

Akka.NET

Aufgabe 4: Wörter zählen

Pipes and Filter

- Erzeugung eines Datenflusses
- „Filter“ sind verarbeitende Instanzen
- „Pipes“ verbinden die Einheiten
- Quelle -> Filter -> Filter ... -> Ziel

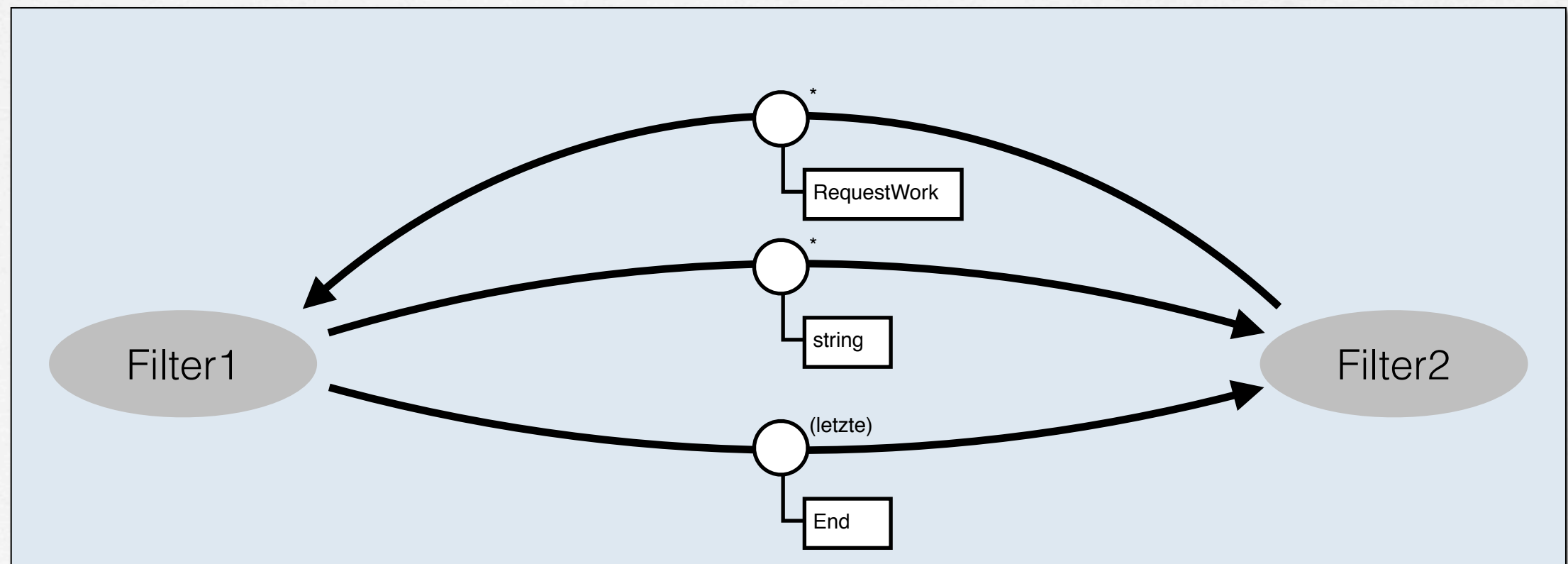


Stream Modelle

- Push Modell: Unidirektional
 - + einfach
 - möglicher Puffer-Überlauf
- Pull Modell: Bidirektional
 - + Empfänger kontrolliert Datenmenge
 - komplizierter
- Managed Queue, Throttling, Drop, ...

