Akka.NET

10. Remote

Überblick

- Akka:

  CPUs einer Maschine

- Akka.Remote:

  CPUs mehrerer Maschinen

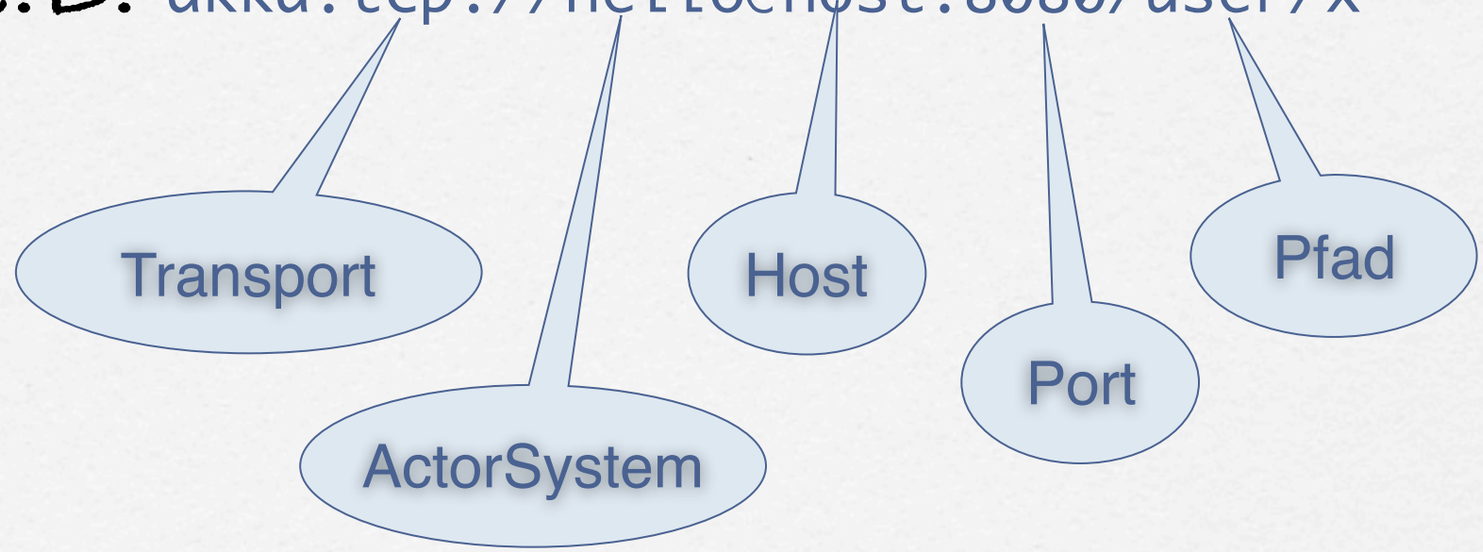
- Akka.Cluster:

  Dynamische Skalierung

Remote Aktor ansprechen

- via `IActorRef` (zeig mir den Weg dorthin)

  z.B. `akka.tcp://hello@host:8080/user/x`



- Zum Vergleich Lokal: `akka://hello/user/x`

Lokal oder Remote?

- Konfiguration

```
akka.actor.provider =
"Akka.Remote.RemoteActorRefProvider, Akka.Remote“

akka.remote {
    helios.tcp.port = 8000       # oder 0 (ausgehend)
    helios.tcp.hostname = localhost
}
```

- Umgang mit „Remote“ Aktoren

Verbindung

- Beim ersten Sendeversuch: Verbindungsaufbau
- Nach Verbindungsabbruch: erneuter Aufbau
- Symmetrische Verbindung muss gewährleistet sein!
- Verbindung nicht geschützt!

Erster Versuch

- Aktor auf Remote Seite erzeugen

```
var config = ConfigurationFactory.Load();
var system = ActorSystem.Create("Remote", config);
var increment = system.ActorOf(
    Props.Create<IncrementActor>(), "increment"
);
```

- von Lokaler Seite ansprechen

```
var config = ConfigurationFactory.Load();
var system = ActorSystem.Create("Select", config);
var actor = system.ActorSelection(
    "akka.tcp://Remote@localhost:8080/user/increment"
);
actor.Tell("hello over there");
```

Coding Time

- [ ] Solution: Remote, Projekte: Simple*
- [ ] TODOS:
  - [ ] Aktorsystem Lokal/Remote
  - [ ] Aktoren Local/Remote
  - [ ] Nachrichten senden + beantworten
  - [ ] Zeit messen / Vergleich

Nachteile

- Systemaufbau an mehreren Stellen
- Ansprechen der Aktoren mühsam
- unflexibel

Vorteile

- Lokal- und Remote-Seite in getrennten Code-Basen
- C# und F# mischbar

Die Lösung: Deployment

- alles auf Lokalem Actor System
- Konfiguration

```
akka.actor.deployment {
    /pfad_innerhalb_user {
        remote = “akka.tcp://system@host:port“
    }
}
```

- Actor Erzeugung

```
system.ActorOf(…, “name“);
```

Remote Deployment

- Remote Deployment : mehrere Services

```
akka.actor.deployment {
    /Inc8001 {
        remote = “akka.tcp://Service@localhost:8081“
    }
    /Inc8002 {
        remote = “akka.tcp://Service@localhost:8082“
    }
}
```

Lokal

Entferntes System

```
system.ActorOf(
    Props.Create<Increment>(),
    "Inc8001"
);
```

statische Verteilung

- Router-Group zum Verteilen

```
akka.actor.deployment {
    /Inc {
        router = round-robin-group
        route.paths = [“/user/Inc8081“, “/user/Inc8082“…]
    }
}
```

Vorteile

- zentrales Deployment
- Ausnutzung mehrerer CPUs

Nachteile

- statisch
- keine Fehlerbehandlung bei Ausfall eines Remote Actorsystems
  —> andere Routing-Strategie