CS492 Distributed Transactions and Blockchain

Tutorial on Go language and HyperLedger

Himchan Park

School of Computing, KAIST

Outline

- **The Go programming Language**
  - basic constructs, simple programs
  - arrays & slices
  - maps
  - methods, interfaces
  - concurrency, goroutines
  - Homework 1. simple key-value database written in Go

- **HyperLedger Fabric (HLF)**
  - Overview of HLF
  - Building Fabric network on Docker VM
  - Writing ChainCode as Smart contract
  - Homework 2. Bank transfer
  - Homework 3. Bank transfer alleviating transaction conflict

The Go Programming Language

Himchan Park
School of Computing
KAIST

Go

- developed ~2007 at Google by
  Robert Griesemer, Rob Pike, Ken Thompson
- open source

- C-like syntax
- compiled, statically typed
  - very fast compilation
- garbage collection
- built-in concurrency
- no classes or type inheritance or overloading or generics
  - unusual interface mechanism instead of inheritance

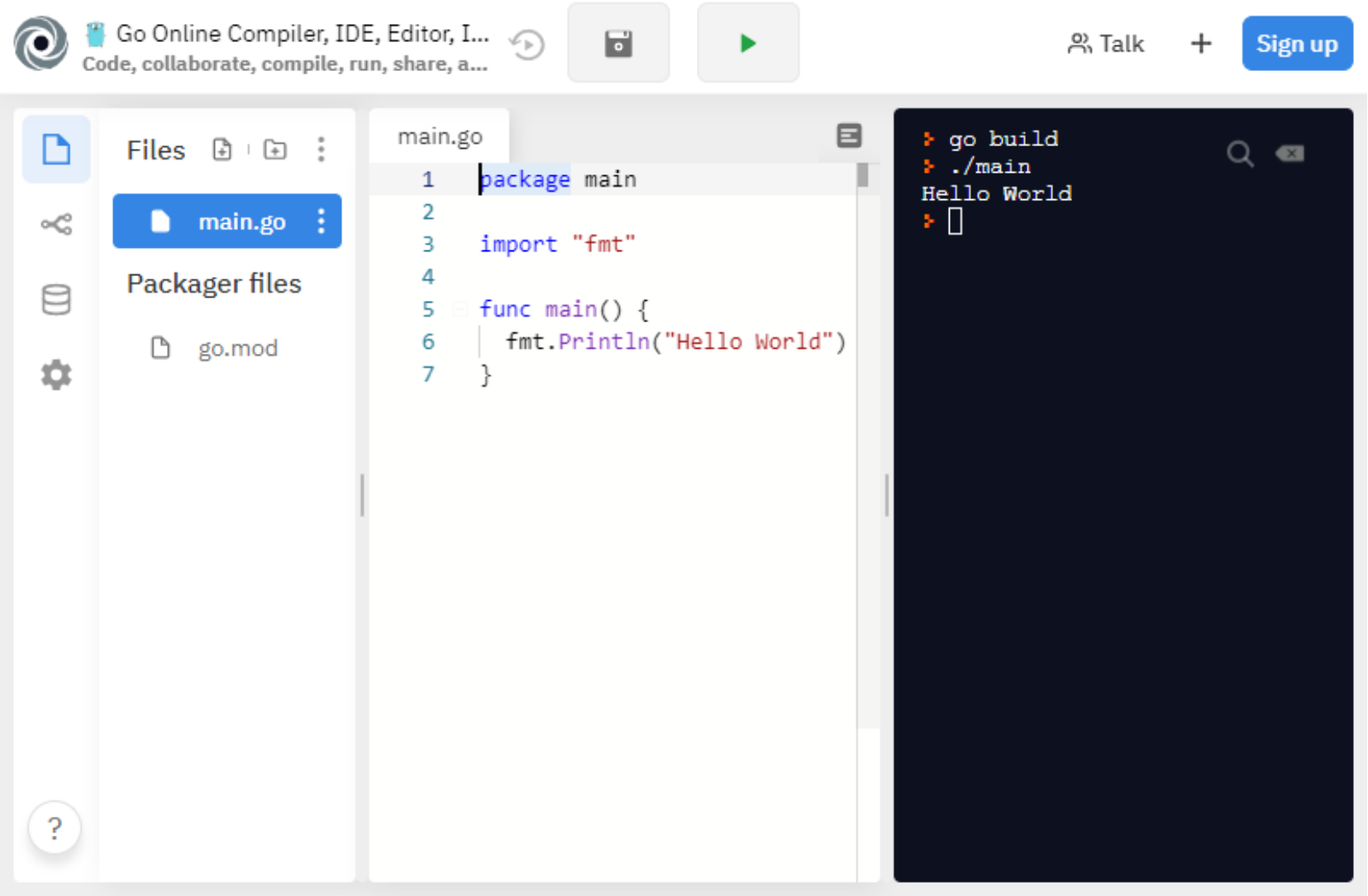
Hello world in Go

```
package main
import "fmt"
func main() {
    fmt.Println("Hello, 世界")
}
```

```
$ go run hello.go     # to compile and run
$ go build hello.go   # to create a binary

$ go help             # for more
```

Repl.it (Web IDE)

- **REPL (Read-Eval-Print-Loop)**
  - For go-lang: https://repl.it/languages/go



Types, constants, variables

- basic types

  bool string int8 int16 int32 int64 uint8 … int uint

  float32 float64 complex64 complex128

  quotes: ‘世’, “UTF-8 string”, `raw string`

- variables

```
var c1, c2 int
var x, y, z = 0, 1.23, false   // variable decls

x := 0; y := 1.23; z := false  // short variable decl
```

  Go infers the type from the type of the initializer

  assignment between items of different type requires an explicit conversion, e.g., int(float_expression)

- operators
  - mostly like C, but `++` and `--` are postfix only and not expressions
  - assignment is not an expression
  - no `?:` operator

Echo command:

```
// Echo prints its command-line arguments.
package main

import (
  "fmt"
  "os"
)

func main() {
   var s, sep string
   for i := 1; i < len(os.Args); i++ {
      s += sep + os.Args[i]
      sep = " "
   }
   fmt.Println(s)
}
```

Echo command (version 2):

```
// Echo prints its command-line arguments.
package main

import (
  "fmt"
  "os"
)

func main() {
   s, sep := "", ""
   for _, arg := range os.Args[1:] {
      s += sep + arg
      sep = " "
   }
   fmt.Println(s)
}
```

Arrays and slices

- an array is a fixed-length sequence of same-type items

```
months := [...]string {1:"Jan", 2:"Feb", /*...,*/ 12:"Dec"}
```

- a slice is a subsequence of an array

```
summer := months[6:8]; Q2 := months[4:7]
```

- elements accessed as `slice[index]`
  - indices from 0 to `len(slice)-1` inclusive

