

FORWARDER-RECEIVER
CLIENT-DISPATCHER-SERVER

KOMUNIKAČNÍ NÁVRHOVÉ VZORY

Komunikace mezi

- Vlákny
- Procesy
- Server-client uzly

Vzory

- Forwarder – Receiver
- Client – Dispatcher – Server
- Publisher – Subscriber

Problémy

- Heterogenní způsob komunikace mezi účastníky
- Umístění účastníků komunikace

PROBLÉM

Kád'



Kolik je piva??



42 litrů



Monitor





```
1  class Kád {  
2      public void Listen() {  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  }  
22 }  
23
```



```
1  class Monitor {  
2      public void DisplayAmountOfPivo() {  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  }  
19 }  
20
```



```
1 class Kád {  
2     public void Listen() {  
3  
4         var connection = new Connection("142.69.42.69", 42222);  
5  
6         string msgString = connection.ReceiveMsg();  
7  
8         var msg = Json.Deserialize<Message>(msgString);  
9  
10        Response response = new Response(...);  
11  
12        if (msg.Body == "Kolik je pivka??") {  
13            response.AmountOfPivo = // read from sensors  
14        } else {  
15            // handle unknown msg ...  
16        }  
17  
18        string responseString = Json.Serialize(response);  
19  
20        connection.Send(msg.Sender, responseString);  
21    }  
22 }  
23
```

Musí znát svůj port a adresu



Znalost formátu zprávy



Musí znát adresu kádě



```
1 class Monitor {  
2     public void DisplayAmountOfPivo() {  
3  
4         var connection = new Connection("42.69.142.69", 3233);  
5         var kádLocation = new Address("142.69.42.69", 42222);  
6  
7         var msg = new Message("Kolik je pivka??");  
8  
9         string msgString = Json.Serialize(msg);  
10  
11        connection.Send(kádLocation, msgString);  
12  
13        string responseString = connection.ReceiveResponse();  
14  
15        string response = Json.Deserialize<Response>(responseString);  
16  
17        Display.Show(response.AmountOfPivo);  
18    }  
19 }  
20
```

Celková závislost na
použité doručovací
technologii





```
1 class Kád {  
2     public void Listen() {  
3  
4         var connection = new Connection("142.69.42.69", 42222);  
5  
6         string msgString = connection.ReceiveMsg();  
7  
8         var msg = Json.Deserialize<Message>(msgString);  
9  
10        Response response = new Response(...);  
11  
12        if (msg.Body == "Kolik je pivka??") {  
13            response.AmountOfPivo = // read from sensors  
14        } else {  
15            // handle unknown msg ...  
16        }  
17  
18        string responseString = Json.Serialize(response);  
19  
20        connection.Send(msg.Sender, responseString);  
21    }  
22 }  
23
```

Infrastruktura



Business logic

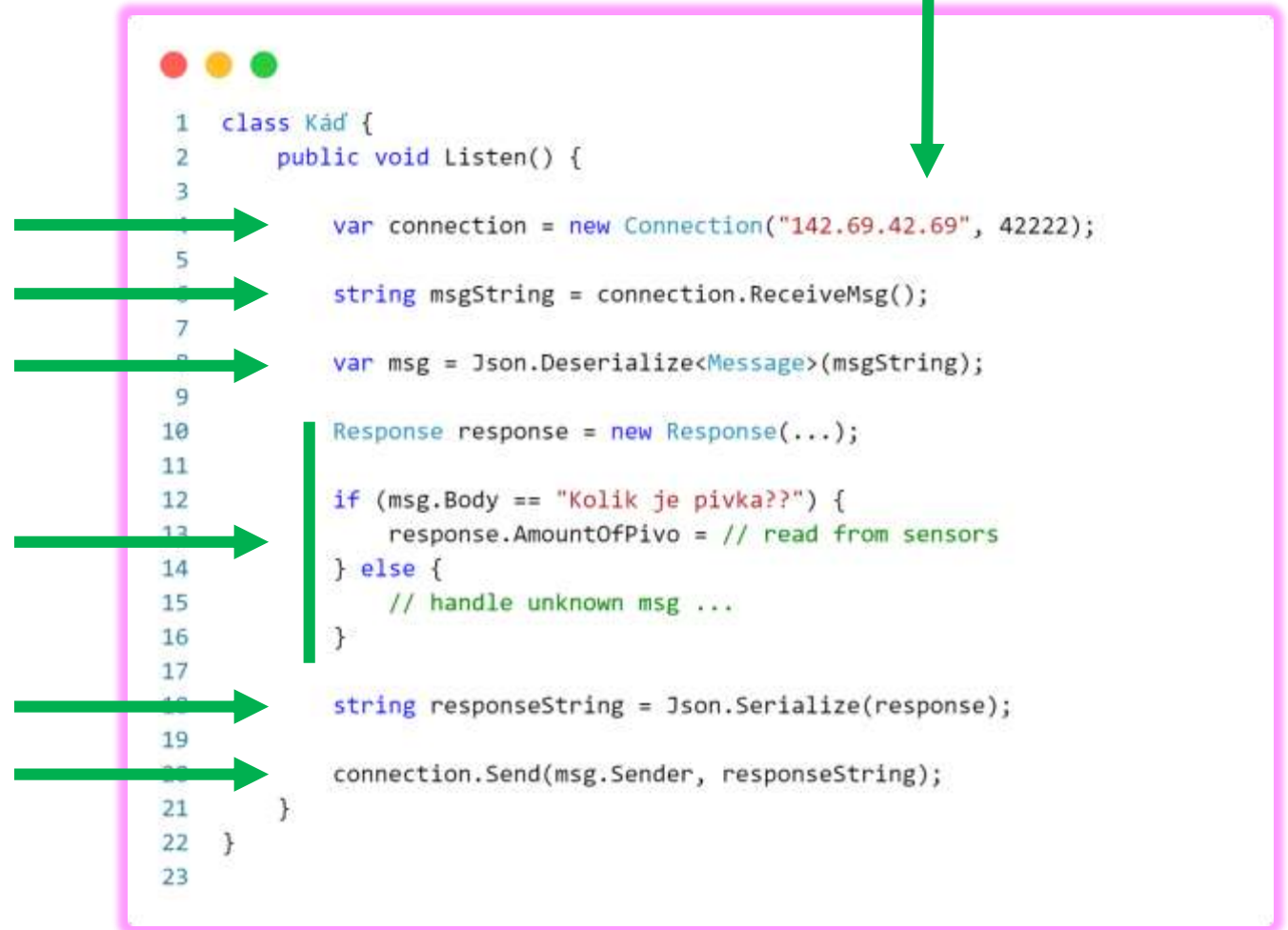


```
1 class Monitor {  
2     public void DisplayAmountOfPivo() {  
3  
4         var connection = new Connection("42.69.142.69", 3233);  
5         var kádLocation = new Address("142.69.42.69", 42222);  
6  
7         var msg = new Message("Kolik je pivka??");  
8  
9         string msgString = Json.Serialize(msg);  
10  
11        connection.Send(kádLocation, msgString);  
12  
13        string responseString = connection.ReceiveResponse();  
14  
15        string response = Json.Deserialize<Response>(responseString);  
16  
17        Display.Show(response.AmountOfPivo);  
18    }  
19 }  
20
```


