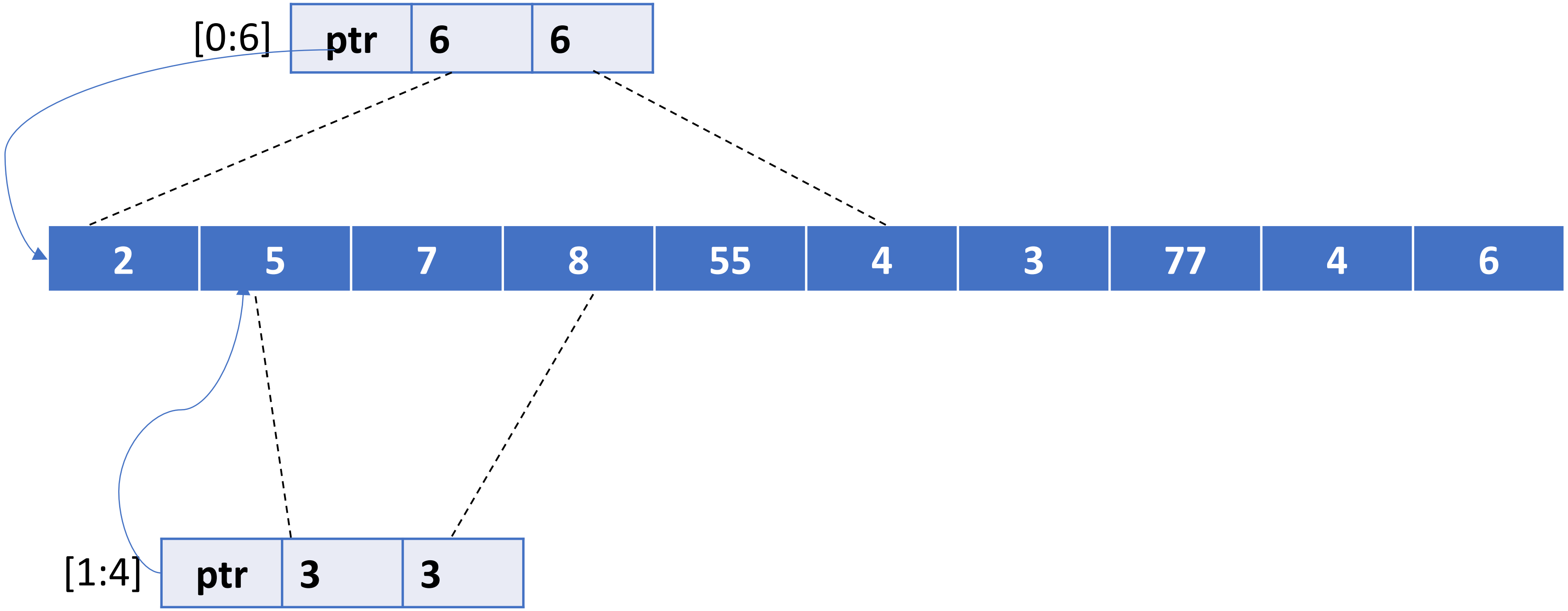


Value (5) Next (*)

Value (10) Next (*)

Value (15) Next (*)

temp



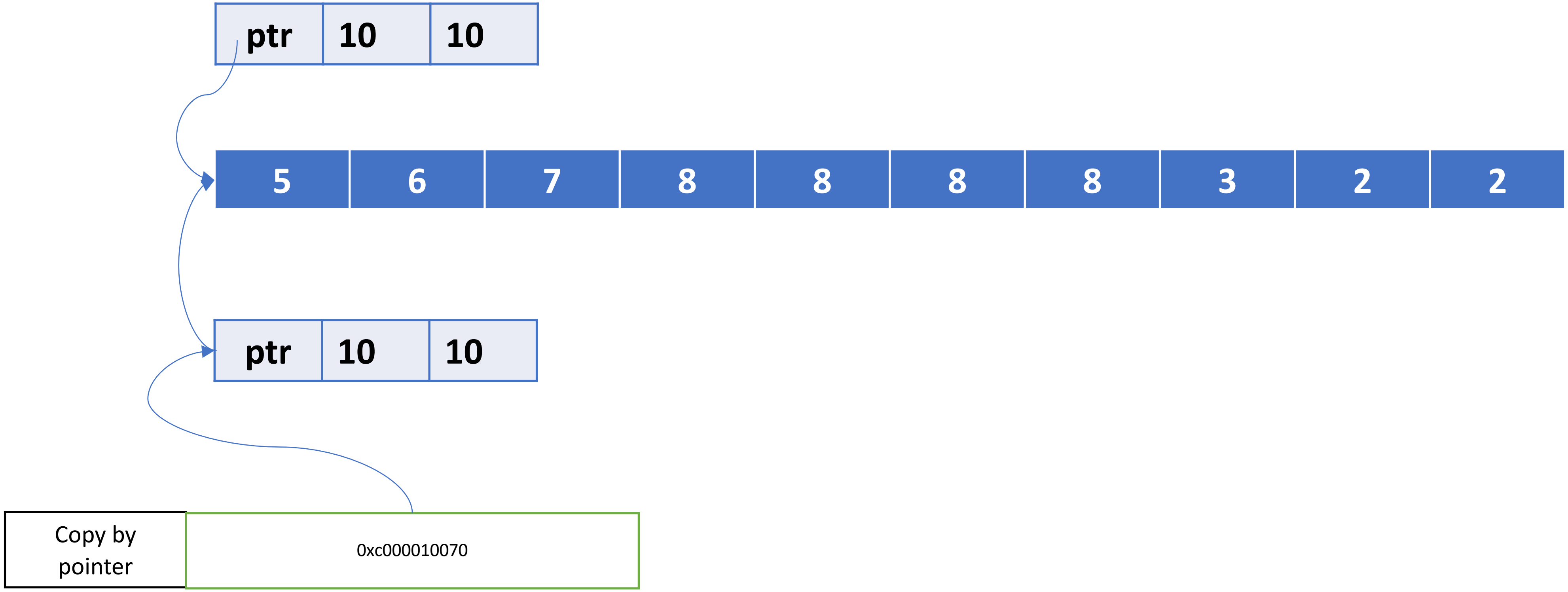
[0:6]

ptr	6	6

2	5	7	8	55	4	3	77	4	6

[1:4]

