DConf 2018
Munich

D for the Blockchain

Kai Nacke
kai@redstar.de

2 May 2018

Agenda

What is a blockchain, anyway?

Examine the Hyperledger Fabric project

The case for D

Summary

Agenda

What is a blockchain, anyway?

Examine the Hyperledger Fabric project

The case for D

Summary

What is a blockchain, anyway?

“A shared replicated, permissioned ledger with consensus, provenance, immutability and finality.”

WOW!

Source: https://developer.ibm.com/courses/wp-content/uploads/sites/83/BlockchainOverview.pdf, p. 8

Building blocks of a blockchain

- Shared
- Replicated
- Permissioned

Shared Ledger

Smart Contract

- Business rules
- Executed in transaction
- Encode in a programming language

- Participants require confidentiality
- Identity not linked to transaction
- Transactions are authenticated

Privacy

Trust

- Endorsed by participants
- Verifiable
- Transactions cannot be modified, inserted or deleted

Source: https://developer.ibm.com/courses/wp-content/uploads/sites/83/BlockchainOverview.pdf, p. 10

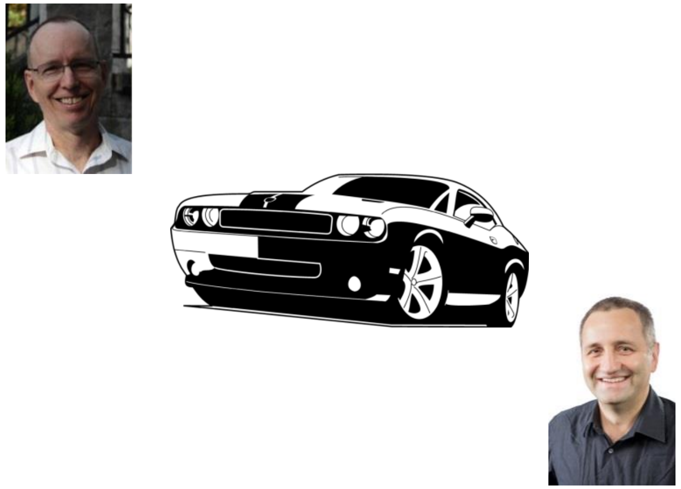
Example: manage car ownership (1)

- Assets managed by ledger are cars
  - Has attributes like model, color, …

- Participants can be
  - Car producers
  - Car owners
  - Insurance companies
  - Banks
  - Car disposal companies

Example: manage car ownership (2)

- Sample transactions
  - A well-known company from Bavaria produces a car
  - Walter gives his Mustang to Andrei



- Sample contracts
  - If Scott receives money from Andrei then ownership of Scott's car passes to Andrei
  - If the car won't start (verified by a third party) then Scott will receive no money

Agenda

What is a blockchain, anyway?

Examine the Hyperledger Fabric project

The case for D

Summary

Examine the Hyperledger Fabric project

- Hyperledger is an umbrella project of The Linux Foundation
- Hosts and promotes Open Source Business Blockchain Frameworks
- Hyperledger Fabric is an implementation of blockchain technology intended for developing blockchain applications or solutions

Elements of Fabric (1)

HYPERLEDGER FABRIC

Peer

- Peers host the ledger and the chaincode
- Every member can run one or more peers
- Endorses transactions
- Applications talk to peers

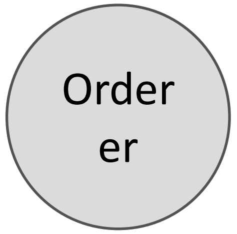
Chaincode

- Implementation of the Smart Contract (chaincode)

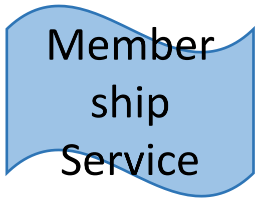
Ledger

- Holds the state of the blockchain

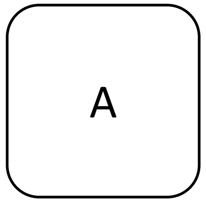
Elements of Fabric (2)