JAK TO ZLEPŠIT ??

Co izolovat ??

- Logiku
 - Co znamenají zprávy
 - Jak reagovat na zprávy
- Komunikaci
 - Použitá technologie komunikace
 - Řešení problémů (timeouty, opakování requestů...)
- Serializaci / Deserializaci
 - JSON, XML, Plain text, CSV, Binary ...
- Adresaci
 - Překlad jmen na adresy



```
1 class Kád {
2     public void Listen() {
3
4         var connection = new Connection("142.69.42.69", 42222);
5
6         string msgString = connection.ReceiveMsg();
7
8         var msg = Json.Deserialize<Message>(msgString);
9
10        Response response = new Response(...);
11
12        if (msg.Body == "Kolik je pivka?") {
13            response.AmountOfPivo = // read from sensors
14        } else {
15            // handle unknown msg ...
16        }
17
18        string responseString = Json.Serialize(response);
19
20        connection.Send(msg.Sender, responseString);
21    }
22 }
23
```

The diagram illustrates the isolation of different components in the provided C# code. Green arrows point to specific lines, indicating which parts of the code are being targeted for improvement or isolation:

- A vertical arrow points to the `Listen()` method signature (lines 2-3).
- Horizontal arrows point to the following lines:
 - Line 4: `var connection = new Connection("142.69.42.69", 42222);`
 - Line 6: `string msgString = connection.ReceiveMsg();`
 - Line 8: `var msg = Json.Deserialize<Message>(msgString);`
 - Line 12: The start of the `if` statement block.
 - Line 18: `string responseString = Json.Serialize(response);`
 - Line 20: `connection.Send(msg.Sender, responseString);`



```
1 string msgString = connection.GetNextMessage();
2 string msg = Json.Deserialize<Message>(responseString);
3
4 return msg;
5
```



```
1 var msg = new Message() {
2     Sender = MyName, // name of node
3     Body = data
4 };
5
6 string msgString = Json.Serialize(msg);
7 var address = GetPhysicalAddress(recipient); // magic
8
9 connection.Send(address, msgString);
```



```
1 forwarder.SendMsg(
2     recipient: "Kád",
3     data: "Kolik je piva??"
4 );
5
```



```
1 var msg = receiver.ReceiveMsg();
2
3 if (msg.Data == "Kolik je pivka?") {
4     amountOfPivo = // read from sensors
5 }
6
7 forwarder.Send(msg.Sender, amountOfPivo);
```



```
1 var msg = receiver.ReceiveMsg();
2 Display.Show($"Amount of pivo: {msg.Data}");
3
```

Receiver

Forwarder

Forwarder

Receiver

```
{
  "sender": "Monitor",
  "body": "Kolik je pivka??"
}
```

```
{
  "sender": "Kád",
  "body": 42
}
```



```

1 class Forwarder {
2     Connection connection; //from external configuration
3
4     public void SendMsg(CanonicName recipient, object data) {
5         var msg = new Message() {
6             Sender = MyName, // name of node
7             Body = data
8         };
9
10        string msgString = Json.Serialize(msg);
11        var address = GetPhysicalAddress(recipient);
12
13        connection.Send(address, msgString);
14    }
15
16    // from external configuration e.g. file
17    HashTable nameTranslations;
18
19    private IPAddress GetPhysicalAddress(CanonicName name) {
20        return nameTranslations[name];
21    }
22 }