ptr	3	3

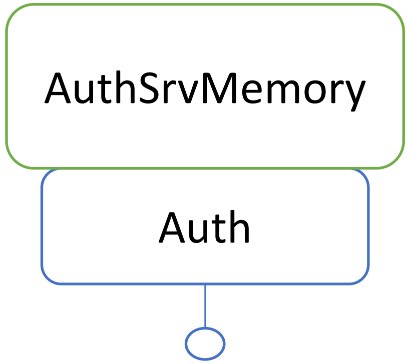


ptr	10	10

5	6	7	8	8	8	8	3	2	2

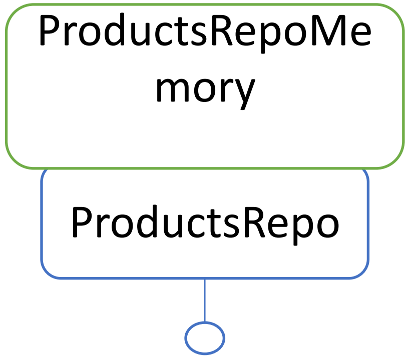
ptr	10	10

Copy by pointer	0xc000010070



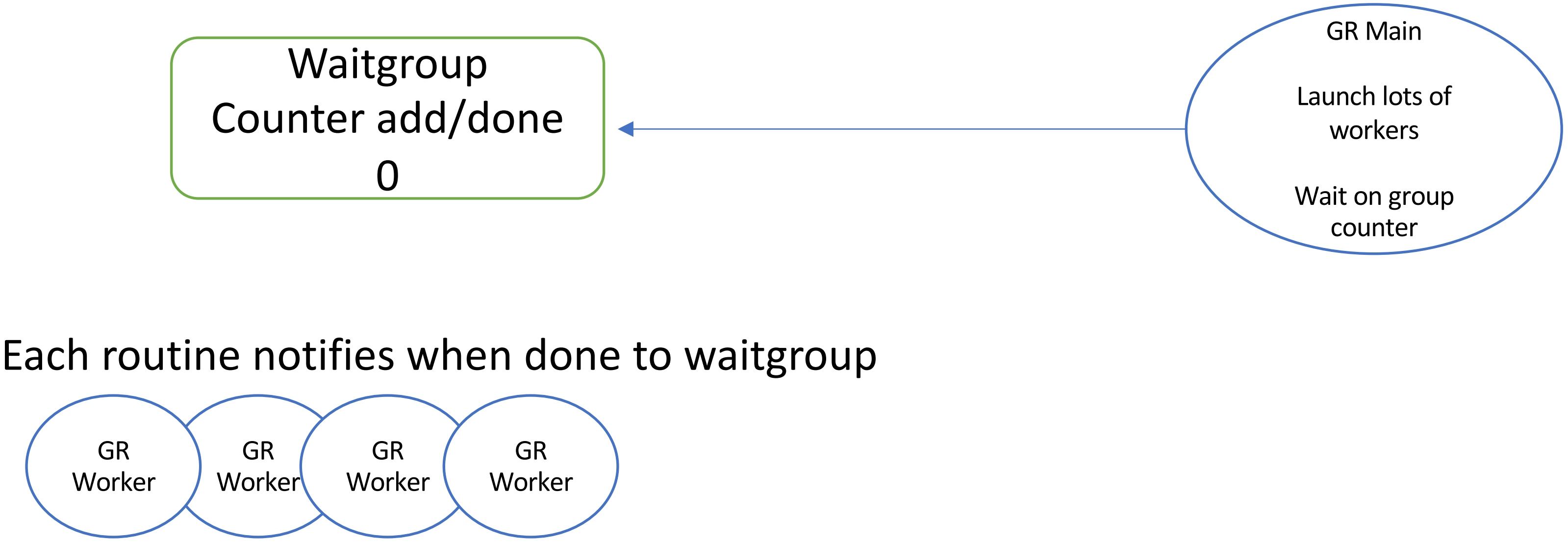
AuthSrvMemory

Auth



ProductsRepoMemory

ProductsRepo



Waitgroup
Counter add/done
0

GR Main

Launch lots of workers

Wait on group counter

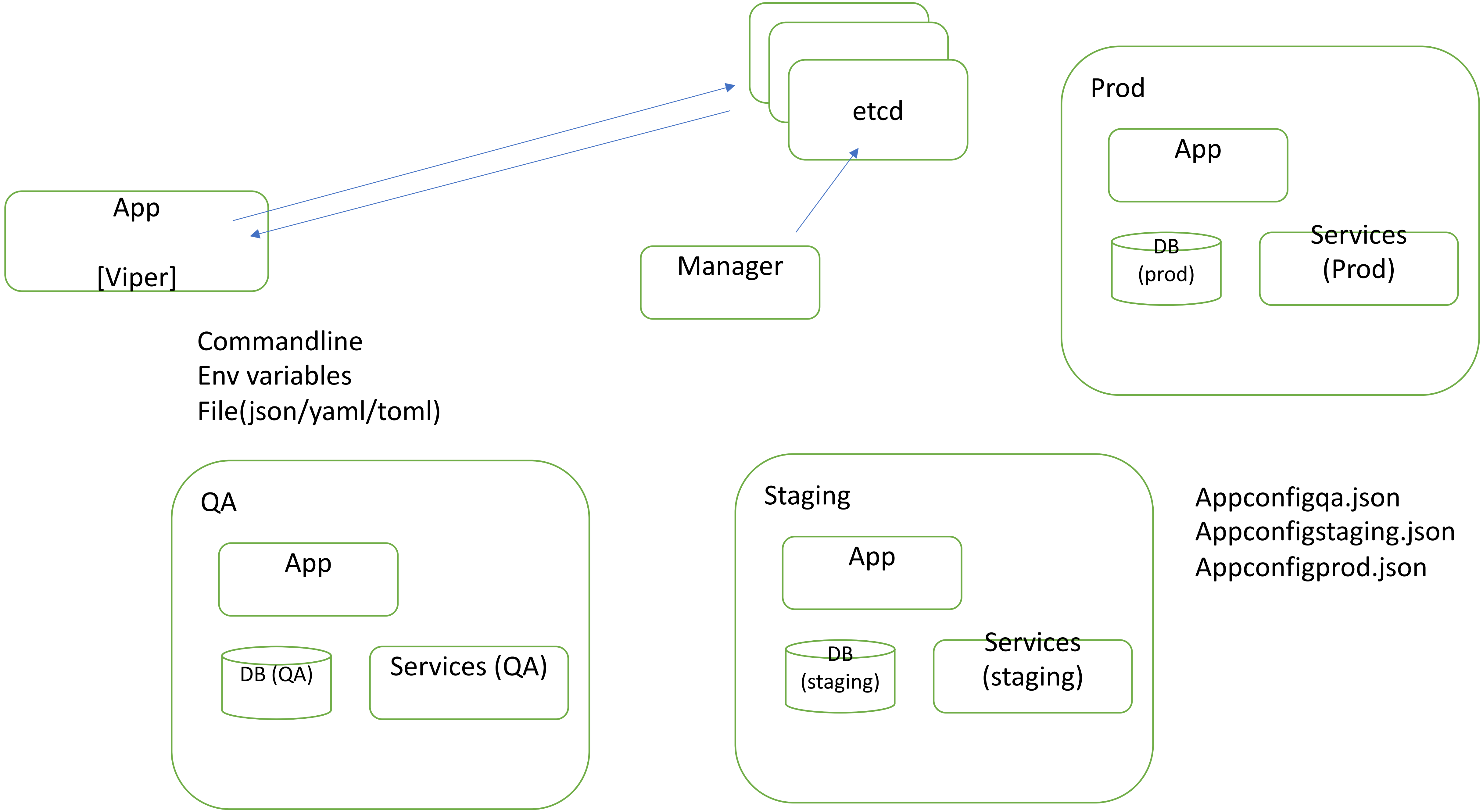
Each routine notifies when done to waitgroup