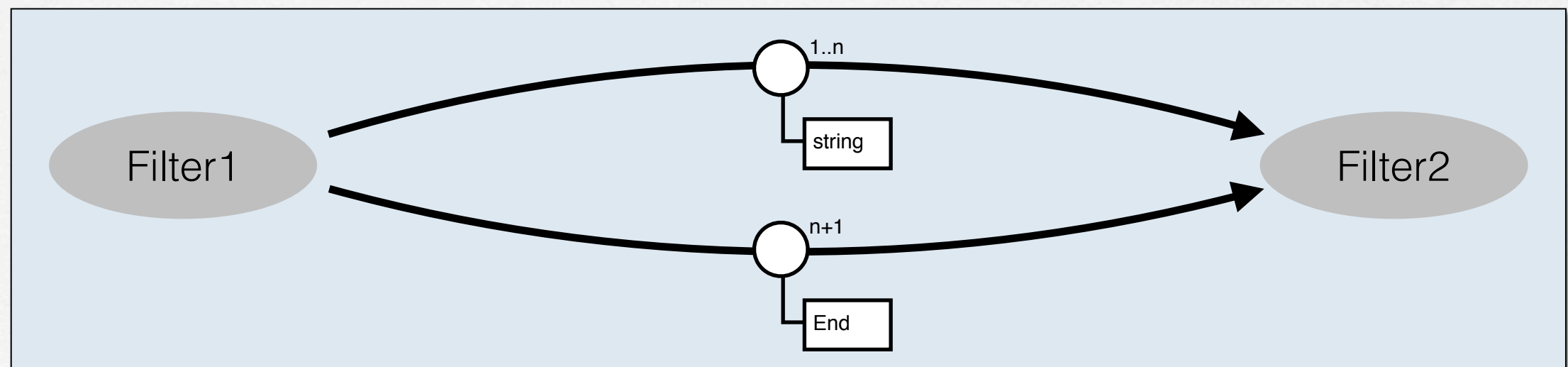
Pull Modell

- Empfänger weiß Puffer Füllstand
- Wenn zu leer -> Daten anfordern



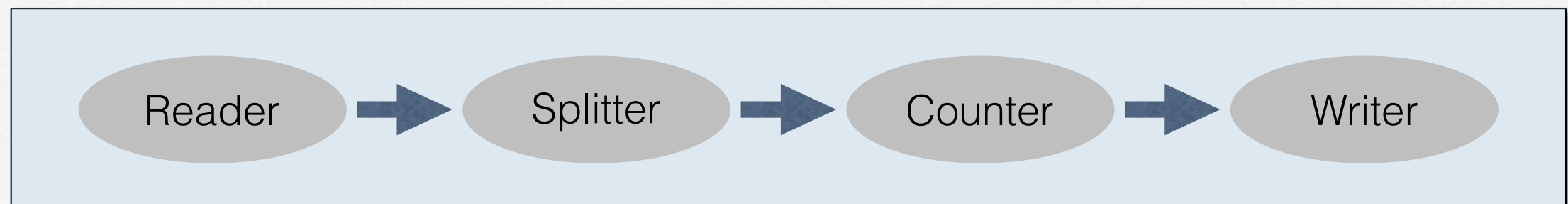
Push Modell

- *string*
Text wird zum nächsten Filter transportiert
- *End*
Ende des Datenstroms



Verarbeitungskette

- Aufteilung der Arbeit auf einzelne spezialisierte Aktoren
- Counter ist besonders, er puffert sein Zwischenergebnis bis Eingabe endet

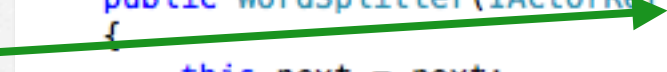


Konstruktion – Push Modell

- ❑ Akteure müssen Nachfolger kennen
- ❑ Aufbau der Kette erfolgt rückwärts

```
var system = ActorSystem.Create("Words");  
  
var console = system.ActorOf(  
    Props.Create<ConsoleWriter>()  
);  
var counter = system.ActorOf(  
    Props.Create<WordCounter>(console)  
);  
var splitter = system.ActorOf(  
    Props.Create<WordSplitter>(counter)  
);  
system.ActorOf(  
    Props.Create<TextReader>(SampleText(), splitter)  
);
```

```
public class WordSplitter : ReceiveActor  
{  
    private readonly IActorRef next;  
  
    public WordSplitter(IActorRef next)  
    {  
        this.next = next;  
  
        Receive<string>(s => SplitIntoWords(s));  
        Receive<End>(next.Tell);  
    }  
}
```



Projekt „WordCount“

- Erweiterung der Verarbeitungskette um einen Filter, der alle Wörter in „kleinbuchstaben“ umwandelt.

Lösungsvorschlag

- Einhaltung des Protokolls
- Einbindung in Verarbeitungskette

```
var console = system.ActorOf(Props.Create<ConsoleWriter>());  
var counter = system.ActorOf(Props.Create<WordCounter>(console));  
var splitter = system.ActorOf(Props.Create<WordSplitter>(counter));  
var caser = system.ActorOf(Props.Create<LowerCaser>(splitter));  
system.ActorOf(Props.Create<FileReader>(args[0], caser));
```

```
using Akka.Actor;  
using WordCount.Messages;  
  
namespace WordCount  
{  
    /// <summary>  
    /// Convert a string stream into lower case  
    /// </summary>  
    1 reference  
    public class LowerCaser : ReceiveActor  
    {  
        0 references  
        public LowerCaser(IACTORRef next)  
        {  
            Receive<string>(s => next.Tell(s.ToLower()));  
            Receive<End>(end => next.Tell(end));  
        }  
    }  
}
```