

# Networking with Akka

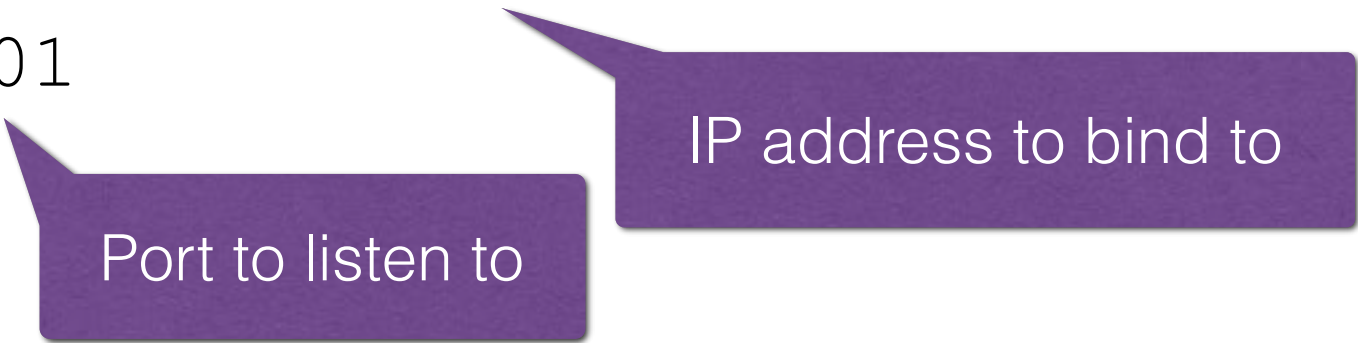
# Remote actors

- So far we were always dealing with local actors, residing within the same Java VM
- Akka allows communicating with *remote actors* using the same abstractions as for the local ones
- However, some additional steps are needed to configure the *distributed* system:
  - Akka system should be configured to accept incoming network connections (local IP address and port)
  - To find actors on a remote Akka system, we should know the remote IP address(es) and port(s)

# Enabling incoming connections

Default: **application.conf** in the root of \$CLASSPATH

```
akka {  
  actor {  
    provider = remote  
  }  
  remote {  
    enabled-transport =  
      ["akka.remote.netty.tcp"]  
    netty.tcp {  
      hostname = "127.0.0.1"  
      port = 10001  
    }  
  }  
}
```



# Note on addresses

- If you only run several instances of your program on a single computer (for development and debugging)
  - Use 127.0.0.1 and a *unique port number* for each instance (you will need a separate configuration file per instance)
- To run the program on multiple computers,
  - Make sure they are connected to the same network
  - Use the actual IP addresses instead of 127.0.0.1

# Application parameters

In the example we use several configuration files in **src/main/resources/node\*.conf** and we add some custom application parameters there

```
akka {  
  ...  
}  
nodeapp {  
  id = 1  
  remote_ip = "127.0.0.1"  
  remote_port = 10000  
}
```

The ID of the current node

Remote address and port  
to connect to

# Reading the config

In the main() function:

Read our custom parameter

```
Config config = ConfigFactory.load();  
  
int myId = config.getInt("nodeapp.id");  
  
ActorSystem system =  
    ActorSystem.create("mysystem", config);
```

Pass the other parameters to Akka

# Accessing remote actors

- To access a remote actor we either need to have its reference (ActorRef) or use the “remote path”:
- “akka.tcp://**mysystem**@**host:port**/user/**node**”



Akka system name

The name of the actor

Remote host and port

```
getContext().actorSelection(path).tell(
    message, getSelf());
```

# Getting the ActorRef

- However, it is more convenient to use **ActorRef** of a remote actor
- To get it, you should either receive a message from the remote actor and use **getSender()**, or receive the ActorRef object from someone else in a message
- In the example: we use the remote path in the very first message to request the group members from an actor, later we communicate using **ActorRef** objects



# Example (on Moodle)

- Simplistic peer-to-peer system:
  - The first node starts alone
  - The others join the existing group:
    - Request the group list from one of the members
    - When the group list arrives, joins the group by announcing their presence to everyone in the list
- After joining, all nodes know **ActorRefs** of everyone else in the group

# Starting multiple nodes on a single computer

- For every node you will need a separate configuration file, e.g.,
  - **src/main/resources/node0.conf**
- To launch:

```
$ gradle run -Dconfig=node0.conf
```