GR Worker

GR Worker

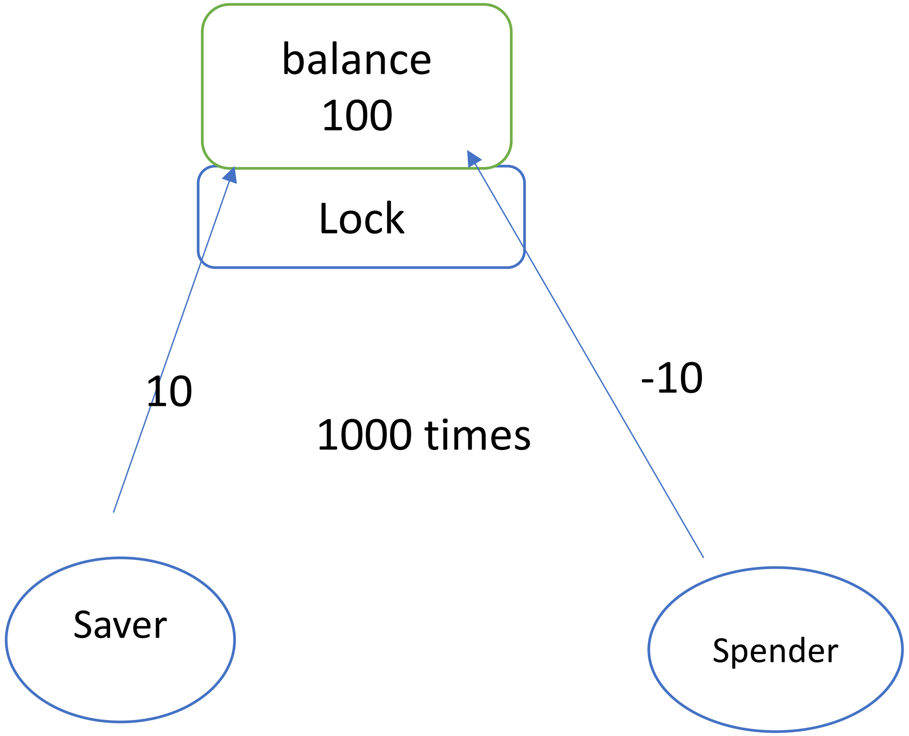
GR Worker

GR Worker



Commandline
Env variables
File(json/yaml/toml)

Appconfigqa.json
Appconfigstaging.json
Appconfigprod.json



balance
100

Lock

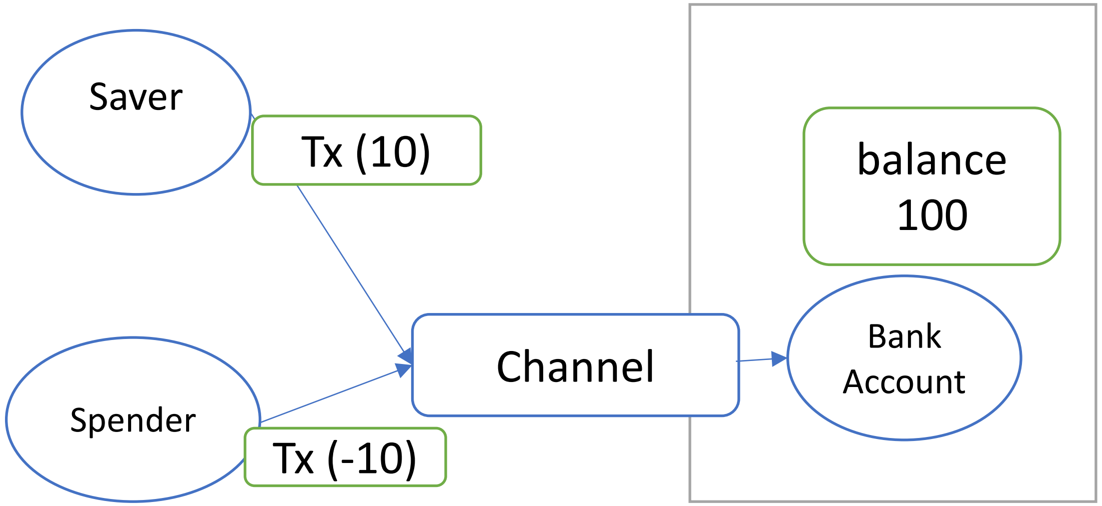
10

1000 times

-10

Saver

Spender



Saver

Tx (10)