```

```

1 class Monitor {
2
3     // both possibly injected
4     Forwarder forwarder;
5     Receiver receiver;
6
7     public void DisplayAmountOfPivo() {
8         forwarder.SendMsg(
9             recipient: "Kád",
10            data: "Kolik je piva??"
11        );
12
13        var msg = receiver.ReceiveMsg();
14        Display.Show($"Amount of pivo: {msg.Data}");
15    }
16 }

```

```

1 class Receiver {
2     Connection connection; //from external configuration
3
4     public Message ReceiveMsg() {
5         string msgString = connection.GetNextMessage(); // blocking
6         string msg = Json.Deserialize<Message>(responseString);
7
8         return msg;
9     }
10 }

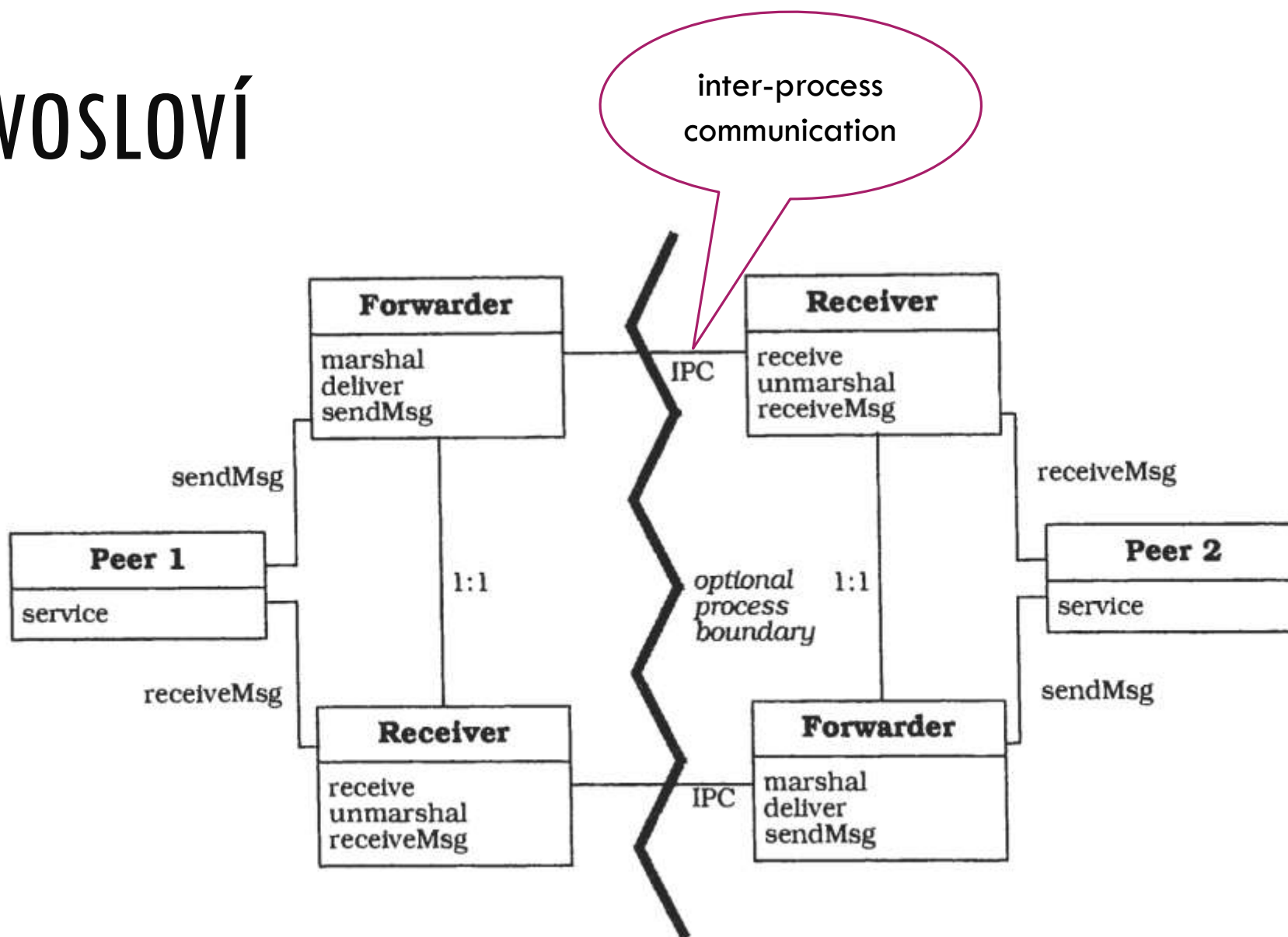
```

```

1 class Kád {
2
3     // both possibly injected
4     Forwarder forwarder;
5     Receiver receiver;
6
7     public void Listen() {
8         var msg = receiver.ReceiveMsg();
9
10        if (msg.Data == "Kolik je pivka??") {
11            amountOfPivo = // read from sensors
12        }
13
14        forwarder.Send(msg.Sender, amountOfPivo);
15    }
16 }