- Receives endorsed transactions from peers
- Orders and packs transactions into blocks
- Distributes blocks back to peers for ledger update

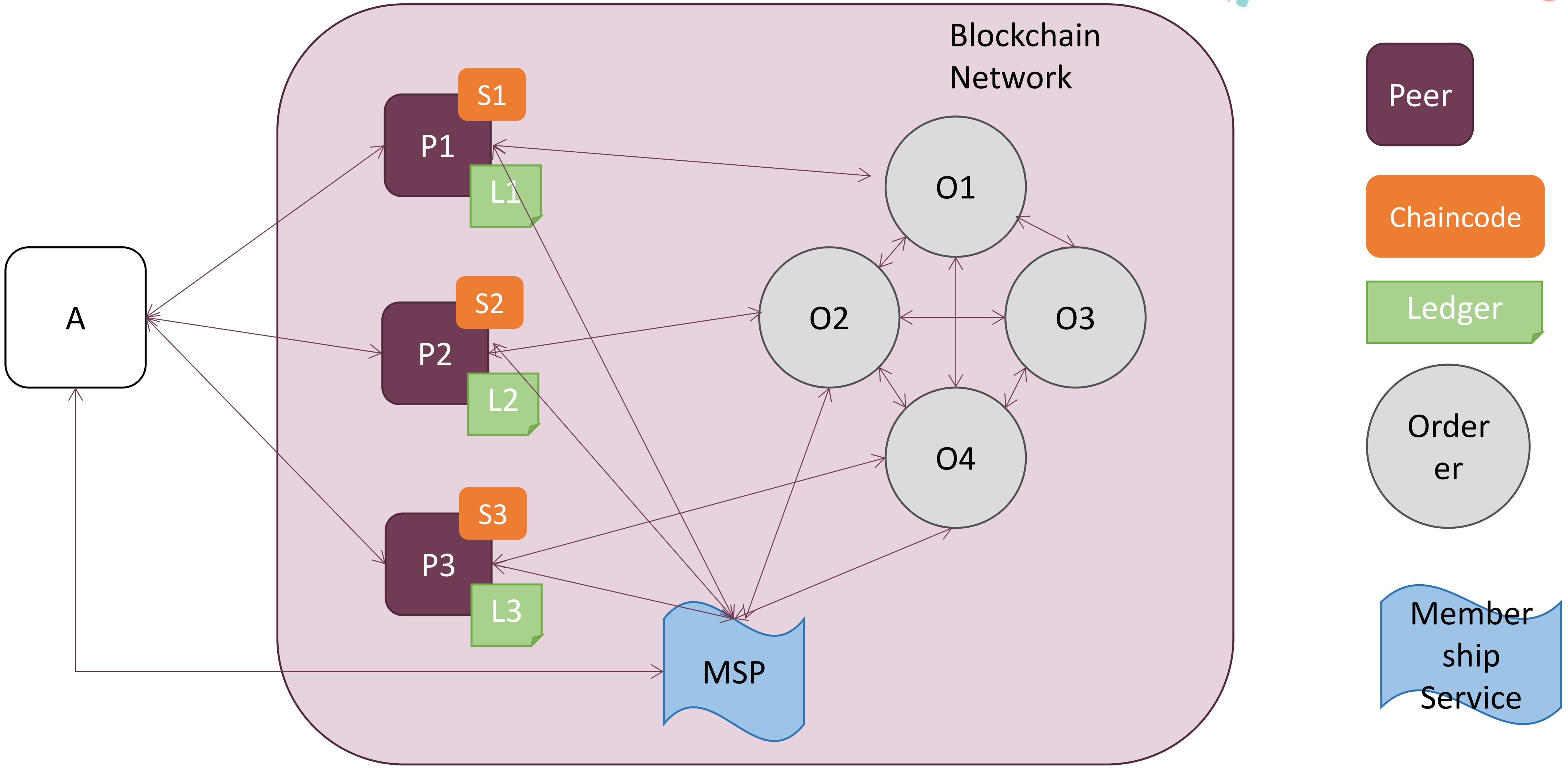


- Defines which Root CA is trusted
- Provides certificates for TLS, revocation lists, …



