

# AvO-Übung 5

## Rechnerübung zu Ice

Andreas I. Schmied und Jan-Patrick Elsholz

Institut für Verteilte Systeme

7. Januar 2008

# Ablauf der Übung

- Kombination aus Tutorial und eigenständigem Arbeiten
  - Einfacher, vorkonfigurierter Einstieg in die Middleware
- Im Windows-Pool O28-1001
  - Schnelle Hilfestellung am Rechner
  - Nutzen Sie das und rufen Sie uns an Ihren Platz
- Folien werden auf zweitem Bildschirm präsentiert
  - PDF-Datei zum eigenen Zurückblättern auch selbst laden
- Code nicht nur abschreiben, sondern durchdenken!
  - Quellcode liegt im Arbeitsverzeichnis der Betreuer (s.u.)
- Entwicklungsumgebung: beliebig
  - Unter Windows und Linux (via SSH)
  - Aufgaben sind i.d.R. mit Text-Editor und Shell lösbar
  - Eclipse/Netbeans/... sprechen Sie bitte mit den Betreuern ab
- Übernehmen Sie vorerst Pfade und Namen von den Folien

# Arbeitsumgebung

- Rechner: `ygramul.informatik.uni-ulm.de` (Ubuntu-Linux)
- Account: `praktikumapx`
- Password: (extra Folie, bitte aufschreiben!)
  
- Home-Verzeichnis „~“: `/home/sikanda/praktikumapx`
- Software-Pakete: `~/arc` (z.B. für zu Hause)
- Software-Installation: `~/pkg` (Pfade gesetzt)
- Dokumentation: `~/doc/index.html` (für Java und Ice)
  - Demos: `~/pkg/ice-3.2.1/demos`
  
- Zugriff von Windows aus
  - Freigabe: `\\sikanda.informatik.uni-ulm.de\praktikumapx`
  - Startmenü/Run „cmd“: `net use U: \\sikanda\praktikumapx`
  - SSH-Client: `~/bin/PUTTY.EXE` etc.

# Arbeitsverzeichnis

- Ihre Email-Adresse sei *Name@uni-ulm.de*
- **Ihr Arbeitsverzeichnis** ist `~/users/Name`
  - im Folgenden das aktuelle Verzeichnis
  - erstellen mit: `cd users ; mkdir Name; cd Name`
- Port-Nummern pro Matrikel-Nummer verschieden
  - fünfstellig: „2xyzN“
  - xyz sind die letzten drei Ziffern Ihrer Matrikel-Nummer
  - N sind Zahlen von 0-9
  - Melden Sie verbleibende Port-Kollisionen den Betreuern

## Alle Teilnehmer teilen sich den selben Account!

- Seien Sie bitte umsichtig mit Dateioperationen
- Verlassen Sie bitte Ihr Arbeitsverzeichnis nicht

# Aufgabe 1: Zeitsignal-Service

- Slice-Schnittstelle
  - Zum Einstieg siehe Ice-Manual, Kapitel 3.2 (in ~/doc)
- Service-Implementierung
- Server-Anwendung
  - Objektname: *time@avo*
  - Stringified Proxy ausgeben
- Client-Anwendung
  - Stringified Proxy als Parameter
  - Shutdown-Operation als *oneway*-Aufruf

# Slice-Modul verfassen und übersetzen

Dateiname: ./TimeService.ice

```
1 package avo.u5;  
2  
3 interface TimeService {  
4  
5     String now();  
6     void shutdown();  
7 }
```

# Slice-Modul verfassen und übersetzen

Dateiname: ./TimeService.ice

```
1 package avo.u5;
2
3 interface TimeService {
4
5     String now();
6     void shutdown();
7 }

```

→

```
1 module avo { module u5 {
2
3     interface TimeService {
4         idempotent string now();
5         idempotent void shutdown();
6     };
7
8 };};

```

# Slice-Modul verfassen und übersetzen

Dateiname: ./TimeService.ice

1	<b>package</b> avo.u5;	1	<b>module</b> avo { <b>module</b> u5 {
2		2	
3	<b>interface</b> TimeService {	3	<b>interface</b> TimeService {
4		4	idempotent <b>string</b> now();
5	String now();	5	idempotent <b>void</b> shutdown();
6	<b>void</b> shutdown();	6	};
7	}	7	
		8	};};

Language Mapping Slice→Java

```
1 rm -fr ./gen
2 mkdir ./gen
3 slice2java --output-dir ./gen TimeService.ice
```

# Service-Implementierung

```
1 package avo.u5; // Datei: src/avo/u5/TimeService1.java
2
3 public class TimeService1 extends _TimeServiceDisp {
4
5     public String now(Ice.Current __current) {
6         System.out.println("Now command received.");
7         java.text.SimpleDateFormat date =
8             new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
9         return date.format(new java.util.Date());
10    }
11
12    public void shutdown(Ice.Current __current) {
13        System.out.println("Shutdown command received.");
14        __current.adapter.getCommunicator().shutdown();
15    }
16 }
```