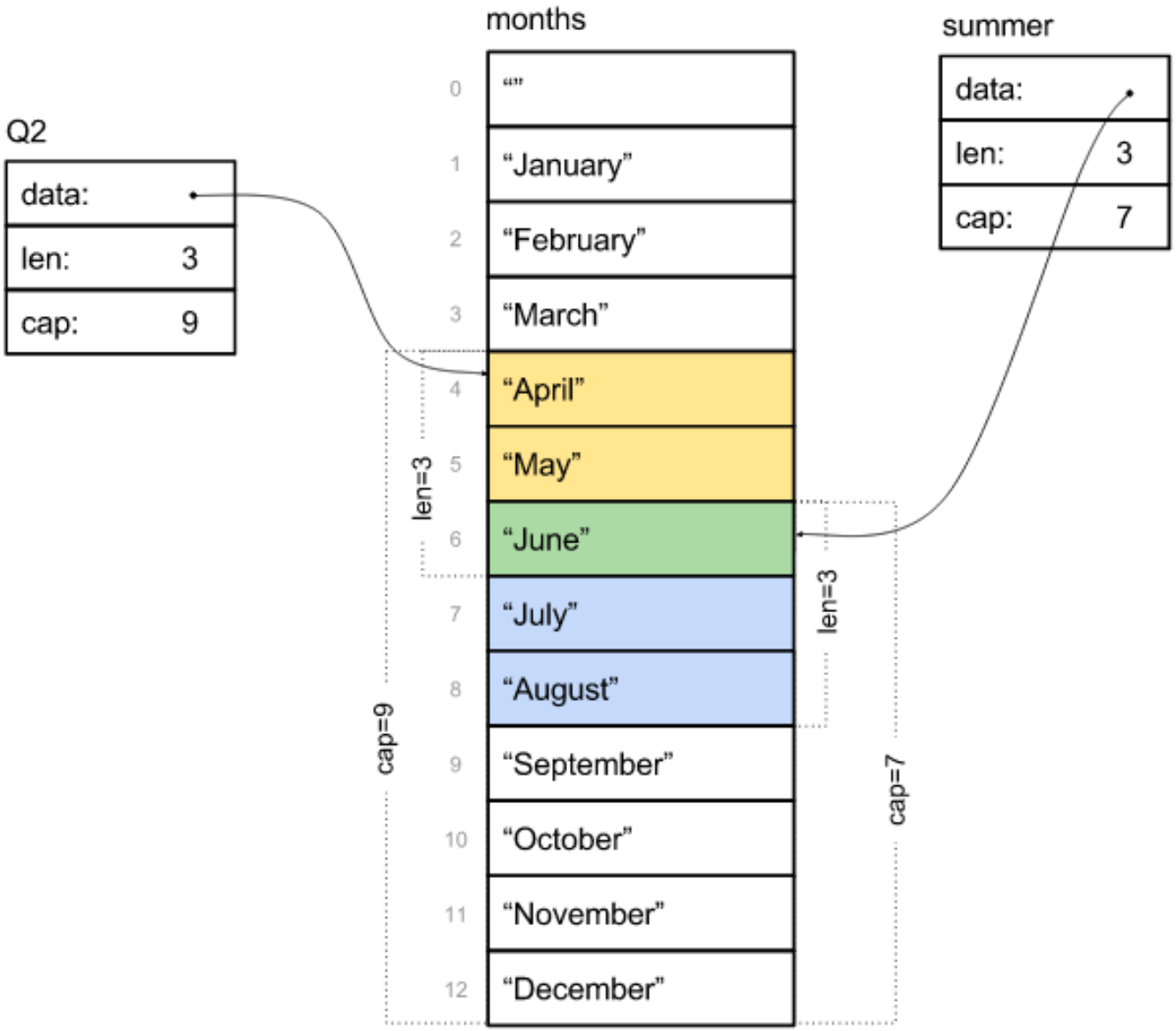
  `summer[0:3]` is elements `months[6:8]`

  `summer[0] = "June"`

- loop over a slice with for range

```
for i, v := range summer {
  fmt.Println(i, v)
}
```

- slices are very efficient (represented as small structures)
- most library functions work on slices


months
summer
Q2
data:
len: 3
cap: 9
data:
len: 3
cap: 7
0 “”
1 “January”
2 “February”
3 “March”
4 “April”
5 “May”
6 “June”
7 “July”
8 “August”
9 “September”
10 “October”
11 “November”
12 “December”
len=3
cap=9
len=3
cap=7

Maps (== associative arrays)

- **unordered collection of key-value pairs**
  - keys are any type that supports `==` and `!=` operators
  - values are any type

```
// Find duplicated lines in stdin.
func main() {
  counts := make(map[string]int)
  in := bufio.NewScanner(os.Stdin)
  for in.Scan() {
    counts[in.Text()]++
  }
  for line, n := range counts {
    if n > 1 {
          fmt.Printf("%d\t%s\n", n, line)
    }
  }
}
```

Methods and pointers

- can define methods that work on any type, including your own:

```
type Vertex struct {
    X, Y float64
}
func (v *Vertex) Scale(f float64) {
    v.X = v.X * f
    v.Y = v.Y * f
}
func (v Vertex) Abs() float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}
func main() {
    v := &Vertex{3, 4}
    v.Scale(5)
    fmt.Println(v, v.Abs())
}
```

Interfaces

- an interface is satisfied by any type that implements all the methods of the interface
- completely abstract: can't instantiate one
- can have a variable with an interface type
- then assign to it a value of any type that has the methods the interface requires

- a type implements an interface merely by defining the required methods
  - it doesn't declare that it implements them

- Writer: the most common interface

```
type Writer interface {
    Write(p []byte) (n int, err error)
}
```

Sort interface

- sort interface defines three methods
- any type that implements those three methods can sort
- algorithms are inside the soft package, invisible outside

```
package sort

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

Sort interface (adapted from Go Tour)

```
type Person struct {
  Name string
  Age  int
}
func (p Person) String() string {
  return fmt.Sprintf("%s: %d", p.Name, p.Age)
}
type ByAge []Person

func (a ByAge) Len() int           { return len(a) }
func (a ByAge) Swap(i, j int)      { a[i], a[j] = a[j], a[i] }
func (a ByAge) Less(i, j int) bool { return a[i].Age < a[j].Age }

func main() {
  people := []Person{{"Bob",31}, {"Sue",42}, {"Ed",17}, {"Jen",26},}
  fmt.Println(people)
  sort.Sort(ByAge(people))
  fmt.Println(people)
}
```

Concurrency: goroutines & channels

- **channel: a type-safe generalization of Unix pipes**
  - inspired by Hoare's Communicating Sequential Processes (1978)

- **goroutine: a function executing concurrently with other goroutines in the same address space**
  - run multiple parallel computations simultaneously
  - loosely like threads but much lighter weight

- **channels coordinate computations by explicit communication**
  - locks, semaphores, mutexes, etc., are much less often used

Example: web crawler

- **want to crawl a bunch of web pages to do something**
  - e.g., figure out how big they are
- **problem: network communication takes relatively long time**
  - program does nothing useful while waiting for a response
- **solution: access pages in parallel**
  - send requests asynchronously
  - display results as they arrive
  - needs some kind of threading or other parallel process mechanism
- **takes less time than doing them sequentially**

Version 1: no parallelism

```
func main() {
  start := time.Now()
  for _, site := range os.Args[1:] {
    count("http://" + site)
  }
  fmt.Printf("%.2fs total\n", time.Since(start).Seconds())
}

func count(url string) {
  start := time.Now()
  r, err := http.Get(url)
  if err != nil {
    fmt.Printf("%s: %s\n", url, err)
    return
  }
  n, _ := io.Copy(ioutil.Discard, r.Body)
  r.Body.Close()
  dt := time.Since(start).Seconds()
  fmt.Printf("%s %d [%.2fs]\n", url, n, dt)
}
```

Version 2: parallelism with goroutines

```
func main() {
  start := time.Now()
  c := make(chan string)
  n := 0
  for _, site := range os.Args[1:] {
    n++
    go count("http://" + site, c)
  }
  for i := 0; i < n; i++ {
    fmt.Print(<-c)
  }
  fmt.Printf("%.2fs total\n", time.Since(start).Seconds())
}
func count(url string, c chan<- string) {
  start := time.Now()
  r, err := http.Get(url)
  if err != nil {
    c <- fmt.Sprintf("%s: %s\n", url, err)
    return
  }
  n, _ := io.Copy(ioutil.Discard, r.Body)
  r.Body.Close()
  dt := time.Since(start).Seconds()
  c <- fmt.Sprintf("%s %d [%.2fs]\n", url, n, dt)
}
```

Action in practice: A Tour of Go

- **https://tour.golang.org/**

Interfaces

An *interface type* is defined as a set of method signatures.

A value of interface type can hold any value that implements those methods.

Note: There is an error in the example code on line 22. `Vertex` (the value type) doesn't implement `Abser` because the `Abs` method is defined only on `*Vertex` (the pointer type).

interfaces.go Imports off Syntax off