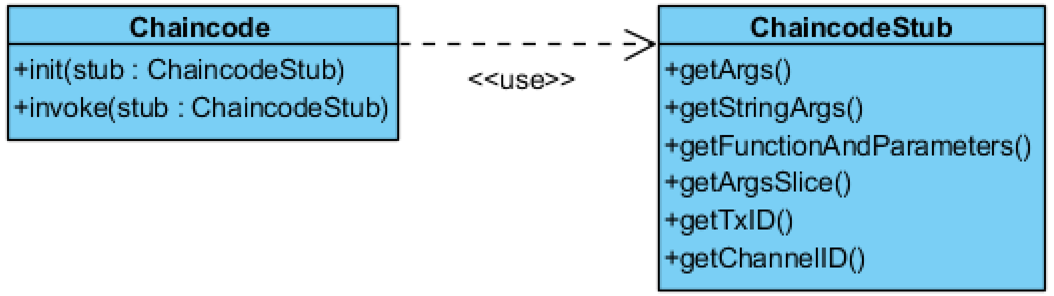
- Application using the blockchain

Architectural view of Fabric



Some technical details

- Hyperledger Fabric uses gRPC as communication protocol
- Data is exchanged in JSON format
- The framework is written in Go
- The API consists of two interfaces



Agenda

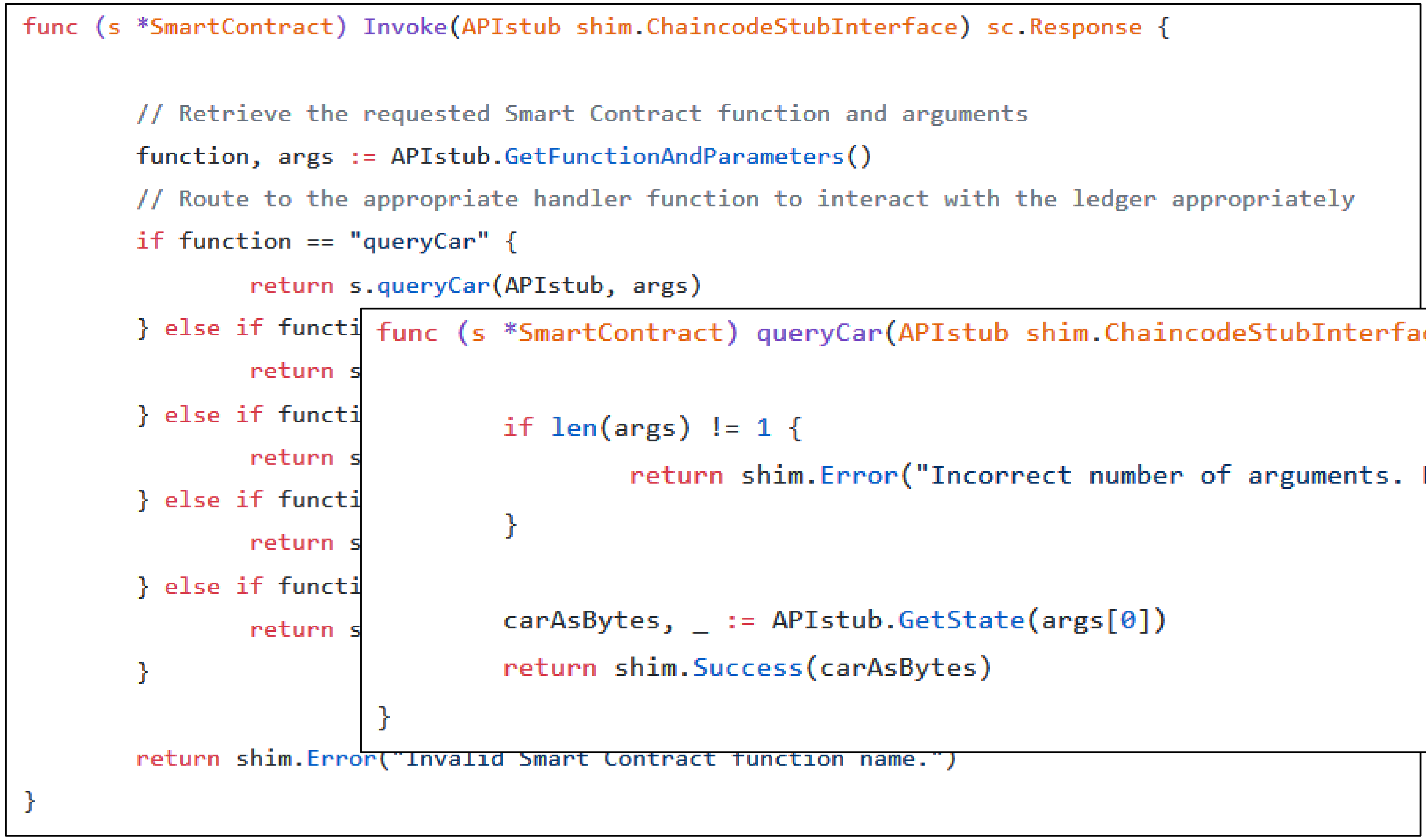
What is a blockchain, anyway?