balance
100

Channel

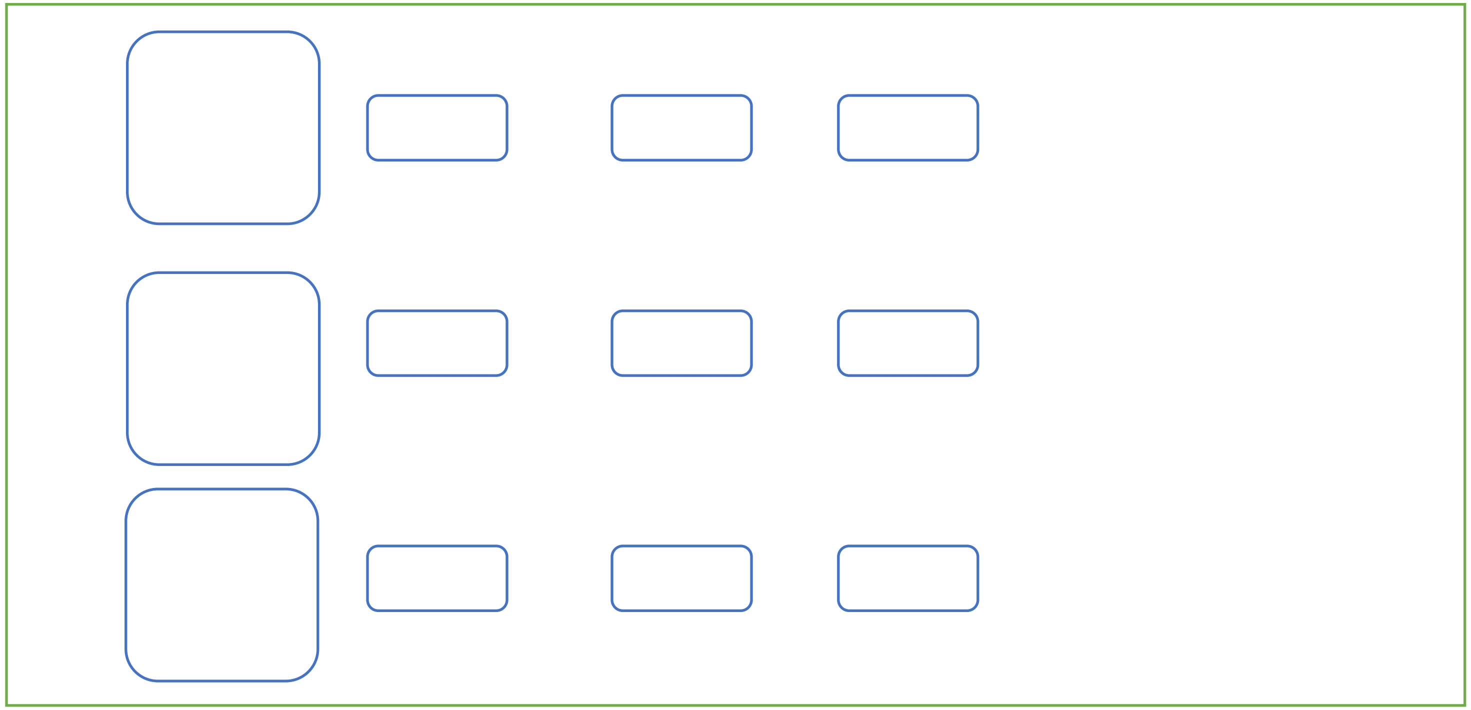
Bank
Account

Spender

Tx (-10)

key	value
Electronics	10
Mobile	5
Grocery	4
etc	15
..	
..	

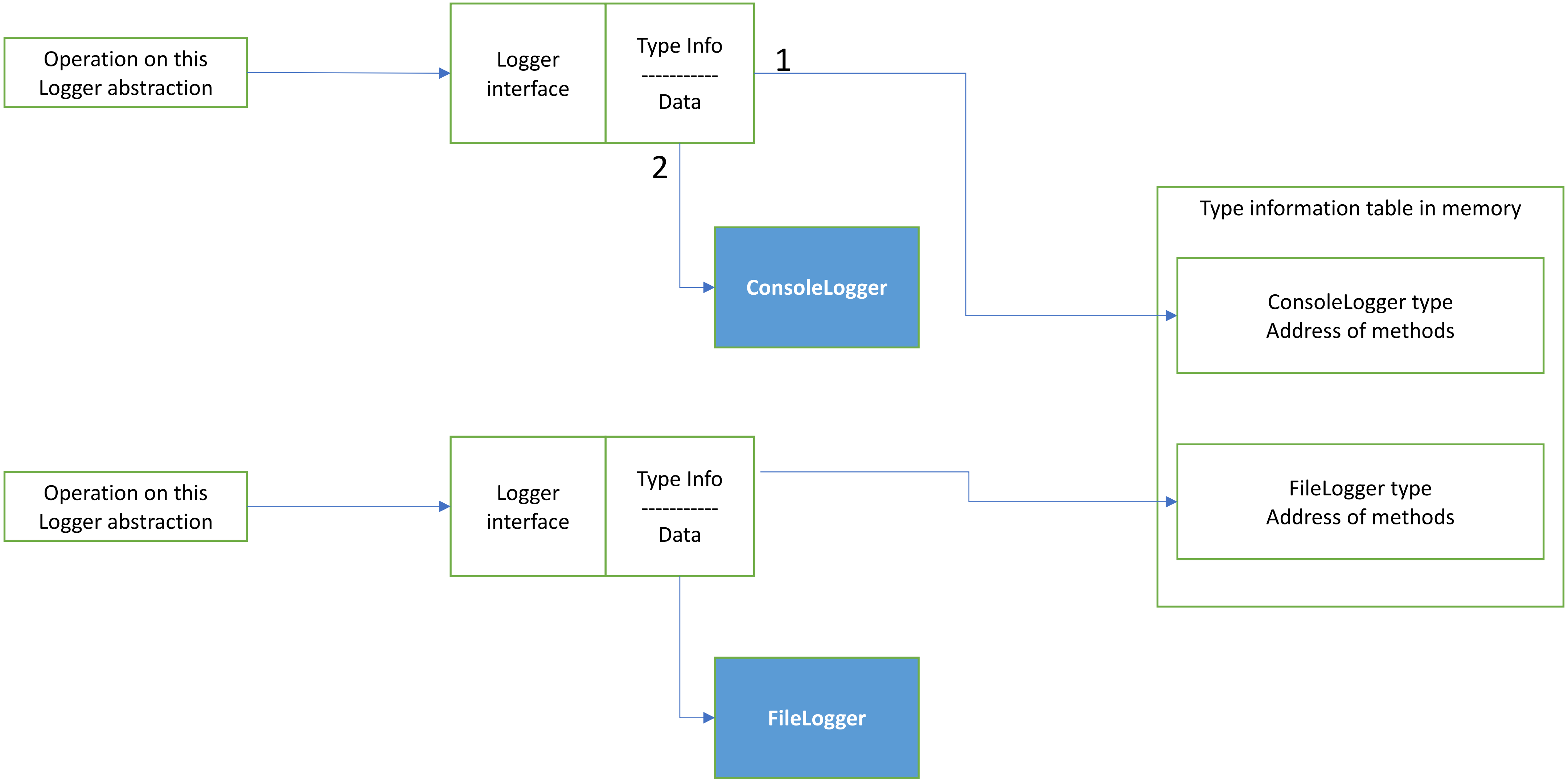
Electronics	1000
Mobile	1001



Interface points to or *contains 2 things:

1) Type info

2) Data, which is actual instance in memory

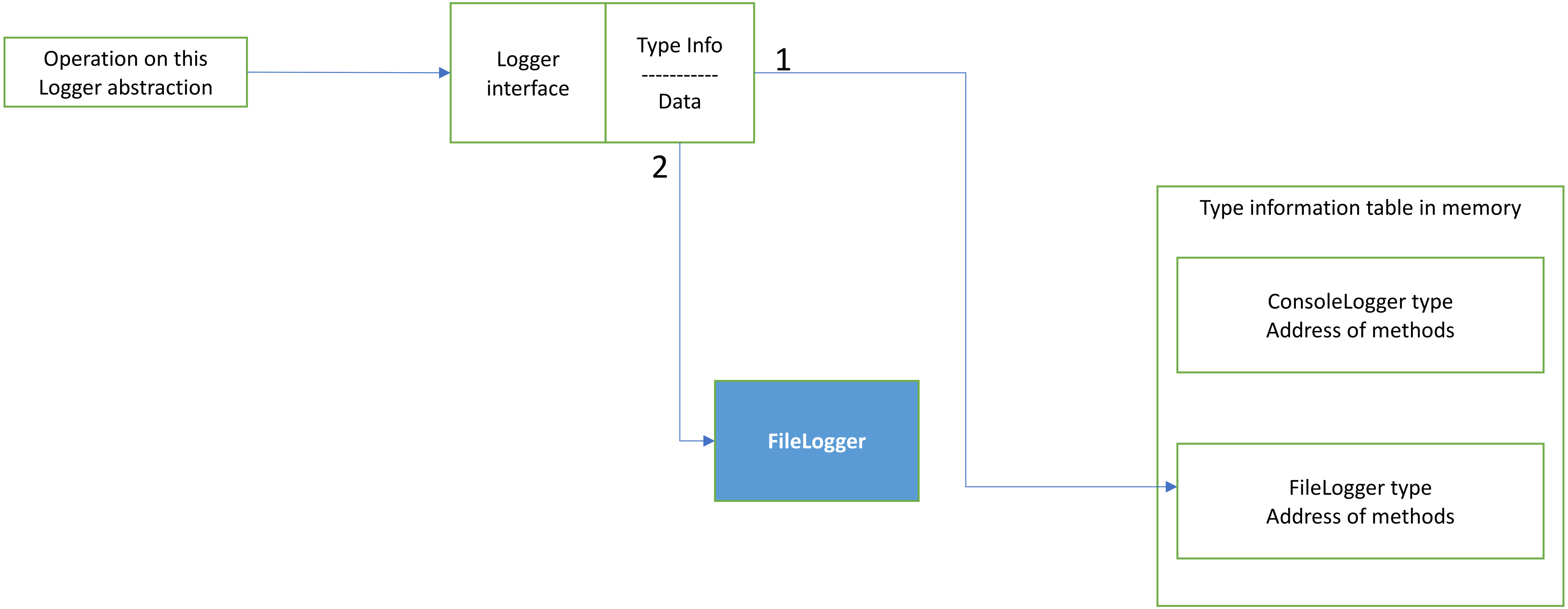

Operation on this Logger abstraction
Logger interface
Type Info

Data
1
2
ConsoleLogger
Type information table in memory
ConsoleLogger type Address of methods
FileLogger type Address of methods
Operation on this Logger abstraction
Logger interface
Type Info

Data
FileLogger

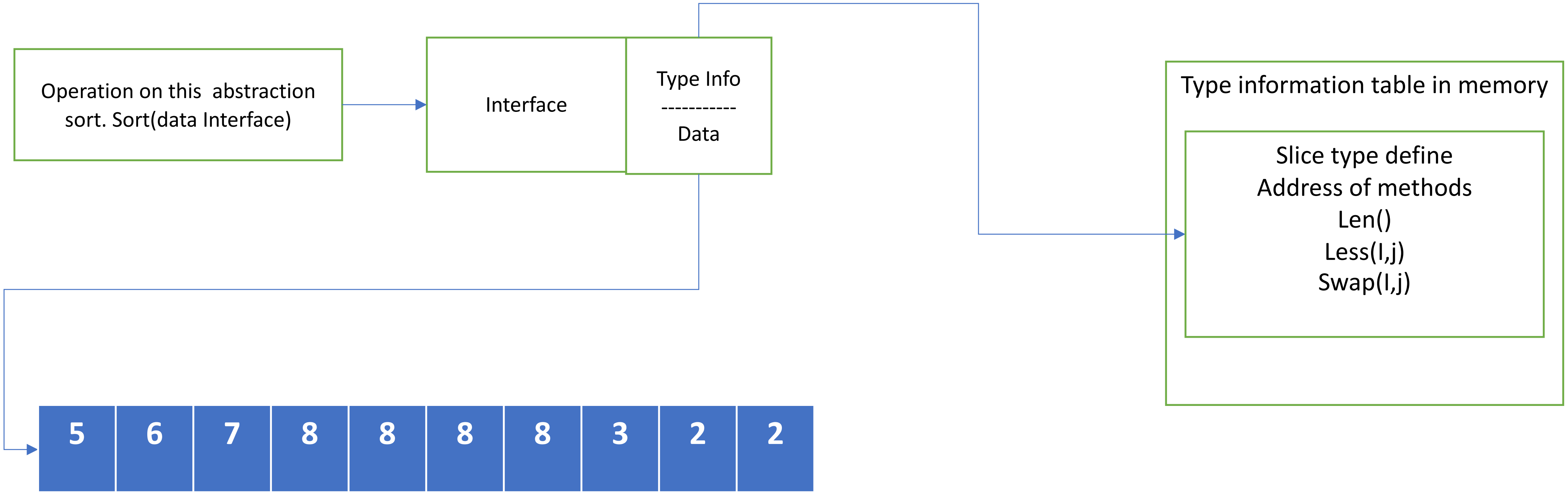
Interface points to or *contains 2 things:

1) Type info
2) Data, which is actual instance in memory



https://pkg.go.dev/sort

```
type Interface interface {
   Len() int
   Less(i, j int) bool
   Swap(i, j int)
}
```



Operation on this abstraction sort. Sort(data Interface)