```
1  package main
2
3  import (
4      "fmt"
5      "math"
6  )
7
8  type Abser interface {
9      Abs() float64
10 }
11
12 func main() {
13     var a Abser
14     f := MyFloat(-math.Sqrt2)
15     v := Vertex{3, 4}
16
17     a = f  // a MyFloat implements Abser
18     a = &v // a *Vertex implements Abser
19
20     // In the following line, v is a Vertex (not *Vertex)
21     // and does NOT implement Abser.
22     a = v
23
24     fmt.Println(a.Abs())
25 }
26
27 type MyFloat float64
28
29 func (f MyFloat) Abs() float64 {
30     if f < 0 {
31         return float64(-f)
32     }
33     return float64(f)
34 }
35
36 type Vertex struct {
37     X, Y float64
38 }
39
40 func (v *Vertex) Abs() float64 {
41     return math.Sqrt(v.X*v.X + v.Y*v.Y)
42 }
43
```

Reset Format Run

Homework1: key-value database (pseudo blockchain)

- **Make an *interface* should have two methods, PutState("key", "value") and GetState("key")**

- **Make a *structure* having a map[string]string and implementation of the two functions**

- **The return of methods should include error information whether there is an error or not**
  - Func PutState(key, value string) error { … }
  - Func GetState(key string) (string, error) { … }

- **All history of transactions are recorded sequentially for each file "history.block.$" (e.g., "history.block.1", "history.block.2", …)**
  - Each block is generated after 10 Txs. are invoked or when the program exits
  - Next block contains the hash value of previous block

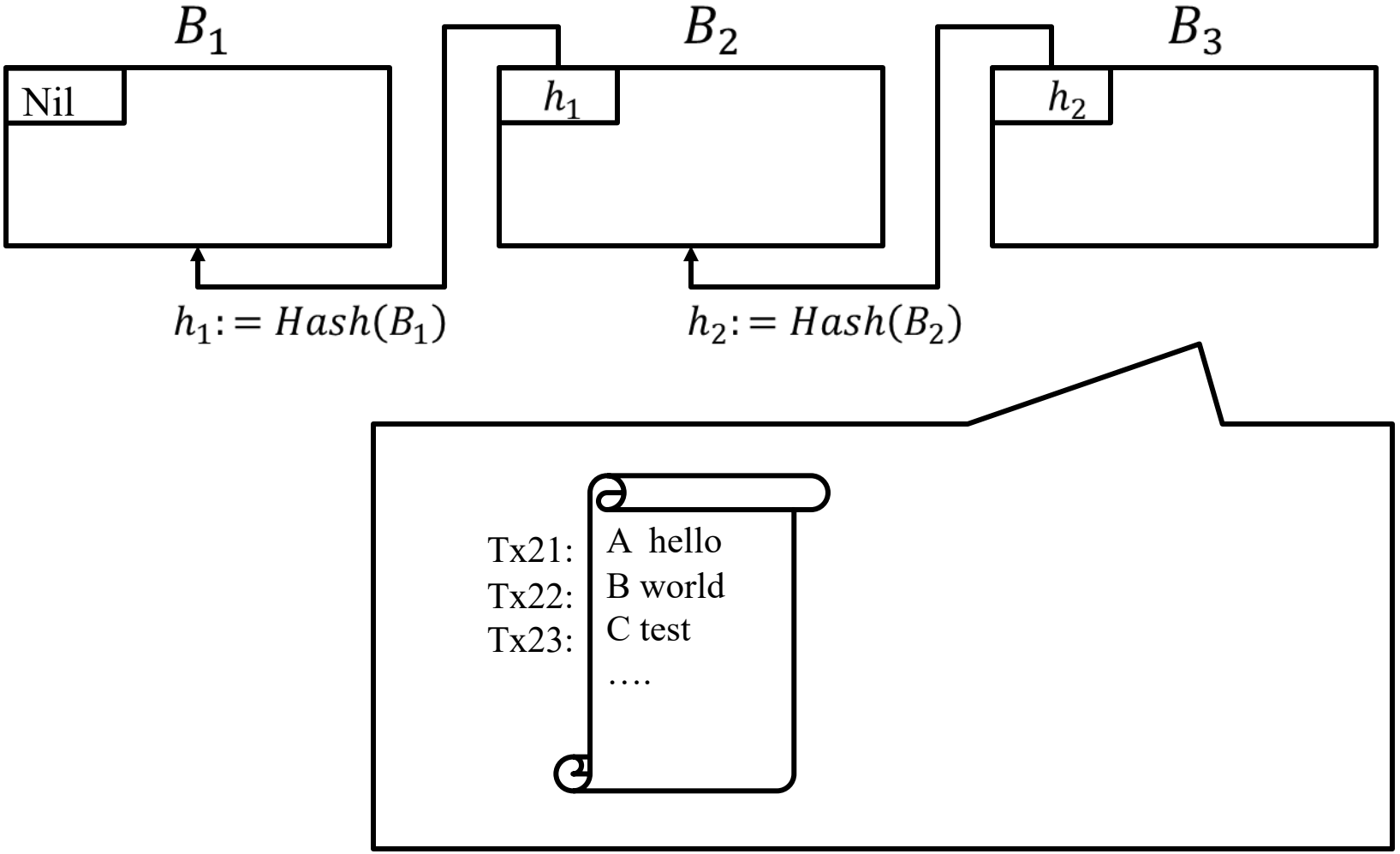
- **When the program exits, all key-value pair are stored in a file "state.db"**

- **When the program launches, the stored DB is loaded**

Get hash function

```go
package main

import (
  "crypto/sha256"
  "fmt"
  "io"
  "log"
  "os"
)

func main() {
  f, err := os.Open("file.txt")
  if err != nil {
    log.Fatal(err)
  }
  defer f.Close()

  h := sha256.New()
  if _, err := io.Copy(h, f); err != nil {
    log.Fatal(err)
  }

  fmt.Printf("%x", h.Sum(nil))
}
```

Blockchain examples



Homework#1 template

- hash
  - hash.go
- history
  - history.go
- kv
  - kv.go
- main.go

```go
package main

import ...

func main() {
    db := &kv.Database{}
    kv.Init(db)
    defer kv.Finalize(db)

    for i, arg := range os.Args[1:] {
        fmt.Println(i, "TestPutState("+arg+"_key,"+arg+"_value): ", db.PutState(arg+"_key", arg+"_value"))
    }

    for i, arg := range os.Args[1:] {
        value, err := db.GetState(arg + "_key")
        fmt.Println(i, "TestGetState("+arg+"_key): ", value, err)
    }
}
```

```go
package kv

import ...

type DatabaseInterface interface {
    GetState(key string) (string, error)
    PutState(key, value string) error
}

type Database struct {
    state     KV
    tempBlock history.History
}

type KV map[string]string
```

```go
package history

import (
    "../hash"
    "fmt"
    "io/ioutil"
    "strconv"
    "strings"
)

type History struct {
    txs            [10]string
    currentTxId    int
    currentBlockId int
}
```

Output example of HW#1

$ ./hw1 hello world

stdout

```
open state.db: The system cannot find the file specified.
0 TestPutState(hello_key,hello_value):  <nil>
1 TestPutState(world_key,world_value):  <nil>
0 TestGetState(hello_key):  hello_value <nil>
1 TestGetState(world_key):  world_value <nil>
```

state.db

```
hello_key   hello_value
world_key   world_value
```

history.block.1

```
hello_key   hello_value
world_key   world_value
```

$ ./hw1 second transactions invoked

state.db