Examine the Hyperledger Fabric project

The case for D

Summary

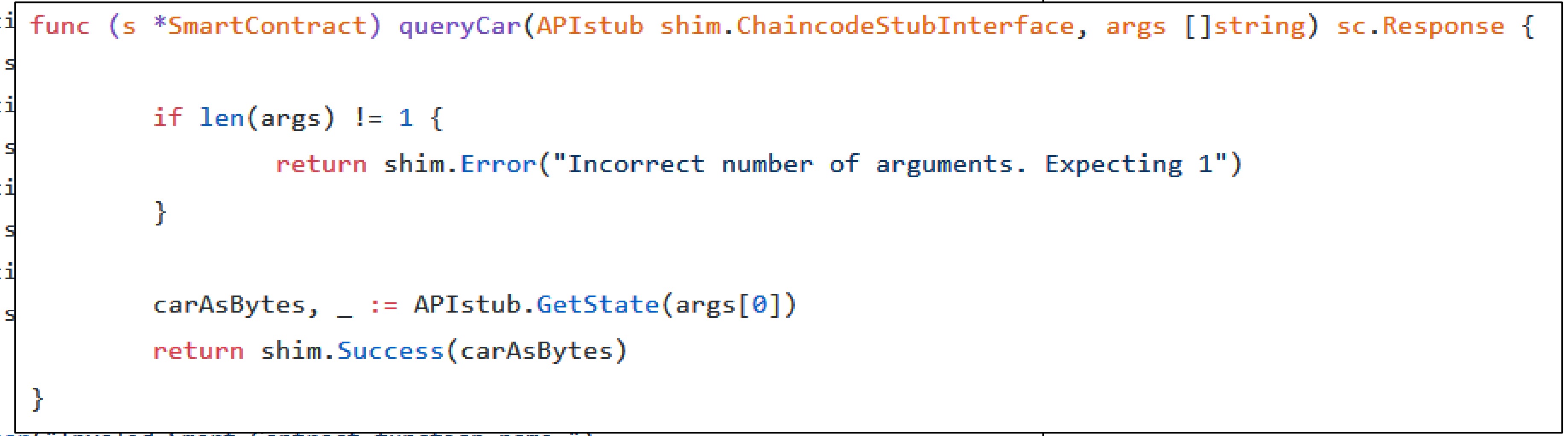
Let’s look at fabcar sample in Go…



```
func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {

	// Retrieve the requested Smart Contract function and arguments
	function, args := APIstub.GetFunctionAndParameters()
	// Route to the appropriate handler function to interact with the ledger appropriately
	if function == "queryCar" {
		return s.queryCar(APIstub, args)
	} else if functi
		return s
	} else if functi
		return s
	} else if functi
		return s
	} else if functi
		return s
	}

	return shim.Error("Invalid Smart Contract function name.")
}
```



```
func (s *SmartContract) queryCar(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

	if len(args) != 1 {
		return shim.Error("Incorrect number of arguments. Expecting 1")
	}

	carAsBytes, _ := APIstub.GetState(args[0])
	return shim.Success(carAsBytes)
}
```