# Server-Grundgerüst

```
1 package avo.u5; // Datei: src/avo/u5/Server1.java
2 public class Server1 {
3     public final static int PORT = 22222;
4     public static void main(String[] args) {
5         Ice.Communicator ic = null;
6         try {
7             ic = Ice.Util.initialize(args);
8             ... naechste Folie ...
9             ic.waitForShutdown();
10        }
11        catch (Exception e) { e.printStackTrace(); }
12        if (ic != null) {
13            try { ic.destroy(); }
14            catch (Exception e) { e.printStackTrace(); }
15        } } }
```

## Adapter und Objekt erzeugen

```
1 Ice.ObjectAdapter adapter
2     = ic.createObjectAdapterWithEndpoints("avo", "default -p "+PORT);
3
4 Ice.Object obj = new TimeServiceI();
5
6 Ice.ObjectPrx prx = adapter.add(
7     obj,
8     Ice.Util.stringToIdentity("time"));
9
10 System.out.println("Service-Proxy:\n" + prx.ice_toString());
11
12 adapter.activate();
```

# Client-Implementierung

```
1 // Gleicher Rahmen wie in Server1, Datei: src/avo/u5/Client.java
2
3 ic = Ice.Util.initialize(args);
4 String str = args[0];
5 Ice.ObjectPrx obj = ic.stringToProxy(str);
6
7 if (args.length>1 && args[1].equals("--shutdown")) {
8     Ice.ObjectPrx oneway = obj.ice_oneway();
9     TimeServicePrx svc = TimeServicePrxHelper.uncheckedCast(oneway);
10    svc.shutdown();
11 }
12 else {
13     TimeServicePrx svc = TimeServicePrxHelper.checkedCast(obj);
14     String now = svc.now();
15     System.out.println(now);
16 }
```

# Java-Code übersetzen

- Ice.jar ist schon in \$CLASSPATH gesetzt, ggf. anpassen:  
–classpath \$ICE\_HOME/lib/Ice.jar:cls

- Neu übersetzen

```
1 rm -fr ./cls
2 mkdir ./cls
3 javac -d ./cls -sourcepath src:gen src/avo/u5/*.java
```

# Test

- Sie benötigen 2 Konsolen zur Ausführung
  - Noch zweimal SSH-Client bemühen oder *screen* nutzen
- Server starten:  
`java avo.u5.Server1`
- Stringified Proxy kopieren
- Client starten:  
`java avo.u5.Client '⟨Stringified Proxy einfügen⟩'`
  - Alternativ `--shutdown` anhängen
- Indirect Proxies und Location Services behandeln wir in Ü6

## Aufgabe 2: Ice.Application

- Grundgerüst immer gleich
  - Ice liefert vorgefertigtes Rahmenprogramm: *Ice.Application*
- Server als Ice.Application umschreiben
- Vorteile?
  - Shutdown-Hooks
  - Behandlung von Config-Parametern
  - ...

## Server als Ice.Application

```
1 package avo.u5; // Datei: src/avo/u5/Server2.java
2 public class Server2 extends Ice.Application {
3     ...
4     public int run(String[] args) {
5         Ice.Communicator ic = communicator();
6         ... siehe Folie 9 ...
7         ic.waitForShutdown();
8         return 0;
9     }
10    public static void main(String[] args) {
11        Server2 app = new Server2();
12        int status = app.main("Server2", args);
13        System.exit(status);
14    } }
```

## Aufgabe 3: IceBox

- Server als IceBox umschreiben
  - Service-Implementierung
  - Deployment Deskriptoren
- Vorteile?
  - Service-Container für Menge an Diensten
  - Gemeinsame Administration via Properties
  - Integration in IceGrid
  - ...

## Server als IceBox

```
1 package avo.u5; // Datei: src/avo/u5/BoxTimeServiceI.java
2 public class BoxTimeServiceI
3     extends Ice.LocalObjectImpl implements IceBox.Service {
4     private Ice.ObjectAdapter adapter;
5     public void
6     start(String name, Ice.Communicator communicator, String[] args) {
7         adapter = communicator.createObjectAdapter(name);
8         Ice.Object obj = new TimeServiceI();
9         Ice.ObjectPrx prx = adapter.add(
10             obj,
11             communicator.stringToIdentity("time"));
12         System.out.println("Service-Proxy:\n" + prx.ice_toString());
13         adapter.activate();
14     }
15
16     public void stop() { adapter.deactivate(); }
17 }
```

# IceBox konfigurieren und einsetzen

- Datei: config-time.icebox

```
IceBox.ServiceManager.Endpoints=tcp -p 22221
```

```
IceBox.Service.Time=avo.u5.BoxTimeService ←
```

```
--Ice.Config=config-time.service
```

- Datei: config-time.service

```
Time.Endpoints=tcp -p 22222
```

- Java-IceBox starten als IceBox.Server

```
java IceBox.Server --Ice.Config=config-time.icebox
```

- Java-IceBox stoppen via IceBox.Admin

```
java IceBox.Admin --Ice.Config=config-time.icebox shutdown
```

- Client-Parameter --shutdown funktioniert nicht mit IceBox!