```

NÁZVOSLOVÍ



POZNÁMKY

Adresování

- Adresa nemusí být pro jediný uzel
- Může mít hierarchickou strukturu ("Plzeň/Kád")
- Překlad adres
 - Sdílená tabulka, každý forwarder vlastní

Druhy zpráv

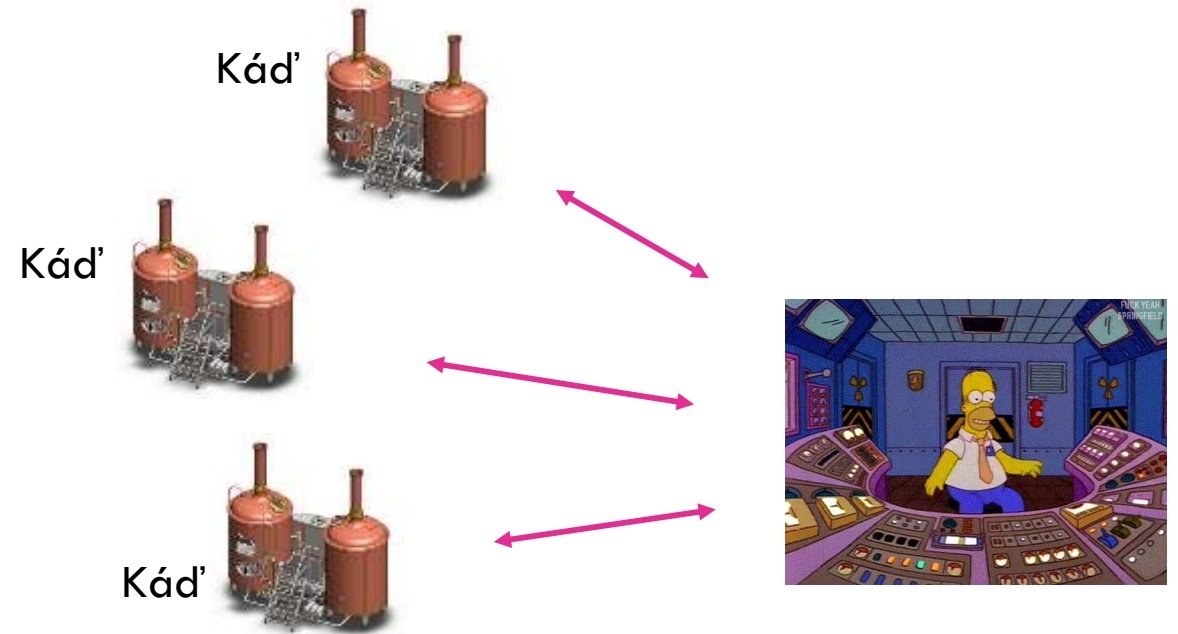
- Command, information, response

Obsluha (chyb) komunikace

- Vypršení času
- Opakování requestu
- Řídící zprávy

Implementace receiveru

- Blokuující
- Neblokuující



```
1 class Receiver {
2     Connection connection; //from external configuration
3
4     public async Task<Message> ReceiveMsg() {
5         string msgString = await connection.GetNextMessage(); // nonblocking
6         string msg = Json.Deserialize<Message>(responseString);
7
8         return msg;
9     }
10 }
```

