

Eike Stepper

stepper@esc-net.de
<http://www.esc-net.de>
<http://thegordian.blogspot.com>

Berlin, Germany



The Net4j Signalling Platform

Developing Pluggable Client/Server Applications

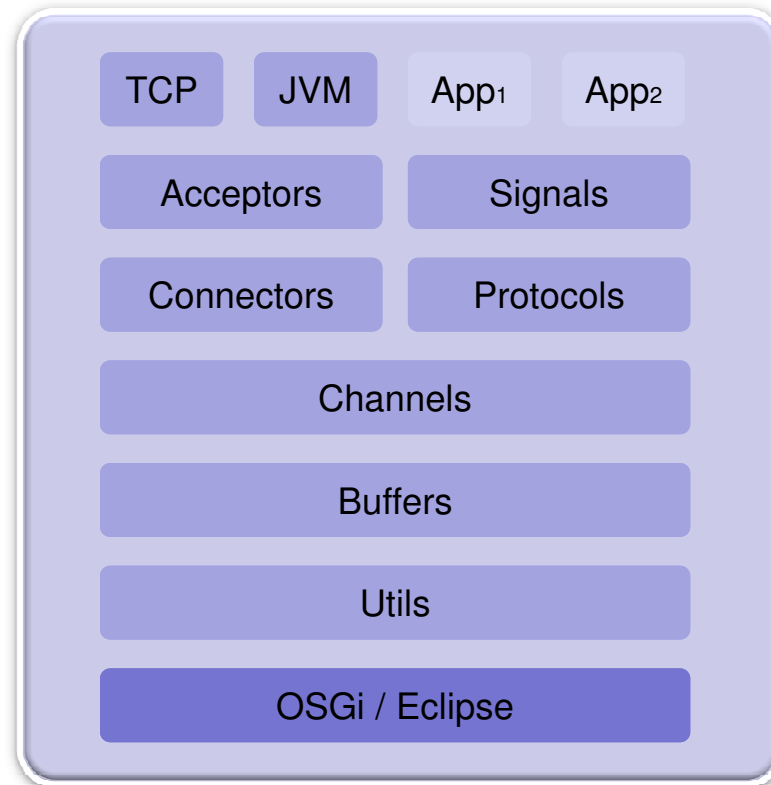
Agenda

- **Requirements**
- **Architecture**
 - w Buffers
 - w Channels
 - w Connectors
 - w Acceptors
 - w Protocols
 - w Signals
- **Examples**
- **FileShare Demo**

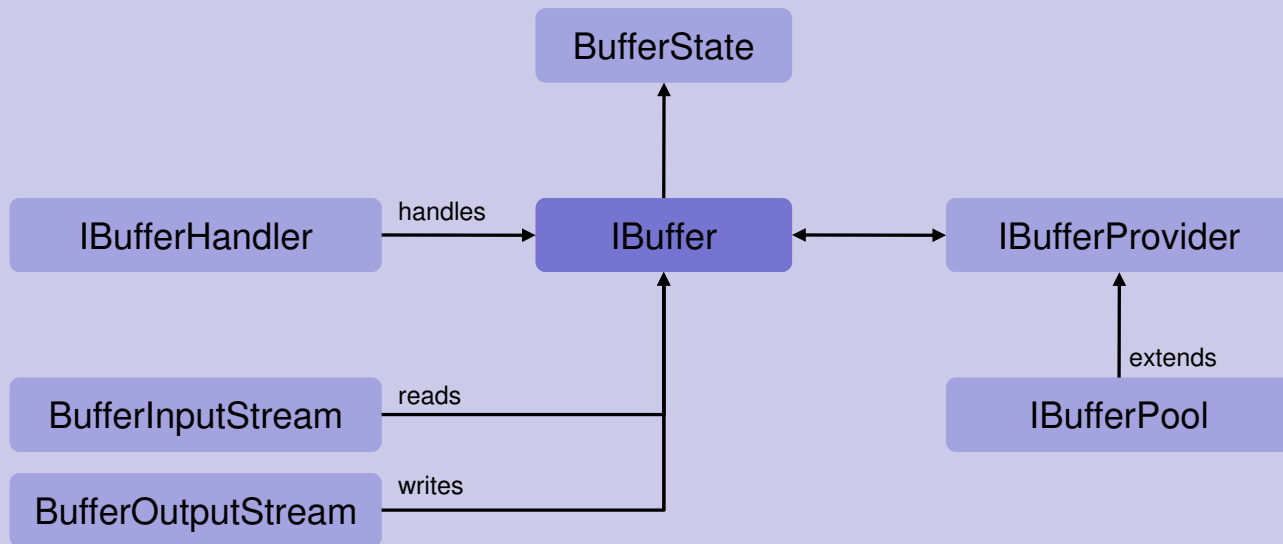
Requirements

- High performance
 - ↳ `java.nio.DirectByteBuffer`, zero copying
- Good scalability
 - ↳ `java.nio.channels.Selector`, single I/O thread possible
- Multiple transports
 - ↳ Shipped with TCP, HTTP and JVM transports
- Pluggable protocols
 - ↳ Independent of chosen transport
- Server-initiated push services (agent paradigm)
 - ↳ Asynchronous and synchronous requests towards the client
- OSGi™ and stand-alone modes