Source: https://github.com/hyperledger/fabric-samples/blob/release-1.1/chaincode/fabcar/go/fabcar.go -

… and in D

```
// Written in the D programming language.
module fabcar;

import fabric.shim.chaincode;
import fabric.shim.response;

class Chaincode : DefaultChaincode
{
    mixin InvokeHelper!Chaincode;

    @SmartContract
    Response queryCar(ChaincodeStub stub, string arg)
    {
        auto carAsBytes = stub.getState(arg);
        return success(carAsBytes);
    }
}
```

- Sample code in Go has lots of boilerplate code
- Go has no templates to hide such stuff
- D reflection, attributes, templates, mixins and CTFE can help here
- Who is this Go, anyway?

How to talk to the blockchain

- gRPC is used for communication
- On the D side
  - There is no official D implementation of gRPC
  - A project exists, but it seems to be dead
  - Several protobuf implementations exists
- Could be implemented in D but lot of effort required

- For client side there exists a REST server
  - Use vibe.d to talk over the REST server to the blockchain

Interface D and Go?

- Go has a C interface
- It is limited
  - You cannot pass pointers to Go objects to C code
  - Difficult to implement ChaincodeStub callback interface
- Go has a garbage collector
  - Requires tweeking of D garbage collector (if possible at all)
  - Go runtime cannot be used because of restriction above
- Seems possible to use with `–betterC`
  - Goal is to use D!

And now?

	Pro	Contra
Implement in D, including missing libraries like grpc	• Enables D-only code • Useful for other D projects	• Lot of effort for non-blockchain code
Integrate with Go	• Fast approach	• Only limited D support • Full support unclear
?		

And now?

	Pro	Contra
Implement in D, including missing libraries like grpc	• Enables D-only code • Useful for other D projects	• Lot of effort for non-blockchain code
Integrate with Go	• Fast approach	• Only limited D support • Full support unclear
Implement base code in C++ and define D interface for it	• Useful for other projects • Limitations of C++/D interface well known	• Lot of C++ coding

Approaches not evaluated

There are tools available which allows to use C header files with D

- dstep by Jacob Carlborg
- dpp by Atila Neves

I did not look at this approach because

- it introduces yet another critical tool dependency
- a C++ version seemed very useful to me

But I am curious to try out these tools!

What is already working

- D interface for chaincode is defined
- C++ interface for chaincode is defined
- Fabcar sample is translated
- Registering of chaincode at peer works

Network protocol still needs to be completed
The technological approach works

Next steps

- Complete coding
- Contribute code to Fabric project

Agenda

What is a blockchain, anyway?

Examine the Hyperledger Fabric project

The case for D

Summary

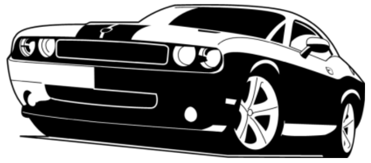
Summary

- Blockchain provides a distributed ledger
- Fabric is a popular framework for implementing blockchain applications
- D lets the developer concentrate on the business logic
- With my approach I can enjoy coding a blockchain application in D

Questions?

Image reference (1)

- PowerPoint ClipArt:



- Dconf web site: http://dconf.org/2018/index.html

- The LDC logo: https://github.com/ldc-developers/ldc#installation

Image reference (2)

- Project logos
  - Hyperledger: https://www.hyperledger.org/
  - Linux Foundation: https://www.linuxfoundation.org/
  - Hyperledger Fabric: https://www.hyperledger.org/projects/fabric