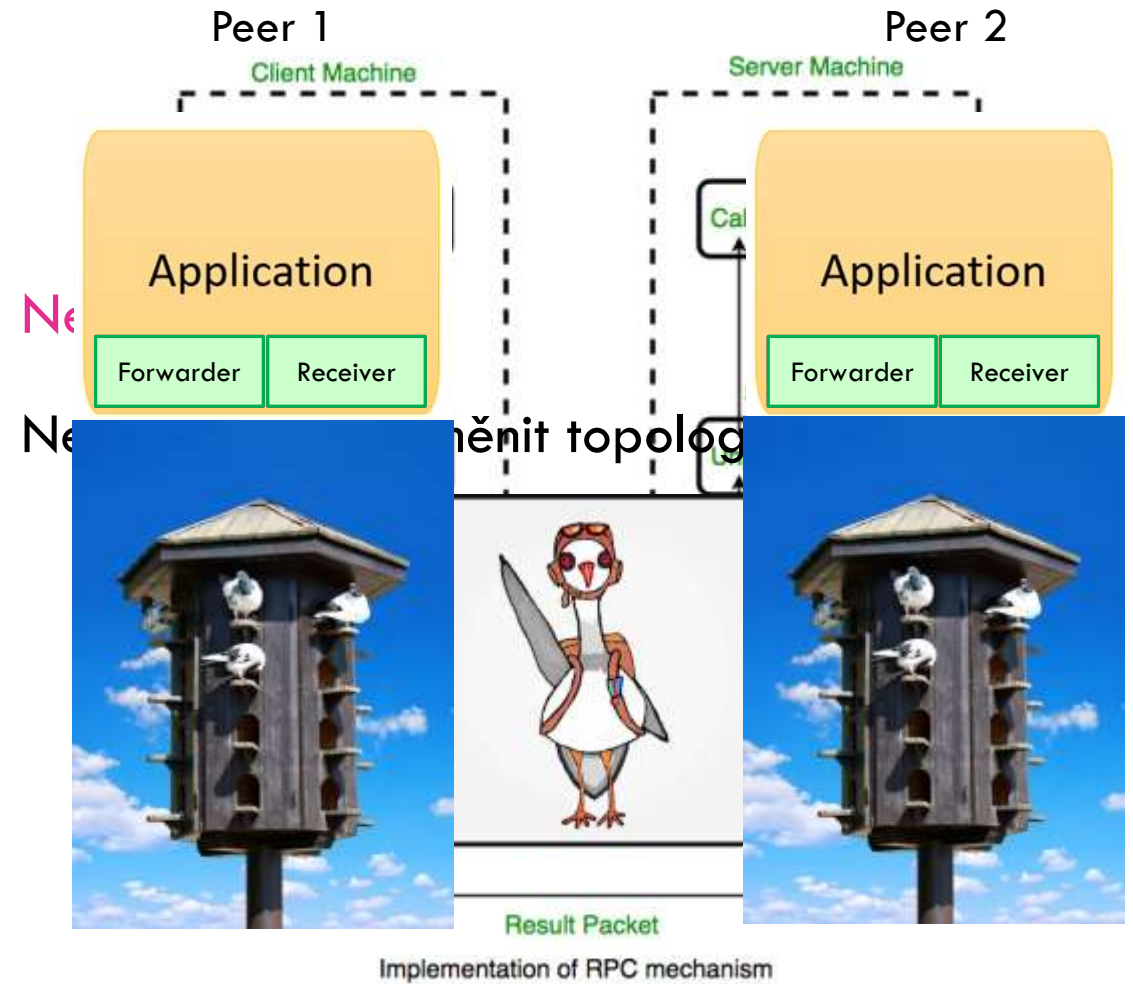
SHRNUTÍ

Výhody

Efektivita

- Minimální overhead oproti samotné komunikaci
- Rychlý překlad adres
- Lightweight oproti např. RPC

Abstrakce nad konkrétní formou komunikace



Kád' na adrese 42.42.42.42:42222



Kolik je piva??