Interface

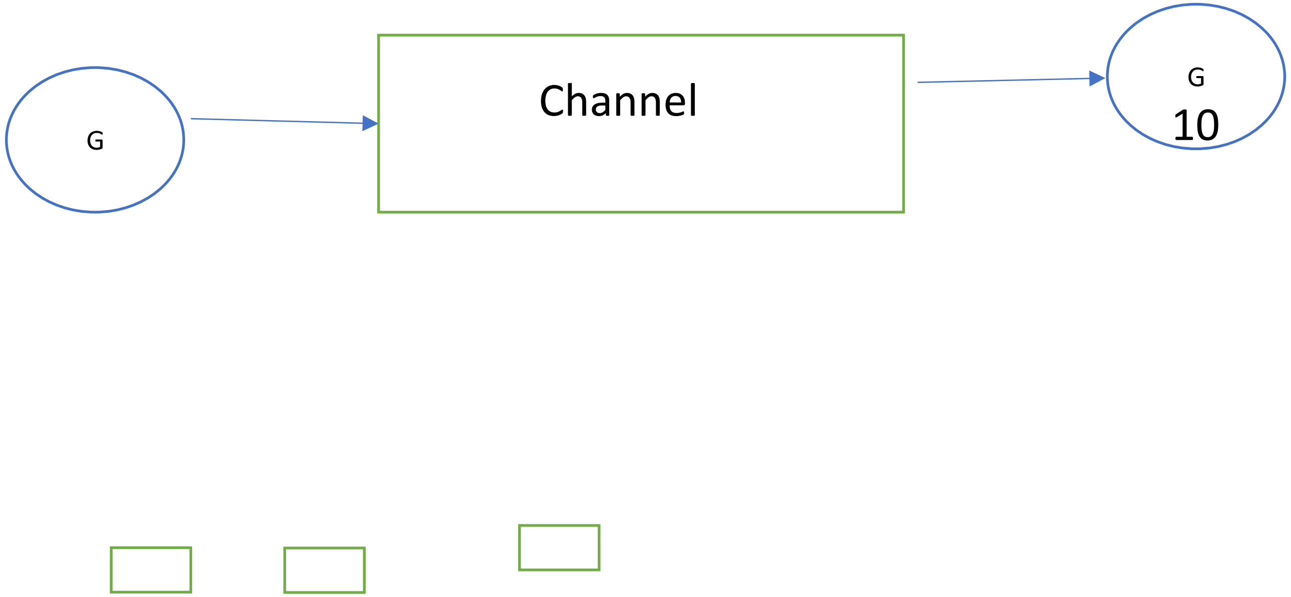
Type Info

Data

Type information table in memory

Slice type define
Address of methods
Len()
Less(I,j)
Swap(I,j)