```
world_key   world_value
second_key  second_value
transactions_key    transactions_value
invoked_key invoked_value
hello_key   hello_value
```

history.block.2

```
84e65ce0c3a3b43f4a71ec6adec0b93ed7c9c55947a02135241f6ac02612cd52
second_key  second_value
transactions_key    transactions_value
invoked_key invoked_value
```

$ ./hw1 1 2 3 4 5 6 7 8 9 10 11 12 13

state.db

```
world_key   world_value
hello_key   hello_value
4_key   4_value
7_key   7_value
10_key  10_value
12_key  12_value
second_key  second_value
1_key   1_value
2_key   2_value
3_key   3_value
9_key   9_value
13_key  13_value
transactions_key    transactions_value
5_key   5_value
6_key   6_value
8_key   8_value
invoked_key invoked_value
11_key  11_value
```

history.block.3

```
117568f6681d537f0ae0fe5fd80969914efa9a7d681595fa3917362c098adf5b
1_key   1_value
2_key   2_value
3_key   3_value
4_key   4_value
5_key   5_value
6_key   6_value
7_key   7_value
8_key   8_value
9_key   9_value
10_key  10_value
```

history.block.4

```
e1239205cf13677c5570f737b8555132891919e23211469a2114c18a00624d1e
11_key  11_value
12_key  12_value
13_key  13_value
```

HyperLedger Fabric

Himchan Park
School of Computing
KAIST

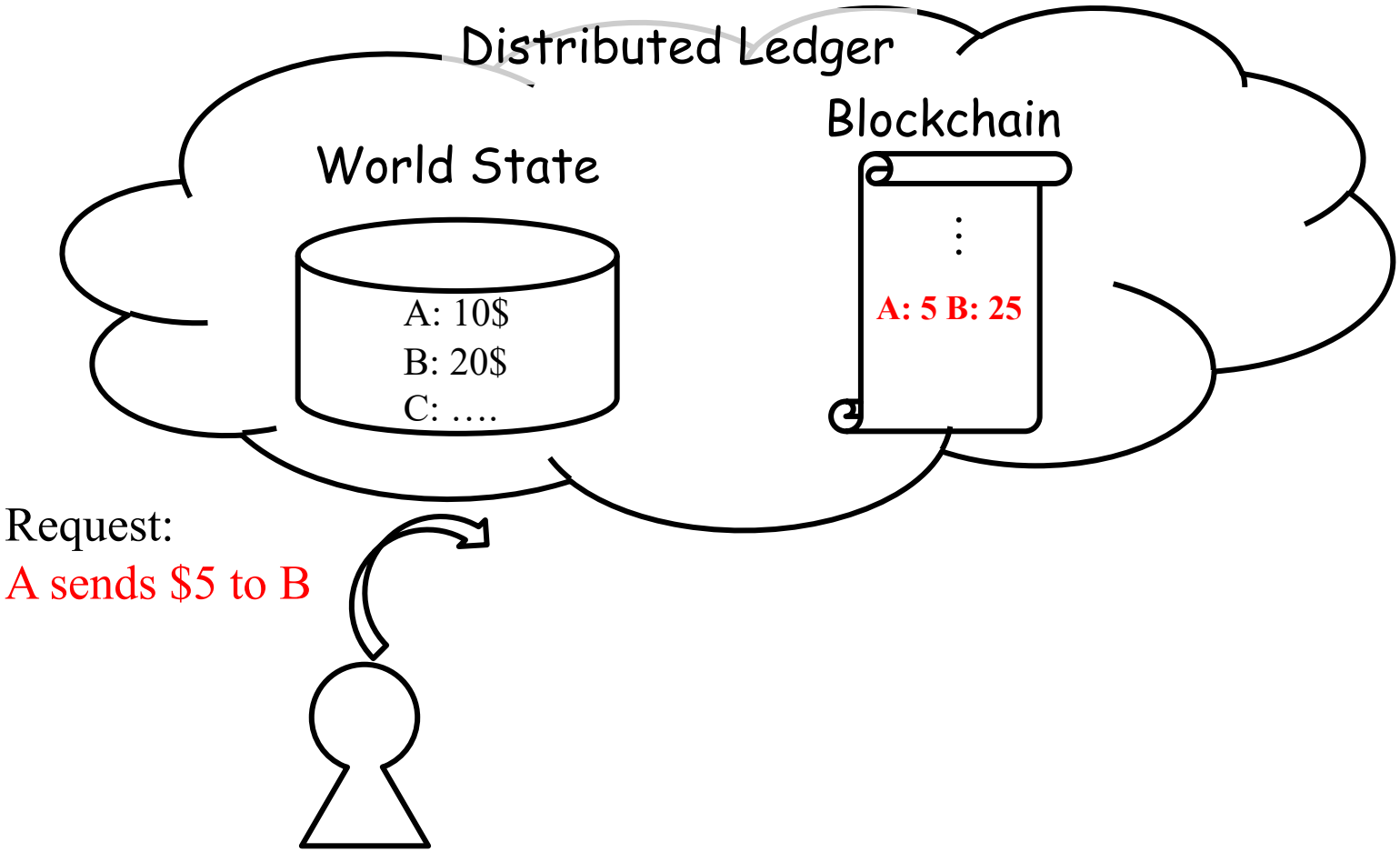