42 litrů

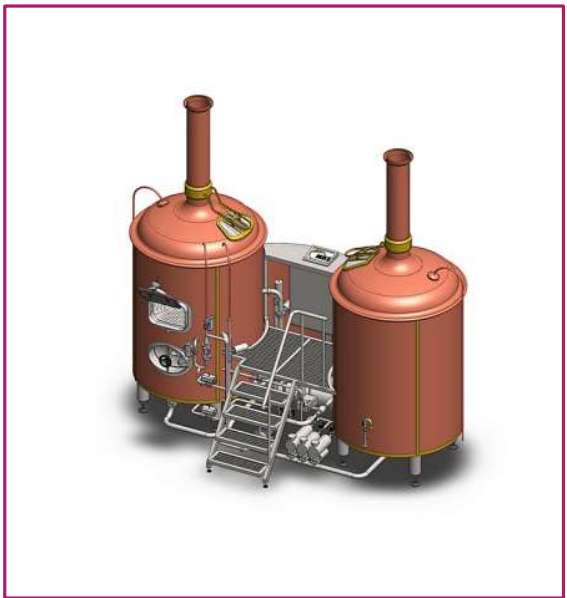


Vím na jaké
adrese Kád' je

*forwarder to ví



Kád' na adrese 42.42.42.42:42222



Kád' na adrese 69.69.69.69:42222



Změna adresy



... musí znát
adresy Monitorů



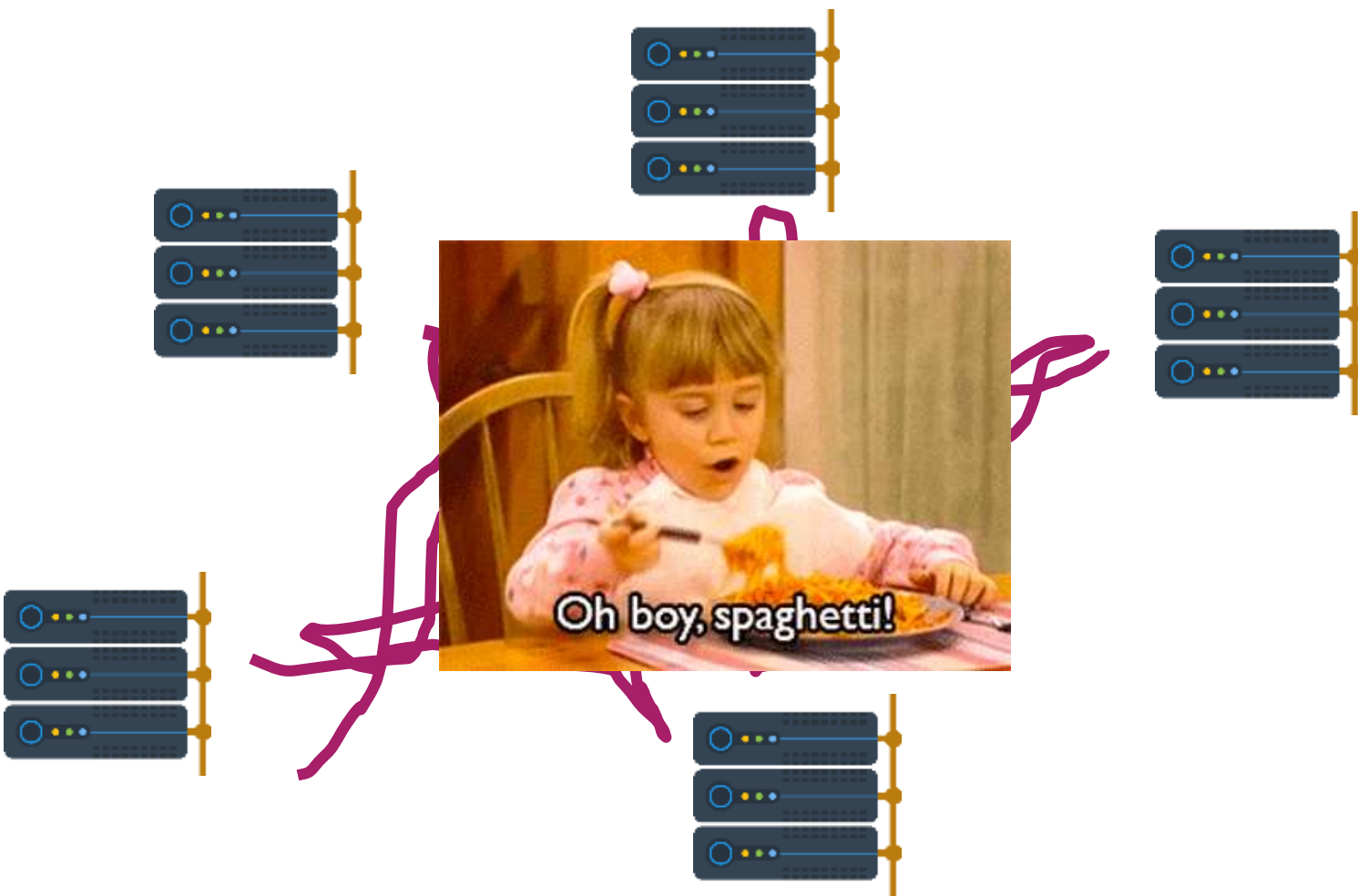
Mám novou adresu

Mám novou adresu

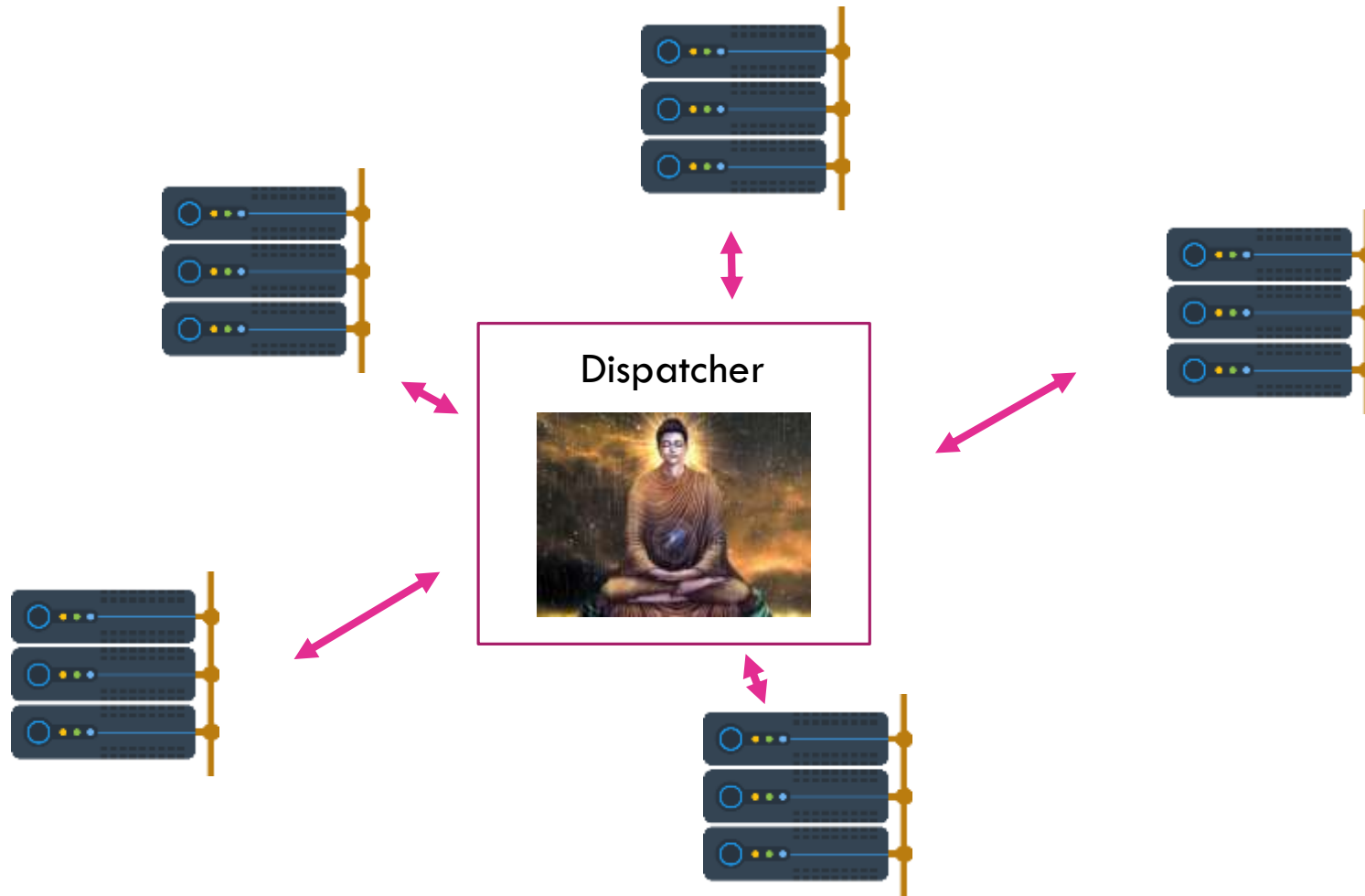
Mám novou adresu

Mám novou adresu