5	6	7	8	8	8	8	3	2	2

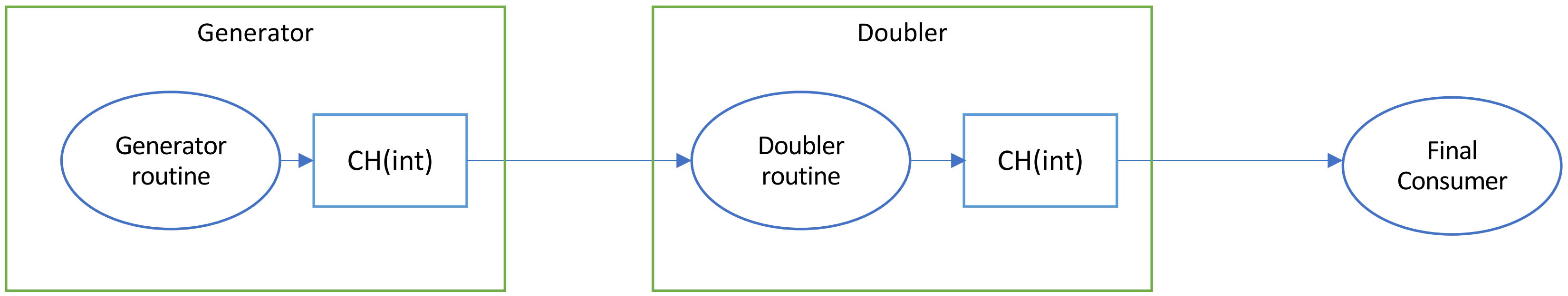


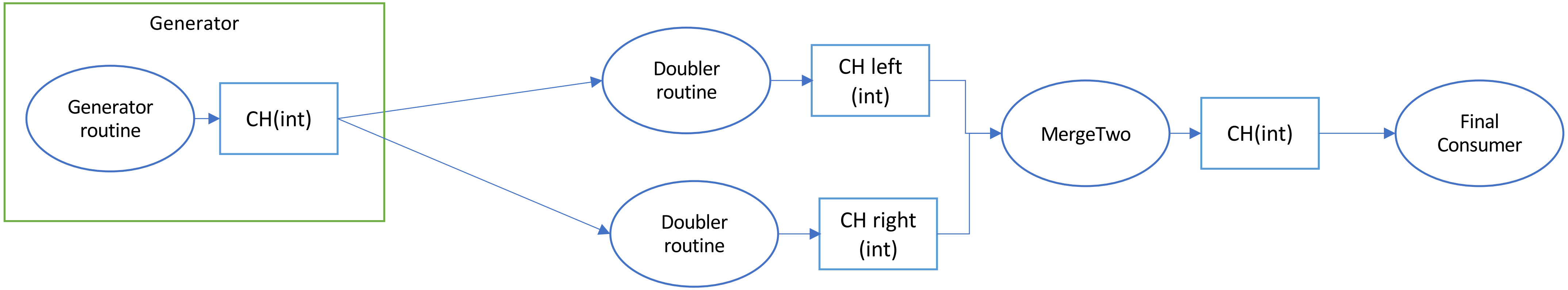
G

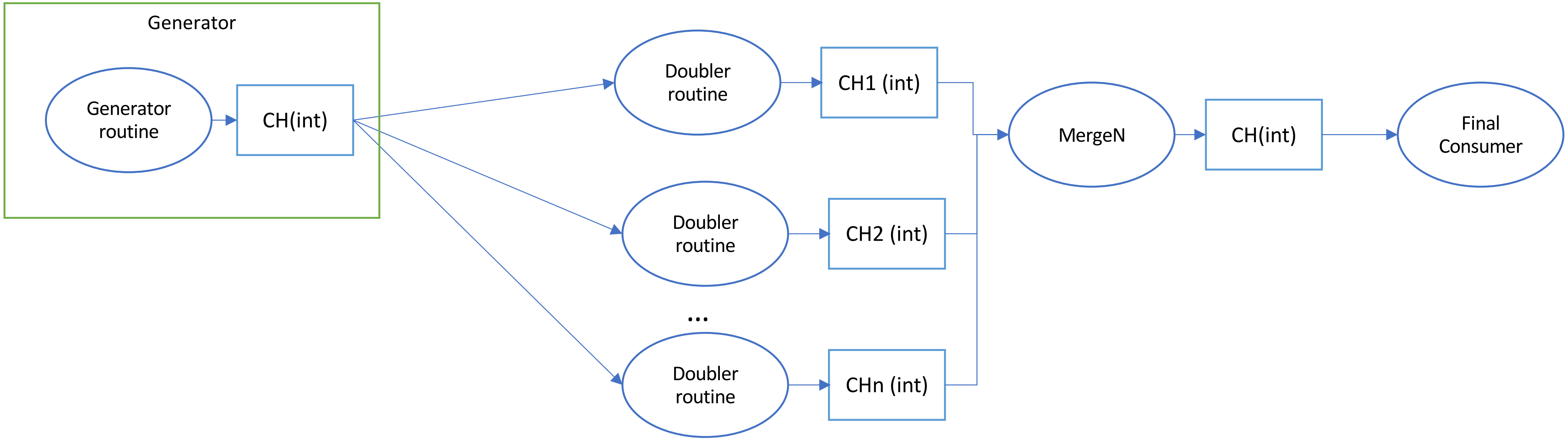
Channel

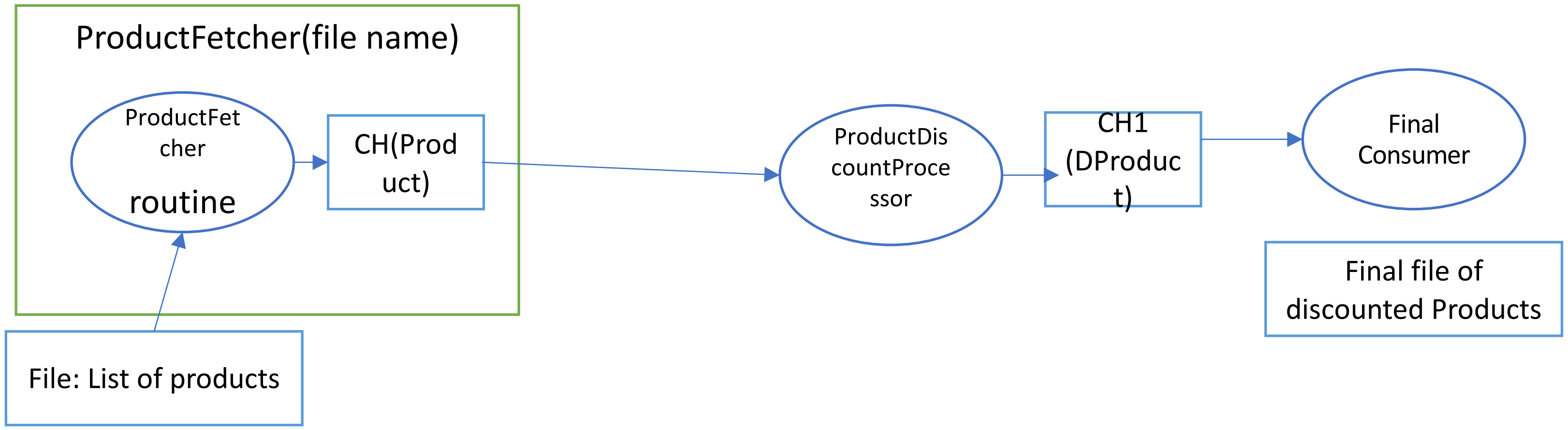
G

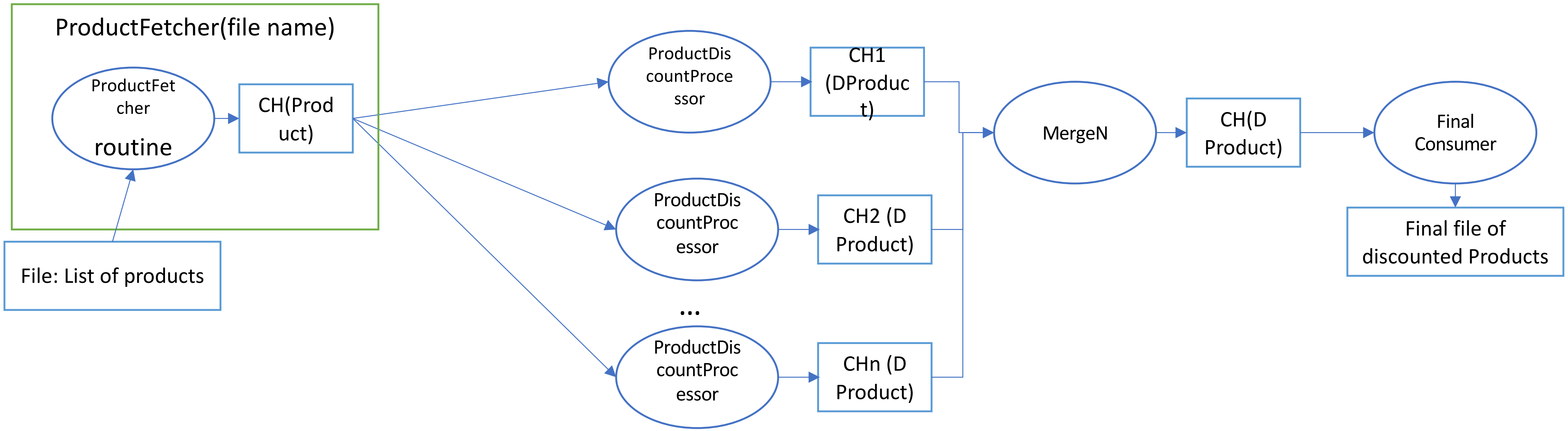
10


Generator
Generator routine
CH(int)
Doubler
Doubler routine
CH(int)
Final Consumer


Generator
Generator routine
CH(int)
Doubler routine
CH left (int)
Doubler routine
CH right (int)
MergeTwo
CH(int)
Final Consumer


Generator
Generator routine
CH(int)
Doubler routine
CH1 (int)
Doubler routine
CH2 (int)
...
Doubler routine
CHn (int)
MergeN
CH(int)
Final Consumer


ProductFetcher(file name)
ProductFetcher routine
CH(Product)
File: List of products
ProductDiscountProcessor
CH1 (DProduct)
Final Consumer
Final file of discounted Products

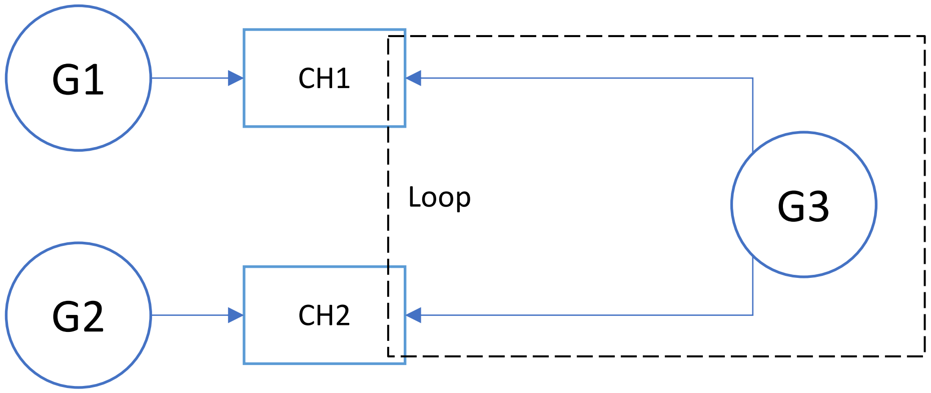

ProductFetcher(file name)
ProductFetcher routine
CH(Product)
File: List of products
ProductDiscountProcessor
CH1 (DProduct)
ProductDiscountProcessor
CH2 (D Product)
…
ProductDiscountProcessor
CHn (D Product)
MergeN
CH(D Product)
Final Consumer
Final file of discounted Products