HyperLedger Fabric (HLF)

- **HLF is a platform for distributed ledger solution in industrial level**
- **A modular architecture of HLF delivers high level of confidentiality, resiliency, flexibility and scalability**
- **HLF supports smart contract development in general-purpose programming languages, such as Go, Java, and Node.js**

Distributed Ledger in HLF

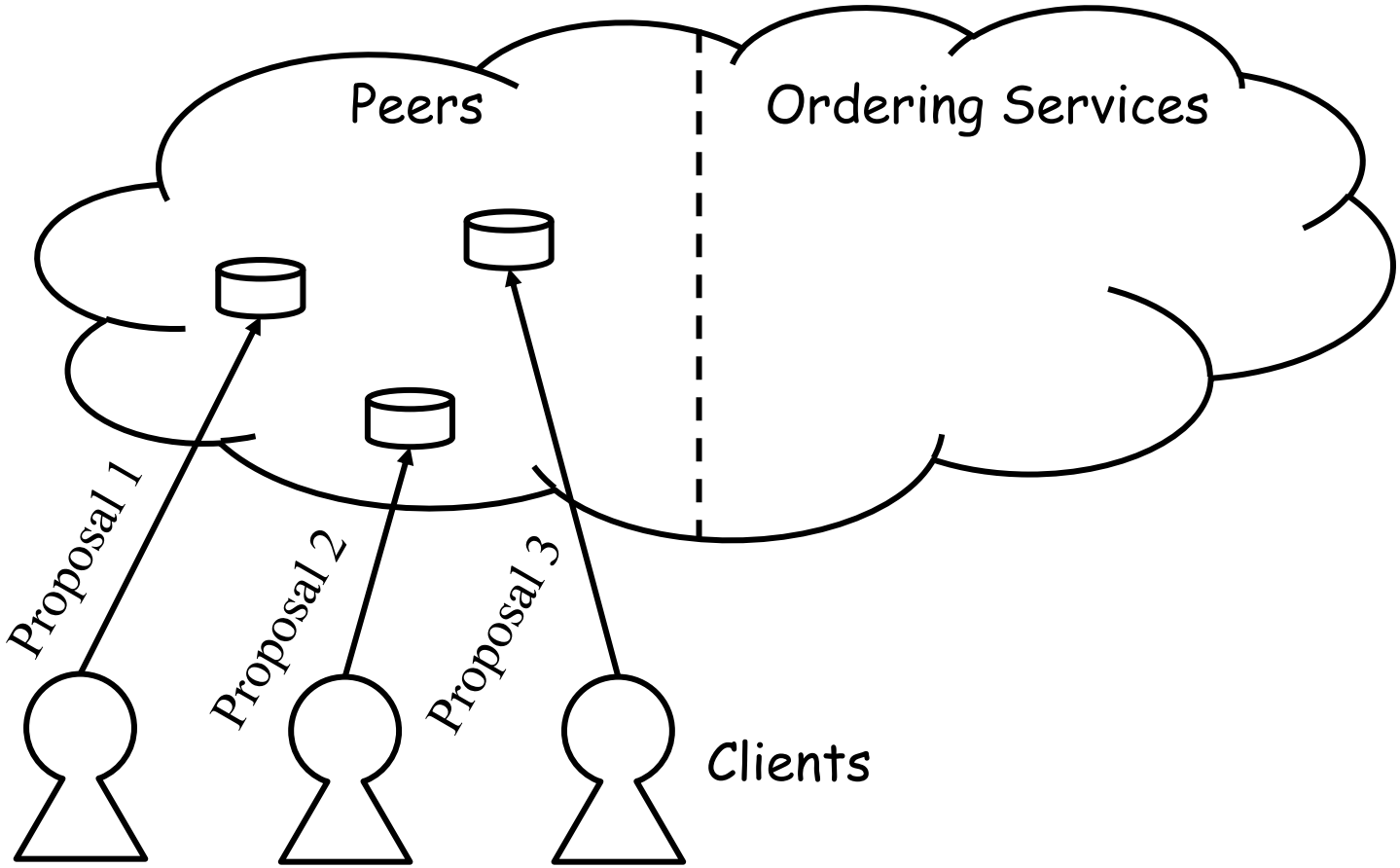
- World state: a current state for each key (maintained by key-value database, e.g., LevelDB or CouchDB)

- Blockchain: a history of transactions



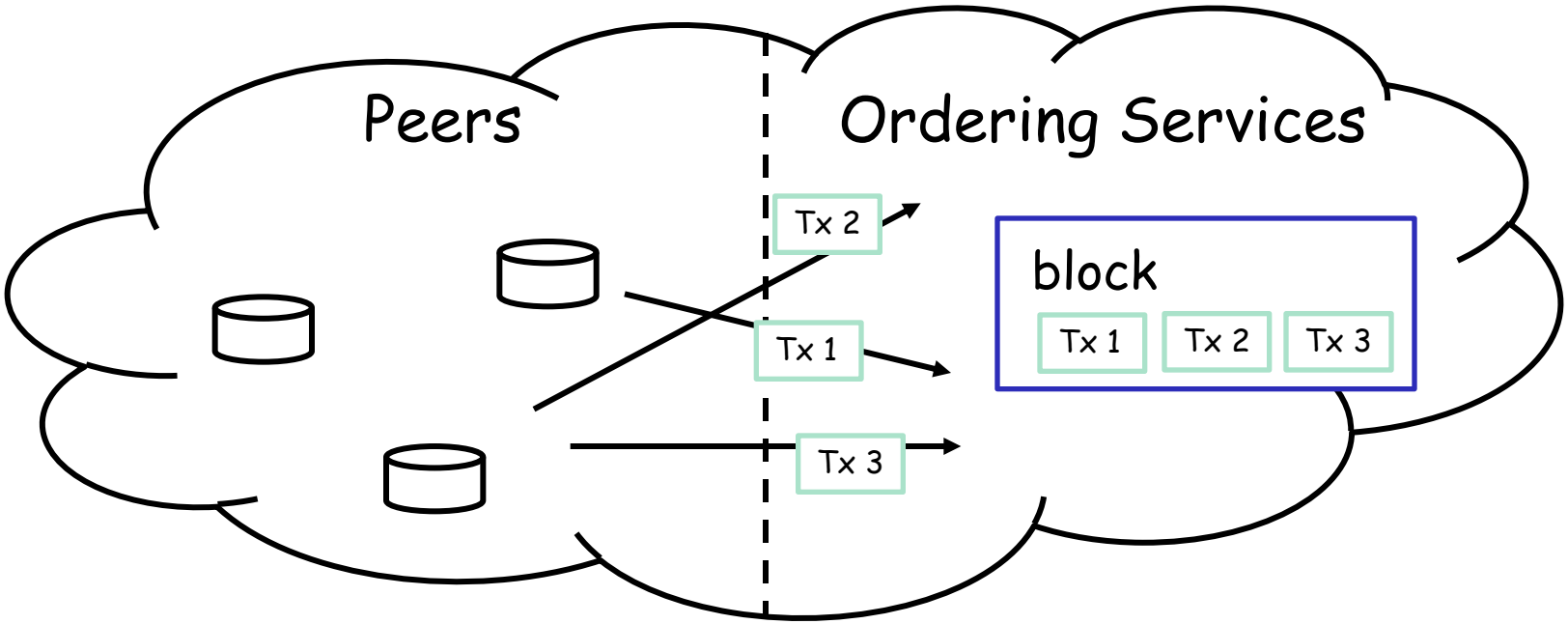
HLF architecture and components

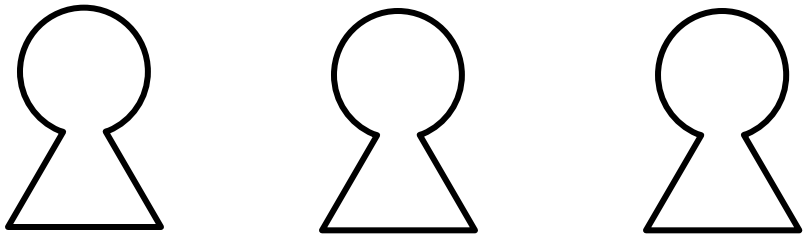
- **Peers: nodes maintain the state of the ledger and manage chaincode (i.e., smart contract)**



HLF architecture and components

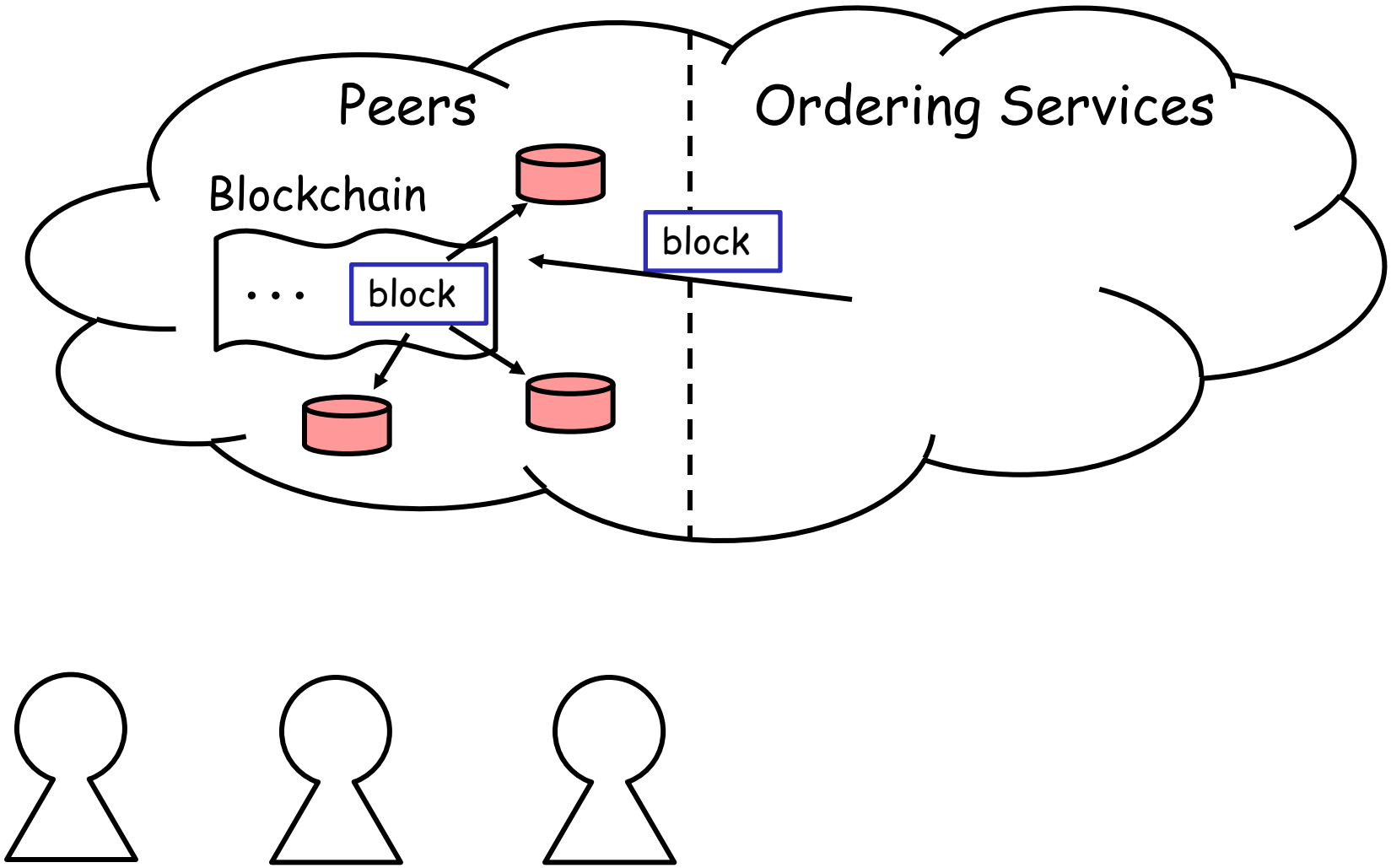
- **Ordering Services: packages transactions into blocks to be delivered to peers**





HLF architecture and components

- **Ordering Services: packages transactions into blocks to be delivered to peers**



Prepare Docker images containing HLF (1/2)

- **Prerequisites**
  - Go, Git (installed), cURL (installed), Docker

- **Install Go language**