CLIENT - DISPATCHER - SERVER



Kád' na adrese 42.42.42.42:42222



Kolik je piva??



42 litrů



Nevím na jaké
adrese Kád' je ..
zeptám se

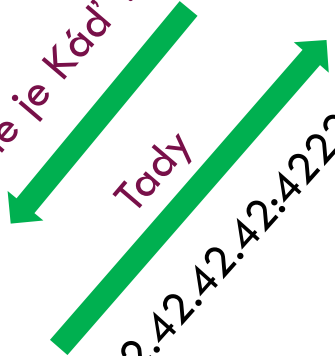
Jsem tady
42.42.42.42:42222



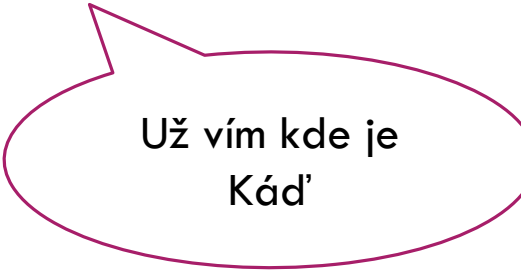
Dispatcher



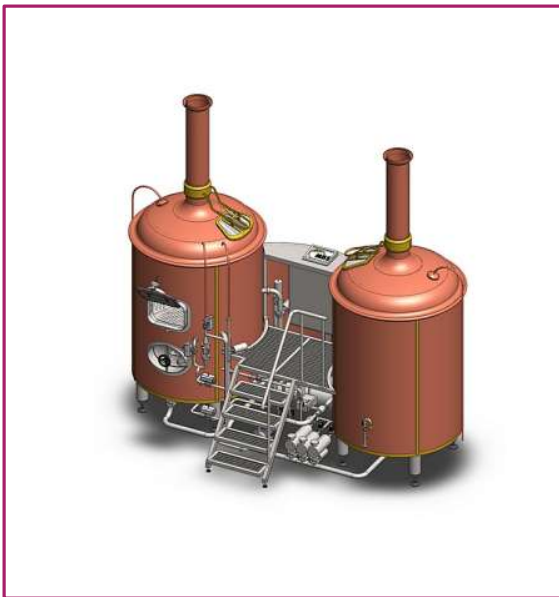
Kde je Kád' ??
Tady
42.42.42.42:42222



Už vím kde je
Kád'