G1 is slow producer

G2 is fast producer

G3 is reading CH1 and then CH2

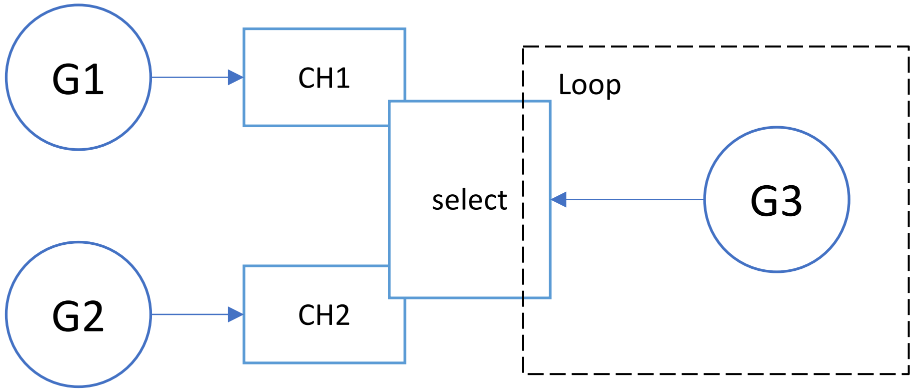
This will block G2 from producing values fast. Since it cannot send further values unless G3 reads them fast



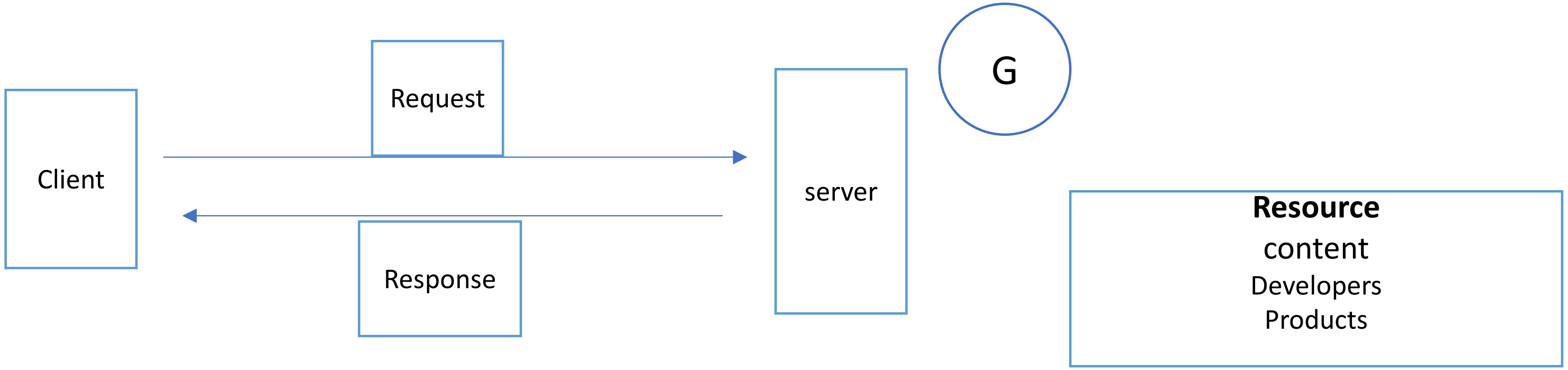
select will let G3 know if there is activity on CH1 or CH2.

Without a default case select is blocking

If you add a default case, then it is non blocking



Go routine for every http request concurrently



Client

Request

Response

server

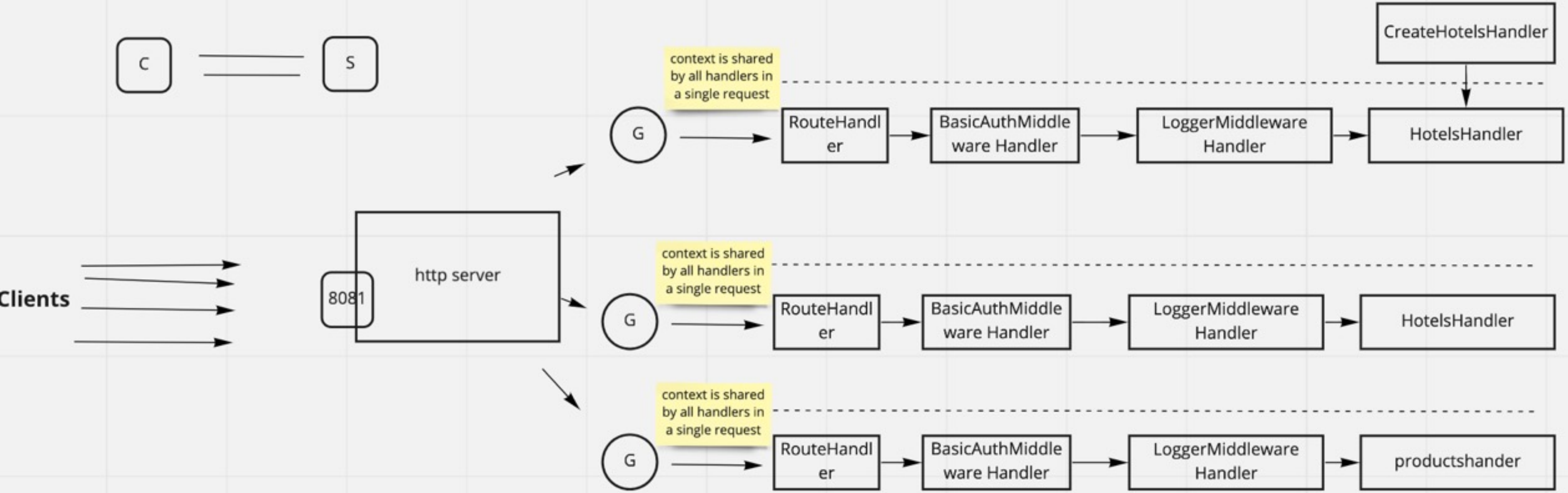
G

Resource

content

Developers

Products


C
S
CreateHotelsHandler
context is shared by all handlers in a single request
G
RouteHandler
BasicAuthMiddleware Handler
LoggerMiddleware Handler
HotelsHandler
Clients
8081
http server
context is shared by all handlers in a single request
G
RouteHandler
BasicAuthMiddleware Handler
LoggerMiddleware Handler
HotelsHandler
context is shared by all handlers in a single request
G
RouteHandler
BasicAuthMiddleware Handler
LoggerMiddleware Handler
productshander

```
type Handler interface {
    ServeHTTP(ResponseWriter, *Request)
}
```

func(http.ResponseWriter, *http.Request)

Adapter for interface to simple handler: (gives equivalent of functional interface in java)

```
type HandlerFunc func(ResponseWriter, *Request)

 // ServeHTTP calls f(w, r).
     func (f HandlerFunc) ServeHTTP(w ResponseWriter, r *Request) {
        f(w, r)
     }
```