```
$ wget https://dl.google.com/go/go1.15.2.linux-amd64.tar.gz
$ sudo tar -xvf go1.15.2.linux-amd64.tar.gz
$ sudo mv go /usr/local
$ echo export GOROOT=/usr/local/go >> $HOME/.profile

$ mkdir $HOME/gopath
$ echo export GOPATH=\$HOME/gopath >> $HOME/.profile

$ echo export PATH=\$GOPATH/bin:\$GOROOT/bin:\$PATH >> $HOME/.profile

$ . $HOME/.profile

$ go version                                  # check go language successfully installed
```

Prepare Docker images containing HLF (2/2)

- https://hyperledger-fabric.readthedocs.io/en/latest/install.html

- **Install Docker compose and set privilege**

```
$ sudo apt install docker-compose
$ sudo usermod -aG docker $USER
$ sudo chmod 666 /var/run/docker.sock
```

- **Install Samples, Binaries, and Docker Images of HLF**

```
$ cd $HOME
$ git clone https://github.com/hyperledger/fabric-samples
$ curl -sSL https://bit.ly/2ysbOFE | bash -s

$ echo export PATH=\$HOME/fabric-samples/bin:\$PATH >> $HOME/.profile
$ echo export FABRIC_CFG_PATH=\$HOME/fabric-samples/config >> $HOME/.profile

$ . $HOME/.profile
```

Building Fabric Network (1/3)

- https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html

- **Bring up the test network**

  $ cd fabric-samples/test-network

  $ ./network.sh up

  # network composition = {Org1:{peer0}, Org2:{peer0}}

```
Creating network "net_test" with the default driver
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating orderer.example.com    ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
CONTAINER ID        IMAGE                               COMMAND             CREATED             STATUS              PORTS                                   NAMES
8d0c74b9d6af        hyperledger/fabric-orderer:latest   "orderer"           4 seconds ago       Up Less than a second   0.0.0.0:7050->7050/tcp                  orderer.example.com
ea1cf82b5b99        hyperledger/fabric-peer:latest      "peer node start"   4 seconds ago       Up Less than a second   0.0.0.0:7051->7051/tcp                  peer0.org1.example.com
cd8d9b23cb56        hyperledger/fabric-peer:latest      "peer node start"   4 seconds ago       Up 1 second             7051/tcp, 0.0.0.0:9051->9051/tcp        peer0.org2.example.com
```

- **Stop the test network**

  $ ./network.sh down

Building Fabric Network (2/3)

- Creating a channel

  $ ./network.sh createChannel

- Start a chaincode on the channel

  $ ./network.sh deployCC

- Note that you can find the detail of parameters in scripts
  - network.sh, scripts/createChannel.sh, scripts/deployCC.sh

- You can find source codes of the basic chaincode
  - ../asset-transfer-basic

Detail of network.sh

```
network.sh <Mode> [Flags]
  Modes:
    up - Bring up Fabric orderer and peer nodes. No channel is created
    up createChannel - Bring up fabric network with one channel
    createChannel - Create and join a channel after the network is created
    deployCC - Deploy a chaincode to a channel (defaults to asset-transfer-basic)
    down - Bring down the network

  Flags:
  Used with network.sh up, network.sh createChannel:
  -ca <use CAs> -  Use Certificate Authorities to generate network crypto material
  -c <channel name> - Name of channel to create (defaults to "mychannel")
  -s <dbtype> - Peer state database to deploy: goleveldb (default) or couchdb
  -r <max retry> - CLI times out after certain number of attempts (defaults to 5)
  -d <delay> - CLI delays for a certain number of seconds (defaults to 3)
  -i <imagetag> - Docker image tag of Fabric to deploy (defaults to "latest")
  -cai <ca_imagetag> - Docker image tag of Fabric CA to deploy (defaults to "latest")
  -verbose - Verbose mode

  Used with network.sh deployCC
  -c <channel name> - Name of channel to deploy chaincode to
  -ccn <name> - Chaincode name. This flag can be used to deploy one of the asset transfer samples to a
 channel. Sample options: basic (default),ledger, private, sbe, secured
  -ccl <language> - Programming language of the chaincode to deploy: go (default), java, javascript, t
ypescript
  -ccv <version>  - Chaincode version. 1.0 (default), v2, version3.x, etc
  -ccs <sequence>  - Chaincode definition sequence. Must be an integer, 1 (default), 2, 3, etc
  -ccp <path>  - (Optional) File path to the chaincode. When provided, the -ccn flag will be used only
 for the chaincode name.
  -ccep <policy>  - (Optional) Chaincode endorsement policy using signature policy syntax. The default
 policy requires an endorsement from Org1 and Org2
  -cccg <collection-config>  - (Optional) File path to private data collections configuration file
  -cci <fcn name>  - (Optional) Name of chaincode initialization function. When a function is provided
, the execution of init will be requested and the function will be invoked.
```

```
Possible Mode and flag combinations
  up -ca -r -d -s -i -cai -verbose
  up createChannel -ca -c -r -d -s -i -cai -verbose
  createChannel -c -r -d -verbose
  deployCC -ccn -ccl -ccv -ccs -ccp -cci -r -d -verbose
```

A chaincode example of asset-transfer-basic

- https://github.com/hyperledger/fabric-samples/tree/master/asset-transfer-basic/chaincode-go

- **Chaincode directory**

```
├── assetTransfer.go
├── chaincode
│   ├── mocks
│   │   ├── chaincodestub.go
│   │   ├── statequeryiterator.go
│   │   └── transaction.go
│   ├── smartcontract.go
│   └── smartcontract_test.go
├── go.mod
└── go.sum
```

- **In assetTransfer.go**