Kád' na adrese 69.69.69.69:42222



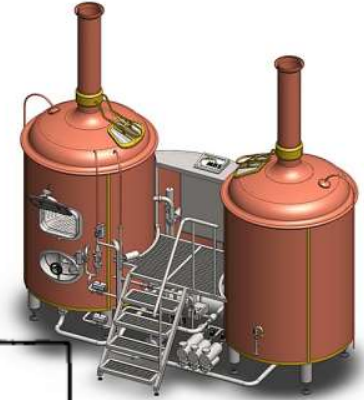
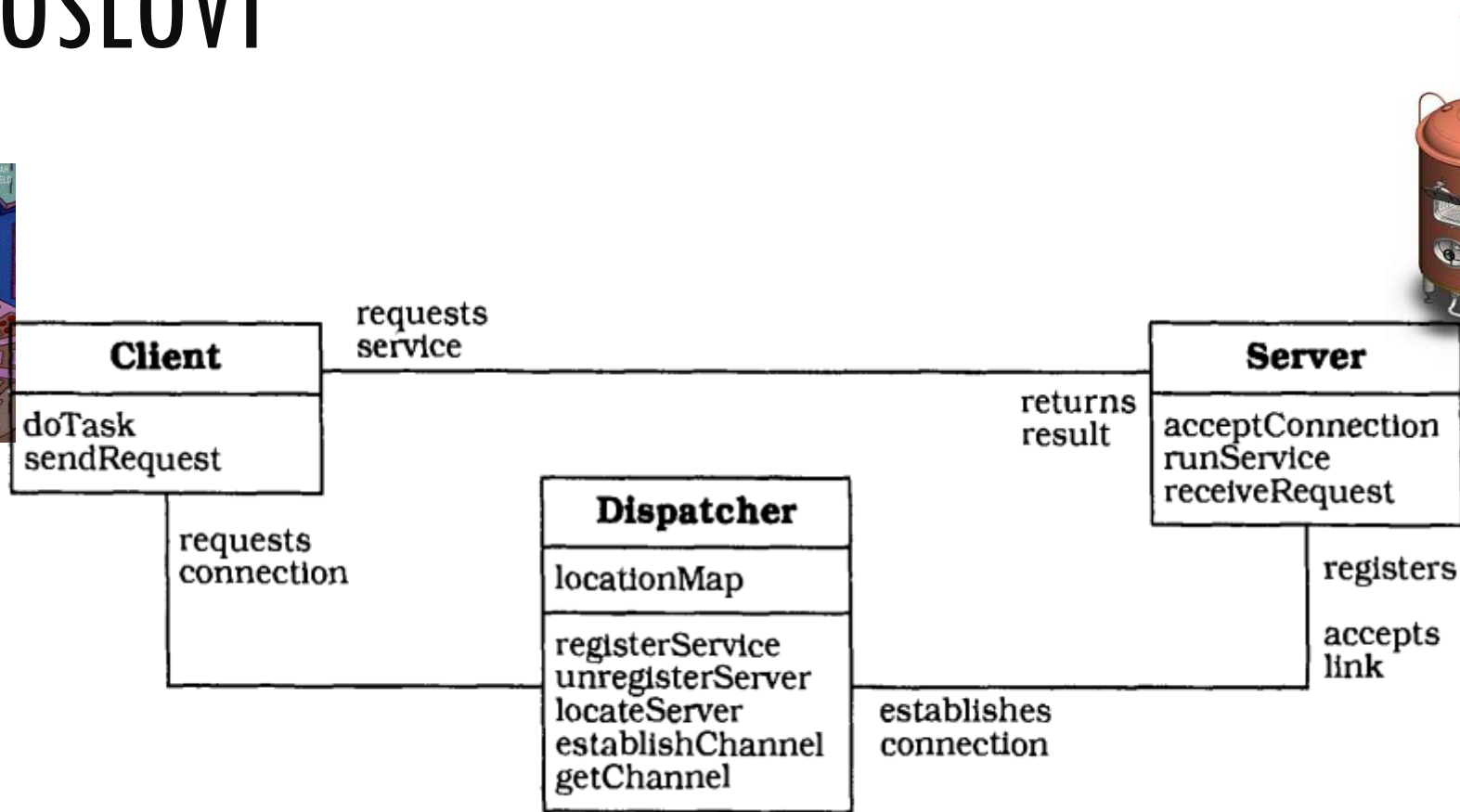
Změna adresy 😊

Ted' jsem tady
69.69.69.69:42222

Dispatcher



NÁZVOSLOVÍ



CLIENT - DISPATCHER — SERVER

* pro domácí studium

dispatcher

- poskytuje službu, jež umožňuje klientovi používat pro identifikaci serverů jejich jména a nikoli fyzická umístění (transparentnost umístění)
- navíc se stará o vytvoření komunikačního kanálu mezi klientem a serverem

server

- poskytuje služby klientům
- registruje se či je registrován pod svým jménem a adresou u dispatchera
- může se nacházet na stejném stroji jako klient, ve stejné místní síti či být dostupný přes Internet

klient

- spoléhá na dispatchera s nalezením serveru a pro vytvoření spojení s ním
 - nadále pak pracuje pouze s kanálem k serveru, který mu dispatcher poskytl
- může si za běhu prohodit roli se serverem

VARIANTY



distribuované dispatchery

komunikace spravována klientem/dispatcherem

- dispatcher vytváří spojení se serverem

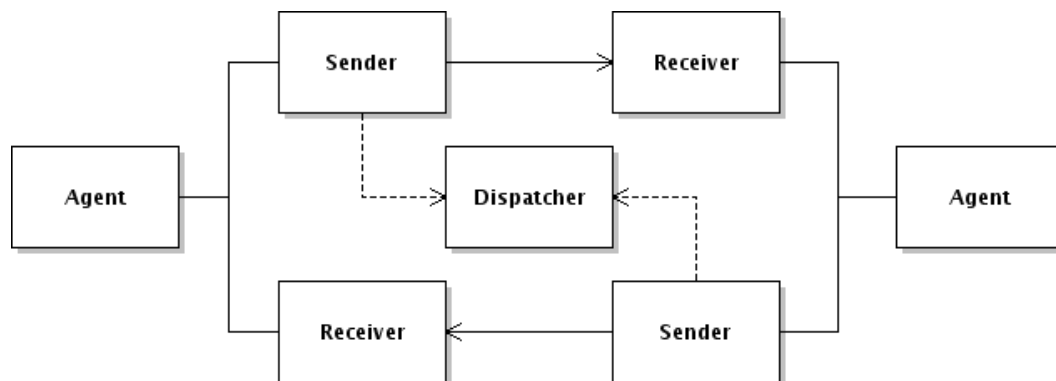
s více způsoby komunikace

- servery se navíc registrují s podporovanými druhy komunikace

client-dispatcher-service

- dispatcher si pamatuje mapování služeb na implementující servery

kombinace se vzorem forwarder-receiver



Kde zjistím kolik je
Kde plavka??



SHRNUTÍ

Dynamická síť



Statická síť



Výhody

Flexibilita

- Zaměnitelnost serverů/služeb runtime

Odolnost

- Selhání serverů => při obnovení se zaregistrují

Transparentnost

- Abstrakce nad fyzickými adresami
- Lze jednoduše měnit topologii sítě

Nevýhody

Efektivita

- Nutná komunikace s dispatcherem

Možnost
cachování