## Aufgabe 4: IceStorm

- Zeitsignal alle 3s asynchron an Clients melden
- Topic

```
1 interface TickEvent {  
2     void tick(String now);  
3 }
```

# IceStorm Anwendung

- IceStorm Server
  - Stellt IceStorm-Service und TopicManager bereit
  - Anwendung in C++ geschrieben, als Binary verfügbar
  - Versuch über Java-IceBox zu starten scheitert!
  - Datenbank-Verzeichnis erstellen mit: `mkdir db`
  - Konfigurationsdateien: `config-tick.icebox`, `config-tick.service`
- Publisher (`Ice.Application`)
  - Erzeugt Ereignisse
  - Konfigurationsdatei: `config-tick.pub`
- Subscriber (`Ice.Application`)
  - Empfängt Ereignisse
  - Konfigurationsdatei: `config-tick.sub`

# IceStorm Server konfigurieren

- Datei: config-tick.icebox

```
1 IceBox.ServiceManager.Endpoints=tcp -p 22223
2 IceBox.Service.IceStorm=IceStormService,32:createIceStorm ←
3 --Ice.Config=config-tick.service
```

- Datei: config-tick.service

```
4 IceStorm.TopicManager.Proxy= ↔
5   AvolceStorm/TopicManager:default -p 22224
6 IceStorm.TopicManager.Endpoints=default -p 22224
7 IceStorm.InstanceName=AvolceStorm
8 IceStorm.Publish.Endpoints=default -p 22225:udp -p 22225
9 IceStorm.Trace.TopicManager=2
10 IceStorm.Trace.Topic=1
11 IceStorm.Trace.Subscriber=1
12 IceStorm.Trace.Flush=1
13 IceStorm.Flush.Timeout=2000
14 Freeze.DbEnv.IceStorm.DbHome=db
```

# Publisher

- Datei: config-tick.pub

```
IceStorm.TopicManager.Proxy= ↔  
AvolceStorm/TopicManager:default -p 22224
```

- Datei: src/avo/u5/Publisher.java

- Ice.Application startet mit

```
public static void main(String[] args) {  
    new Publisher().main("Publisher", args, "config-tick.pub");  
}
```

## Publisher: Topic erzeugen

```
1 public int run(String[] args)
2 {
3     IceStorm.TopicManagerPrx manager =
4         IceStorm.TopicManagerPrxHelper.checkedCast(
5             communicator().propertyToProxy("IceStorm.TopicManager.Proxy"));
6
7     IceStorm.TopicPrx topic=null;
8     try {
9         topic = manager.retrieve(topicName);
10    }
11    catch(IceStorm.NoSuchTopic e) {
12        try {
13            topic = manager.create(topicName);
14        }
15        catch(IceStorm.TopicExists ee) {}
16    }
17    ... naechste Folie ...
18    return 0;
19 }
```

## Publisher: Events verschicken

```
1 Ice.ObjectPrx publisher = topic.getPublisher();
2 publisher = publisher.ice_datagram();
3 TickEventPrx event = TickEventPrxHelper.uncheckedCast(publisher);
4
5 System.out.println("Publishing tick events, press ^C to terminate.");
6 try {
7     java.text.SimpleDateFormat date =
8         new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
9     while(true) {
10         event.tick(date.format(new java.util.Date()));
11         System.out.println("Tick!");
12         Thread.currentThread().sleep(3000);
13     }
14 }
15 catch(Ice.CommunicatorDestroyedException ex) { }
16 catch(java.lang.InterruptedExcepion e) { }
```

# Subscriber

- Datei: config-tick.sub

```
TickEvent.Subscriber.Endpoints=tcp:udp
```

```
IceStorm.TopicManager.Proxy= ↔
```

```
AvolceStorm/TopicManager:default -p 22224
```

- Ice.Application startet analog
  - Topic ebenfalls beziehen oder erzeugen
- Implementierung des Events

```
1 public final static String topicName = "tick";
2
3 public class TickEventI extends _TickEventDisp {
4     public void tick(String now, Ice.Current current) {
5         System.out.println("Got a tick: "+now);
6     }
7 }
```

## Subscriber: Events empfangen

```
1 Ice.ObjectAdapter adapter =
2     communicator().createObjectAdapter("TickEvent.Subscriber");
3
4 java.util.Map<String,String> qos = new java.util.HashMap<String,String>();
5 Ice.ObjectPrx subscriber = adapter.addWithUUID(new TickEventI());
6 subscriber = subscriber.ice_datagram();
7
8 try {
9     topic.subscribeAndGetPublisher(qos, subscriber);
10 }
11 catch(IceStorm.AlreadySubscribed e) { e.printStackTrace(); return 1; }
12 catch(IceStorm.BadQoS e) { e.printStackTrace(); return 1; }
13
14 adapter.activate();
15 shutdownOnInterrupt();
16 communicator().waitForShutdown();
17 topic.unsubscribe(subscriber);
```