```go
func main() {
	assetChaincode, err := contractapi.NewChaincode(&chaincode.SmartContract{})
	if err != nil {
		log.Panicf("Error creating asset-transfer-basic chaincode: %v", err)
	}

	if err := assetChaincode.Start(); err != nil {
		log.Panicf("Error starting asset-transfer-basic chaincode: %v", err)
	}
}
```

Deploying a smart contract to a channel (1/6)

- https://hyperledger-fabric.readthedocs.io/en/release-2.2/commands/peerchaincode.html

- **Register environment variables as a client of HLF**

  $ echo export PATH=\$HOME/fabric-samples/bin:\$PATH >> $HOME/.env_client

  $ echo export FABRIC_CFG_PATH=\$HOME/fabric-samples/config/ >> $HOME/.env_client

  $ echo export CORE_PEER_MSPCONFIGPATH=\$HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp >> $HOME/.env_client

- **Package chaincode code**

  $ source $HOME/.env_client # Swap environment

  $ cd $HOME/fabric-samples/test-network

  $ peer lifecycle chaincode package chaincode.tar.gz --path ../asset-transfer-basic/chaincode-go/ --lang golang --label sample_1.0

Deploying a smart contract to a channel (2/6)

- **Register environment variables as Org1 of HLF**

  $ echo export CORE_PEER_TLS_ENABLED=true >> $HOME/.env_org1

  $ echo export CORE_PEER_LOCALMSPID=\"Org1MSP\" >> $HOME/.env_org1

  $ echo export CORE_PEER_TLS_ROOTCERT_FILE=\$HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt >> $HOME/.env_org1

  $ echo export CORE_PEER_MSPCONFIGPATH=\$HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp >> $HOME/.env_org1

  $ echo export CORE_PEER_ADDRESS=localhost:7051 >> $HOME/.env_org1

- **Install chaincode to Org1**

  $ source $HOME/.env_org1 # Swap environment

  $ peer lifecycle chaincode install chaincode.tar.gz

```
2020-10-23 09:49:19.216 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remote
ly: response:<status:200 payload:"\nKsample_2.0:2ae919542d36563b7eb8210fe71089085ea42a158a4b9c32da7639ef
4e51d503\022\nsample_2.0" >
2020-10-23 09:49:19.216 KST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code p
ackage identifier: sample_2.0:2ae919542d36563b7eb8210fe71089085ea42a158a4b9c32da7639ef4e51d503
```

Deploying a smart contract to a channel (3/6)

- **Register environment variables as Org2 of HLF**

  $ echo export CORE_PEER_TLS_ENABLED=true >> $HOME/.env_org2

  $ echo export CORE_PEER_LOCALMSPID=\"Org2MSP\" >> $HOME/.env_org2

  $ echo export CORE_PEER_TLS_ROOTCERT_FILE=\$HOME/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt >> $HOME/.env_org2

  $ echo export CORE_PEER_MSPCONFIGPATH=\$HOME/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp >> $HOME/.env_org2

  $ echo export CORE_PEER_ADDRESS=localhost:9051 >> $HOME/.env_org2

- **Install chaincode to Org2**

  $ source $HOME/.env_org2 # Swap environment

  $ peer lifecycle chaincode install chaincode.tar.gz

- **Find the package ID of chaincodes**

  $ peer lifecycle chaincode queryinstalled

```
kaist@cs492-c-49:~/fabric-samples/test-network$ peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: sample_1.0:ad474b00b7fa4407ba88167ed1ea210269c89aa78bd19e5775d6285d97b38994, Label: sample_1
.0
```

Deploying a smart contract to a channel (4/6)

- **Approve chaincode definition to Org1**

  $ echo export CC_PACKAGE_ID=sample_1.0:ad474b00b7fa4407ba88167ed1ea210269c89aa78bd19e5775d6285d97b38994 >> $HOME/.env_chaincode

  $ source $HOME/.env_chaincode

  $ source $HOME/.env_org1

  $ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

  $ source $HOME/.env_org2

  $ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

```
kaist@cs492-c-49:~/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost:7
050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 -
-package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/exampl
e.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
2020-10-23 10:04:24.607 KST [chaincodeCmd] ClientWait -> INFO 001 txid [4029bb535734bd63256d0bb1de2ad05
9994c9edc344c897894232402e98024ea] committed with status (VALID) at localhost:7051
```

Deploying a smart contract to a channel (5/6)

- **Check whether channel memembers have approved the same chain definition**

  $ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name simple --version 1.0 --sequence 1 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json

```
kaist@cs492-c-49:~/fabric-samples/test-network$ peer lifecycle chaincode checkcommitreadiness --channel
ID mychannel --name basic --version 1.0 --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganiz
ations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": true
        }
}
```

- **Commit the validation of chaincode by organization**

  $ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 1.0 --sequence 1 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses localhost:7051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

Deploying a smart contract to a channel (6/6)

- **Confirm that the chaincode definition has been committed to the channel**

  $ peer lifecycle chaincode querycommitted --channelID mychannel --name simple --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

```
kaist@cs492-c-49:~/fabric-samples/test-network$ peer lifecycle chaincode querycommitted --channelID myc
hannel --name basic --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/examp
le.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true
, Org2MSP: true]
```

- **Invoke the chaincode (executing a function of the chaincode)**

  $ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n simple --peerAddresses localhost:7051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}'

```
2020-10-23 20:46:25.111 KST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successf
ul. result: status:200
```

Upgrading a smart contract (1/3)

- **Package and install new chaincode**

  $ source $HOME/.env_client

  $ peer lifecycle chaincode package chaincodeV2.tar.gz --path ../asset-transfer-basic/chaincode-go/ --lang golang --label simple_2.0

  $ source $HOME/.env_org1

  $ peer lifecycle chaincode install chaincodeV2.tar.gz

  $ source $HOME/.env_org2

  $ peer lifecycle chaincode install chaincodeV2.tar.gz

- **Find the new package ID by querying our peer**

  $ peer lifecycle chaincode queryinstalled

```
kaist@cs492-c-49:~/fabric-samples/test-network$ peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: sample_1.0:ad474b00b7fa4407ba88167ed1ea210269c89aa78bd19e5775d6285d97b38994, Label: sample_
1.0
Package ID: simple_2.0:ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42, Label: simple_
2.0
```

  $ echo export NEW_CC_PACKAGE_ID=simple_2.0:ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42 >> $HOME/.env_chaincodeV2

  $ source $HOME/.env_chaincodeV2

Upgrading a smart contract (2/3)

- **Approve the new chaincode definition**

  $ source $HOME/.env_org1

  $ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 2.0 --package-id $NEW_CC_PACKAGE_ID --sequence 2 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

  $ source $HOME/.env_org2

  $ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 2.0 --package-id $NEW_CC_PACKAGE_ID --sequence 2 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

- **Check if the chaincode definition with sequence 2 is ready to be commited to the channel**

  $ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name simple --version 2.0 --sequence 2 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json

```
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": true
        }
}
```

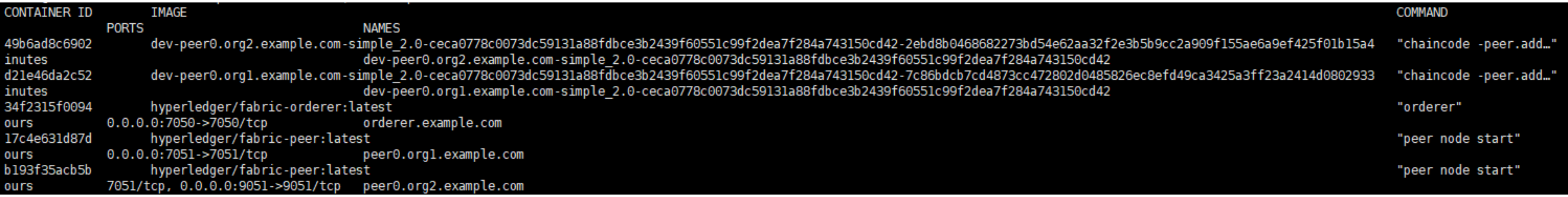
Upgrading a smart contract (3/3)

- **Commit new chaincode definition**

```
$ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name simple --version 2.0 --sequence 2 --tls --cafile $HOME/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --peerAddresses localhost:7051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles $HOME/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
```

- **Verify that the new chaincode has started on your peers**

```
$ docker ps
```



```
CONTAINER ID        IMAGE                                                                                                                                                         COMMAND
              PORTS                              NAMES
49b6ad8c6902        dev-peer0.org2.example.com-simple_2.0-ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42-2ebd8b0468682273bd54e62aa32f2e3b5b9cc2a909f155ae6a9ef425f01b15a4   "chaincode -peer.add…"
inutes                                           dev-peer0.org2.example.com-simple_2.0-ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42
d21e46da2c52        dev-peer0.org1.example.com-simple_2.0-ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42-7c86bdcb7cd4873cc472802d0485826ec8efd49ca3425a3ff23a2414d0802933   "chaincode -peer.add…"
inutes                                           dev-peer0.org1.example.com-simple_2.0-ceca0778c0073dc59131a88fdbce3b2439f60551c99f2dea7f284a743150cd42
34f2315f0094        hyperledger/fabric-orderer:latest                                                                                                                             "orderer"
ours          0.0.0.0:7050->7050/tcp             orderer.example.com
17c4e631d87d        hyperledger/fabric-peer:latest                                                                                                                                "peer node start"
ours          0.0.0.0:7051->7051/tcp             peer0.org1.example.com
b193f35acb5b        hyperledger/fabric-peer:latest                                                                                                                                "peer node start"
ours          7051/tcp, 0.0.0.0:9051->9051/tcp   peer0.org2.example.com
```

Writing own chaincode (1/3)

- **Make a directory workspace and a source file in the workspace**

```
$ mkdir simpleChaincode
$ cd simpleChaincode
$ touch main.go
```

- **Declare the package name in main.go**

```
package main
```

- **Import fabric-chaincode packages**

```
import (
    "github.com/hyperledger/fabric-chaincode-go/shim"
    "github.com/hyperledger/fabric-protos-go/peer"
)
```

- **Declare a structure as an object of chaincode**

```
type CC struct { // important: public method should start with capital letter
}
```

- **Declare two methods for *Init() and Invoke()***

```
func (c *CC) Init(stub shim.ChaincodeStubInterface) peer.Response { … }
func (c *CC) Invoke(stub shim.ChaincodeStubInterface) peer.Response { … }
```

Writing own chaincode (2/3)

```
// Main function

func main() {
   err := shim.Start(new(CC))
   if err != nil {
      panic(err.Error())
   }
   fmt.Println("Start simple chaincode now")
}
```

Writing own chaincode (3/3)

- **Manage library dependencies using go module**

  $ cd simpleChaincode

  $ go mod init simpleChaincode

  $ go mod vendor

  $ cat go.mod

```
kaist@cs492-c-49:~/workspace/simpleChaincode$ cat go.mod
module simpleChaincode

go 1.15

require (
        github.com/hyperledger/fabric-chaincode-go v0.0.0-20200728190242-9b3ae92d8664
        github.com/hyperledger/fabric-protos-go v0.0.0-20200923192742-3897341ac036
)
```

- **Make a binary file from main.go**

  $ go build

```
kaist@cs492-c-49:~/workspace/simpleChaincode$ ls
go.mod  go.sum  main.go  simpleChaincode  vendor
```

Homework 2. An example of a bank transfer system

- **Write a chaincode for a bank transfer system and deploy the chaincode in fabric network**

- **Two functionalities should be supported as following:**

  (1) send money from A to B
  e.g., {send a b 10}
  => aVal-=10; bVal+=10

  (2) withdraw money with fee
  e.g., {withdraw a 20 0.05}
  => aVal -= 20 * (1+0.05); bankVal += 20 * 0.05

- **All errors should be detected for accessing state DB**
  - e.g., lack of bank balance, no element in state DB

- **The chaincode initializes that four person have 100$ and bank have 1000$**

An example of a bank transfer system (cont.)

```
func (c *CC) Invoke(stub shim.ChaincodeStubInterface) peer.Response {
  var f, args = stub.GetFunctionAndParameters()
  switch f {
    case "send" :
      money, _ := strconv.Atoi(args[0])              //import "strconv"
      aValBytes, _ :=  stub.GetState(args[1])
      bValBytes, _ := stub.GetState(args[2])
      aVal, _ := strconv.Atoi(string(aValBytes))
      bVal, _ := strconv.Atoi(string(bValBytes))
      aVal -= money
      bVal += money
      aValByte = []byte(strconv.Itoa(aVal))
      bValByte = []byte(strconv.Itoa(bVal))
      stub.PutState(args[1], aValByte)
      stub.PutState(args[2], bValByte)
      return shim.Success([]byte(("OK"))          // to debug, return updated key-value pairs here
    case "withdraw" :
...
  return shim.Error(fmt.Sprintf("Error: %s not found", f))
                                                  // if there was no action above, return error
}
```

Homework 3. Solving issues with transaction conflict

- **If there are many request for withdrawing money, the bank value is updated simultaneously, and it causes transaction conflict**
  - i.e., {withdraw A 20 0.05}, {withdraw B 30 0.05}, {withdraw C 40 0.05}

    => aVal -= 20 * (1+0.05); bankVal += 20 * 0.05
    bVal -= 30 * (1+0.05); bankVal += 30 * 0.05
    cVal -= 40 * (1+0.05); bankVal += 40 * 0.05

    Transaction conflict occurs!

- **It is due to access same key-value on bank account at the same time**

- **Solve the issues with massive transactions of "Withdraw"**
  - Hint: Delta computation on bank value

- **Detail of chaincode interface are here,** https://github.com/hyperledger/fabric-chaincode-go/blob/master/shim/interfaces.go

Solving issues with transaction conflict (cont.)

- https://github.com/hyperledger/fabric-samples/blob/master/high-throughput

```
func (s *SmartContract) getStandard(APIstub shim.ChaincodeStubInterface, args []string) pb.Response {
	name := args[0]

	val, getErr := APIstub.GetState(name)
	if getErr != nil {
		return shim.Error(fmt.Sprintf("Failed to get state: %s", getErr.Error()))
	}

	return shim.Success(val)
}
```

Solving issues with transaction conflict (cont.)

```
func (s *SmartContract) get(APIstub shim.ChaincodeStubInterface, args []string) pb.Response {
	name := args[0]
	deltaResultsIterator, deltaErr := APIstub.GetStateByPartialCompositeKey("varName~op~value~txID", []string{name})
	defer deltaResultsIterator.Close()

	var finalVal float64 // Iterate through result set and compute final value
	var i int
	for i = 0; deltaResultsIterator.HasNext(); i++ {
		responseRange, nextErr := deltaResultsIterator.Next() // Get the next row
		_, keyParts, splitKeyErr := APIstub.SplitCompositeKey(responseRange.Key)  // Split the composite key

		operation := keyParts[1]
		valueStr := keyParts[2]

		value, convErr := strconv.ParseFloat(valueStr, 64) // Convert the value string
		switch operation {
		case "+":
			finalVal += value
		case "-":
			finalVal -= value
		default:
			return shim.Error(fmt.Sprintf("Unrecognized operation %s", operation))
		}
	}

	return shim.Success([]byte(strconv.FormatFloat(finalVal, 'f', -1, 64)))
}
```