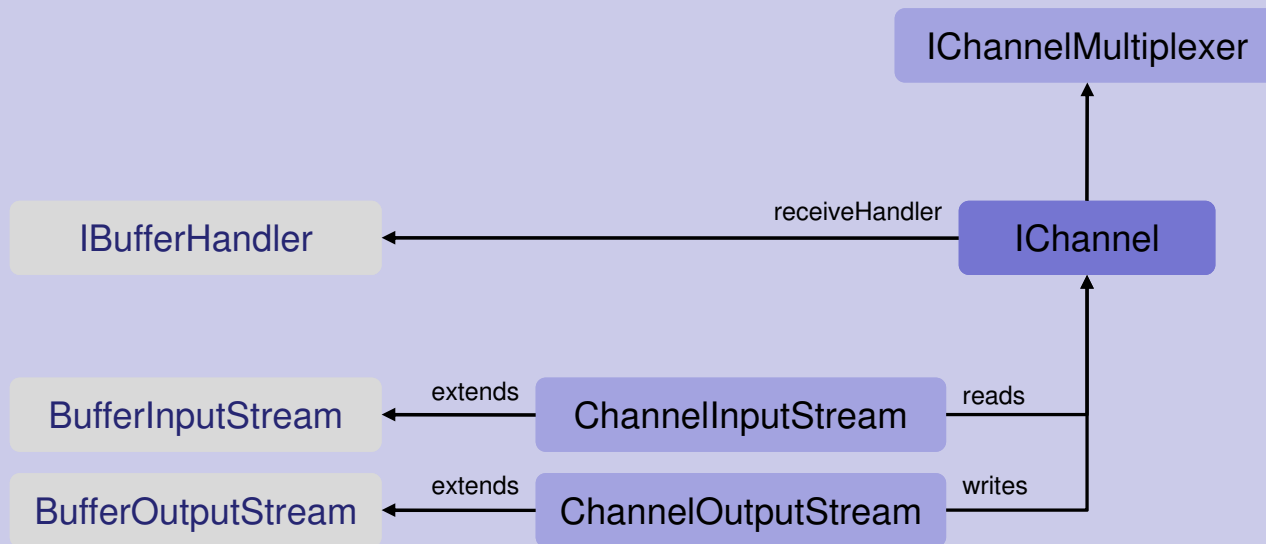
Architecture



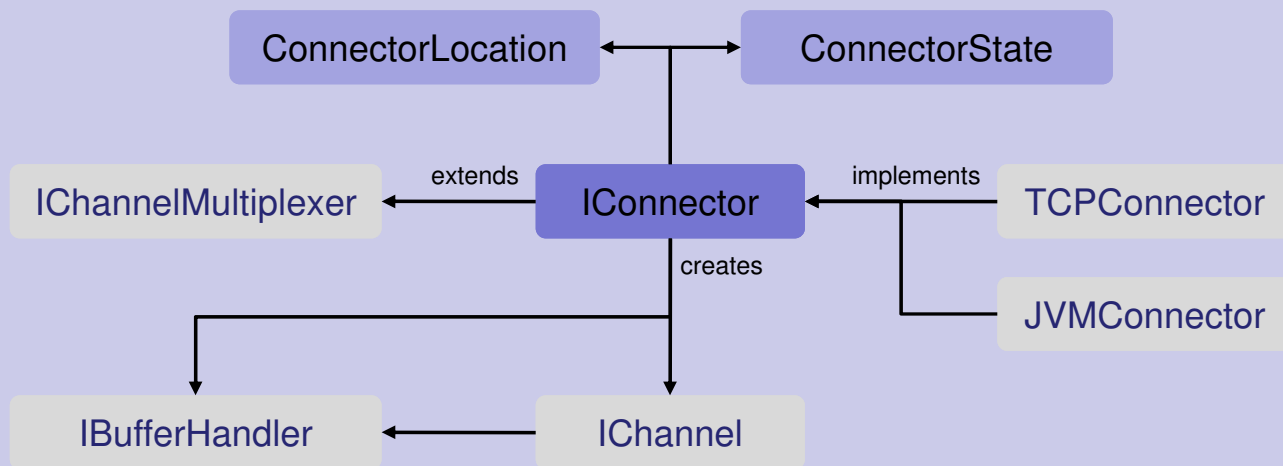
Buffers



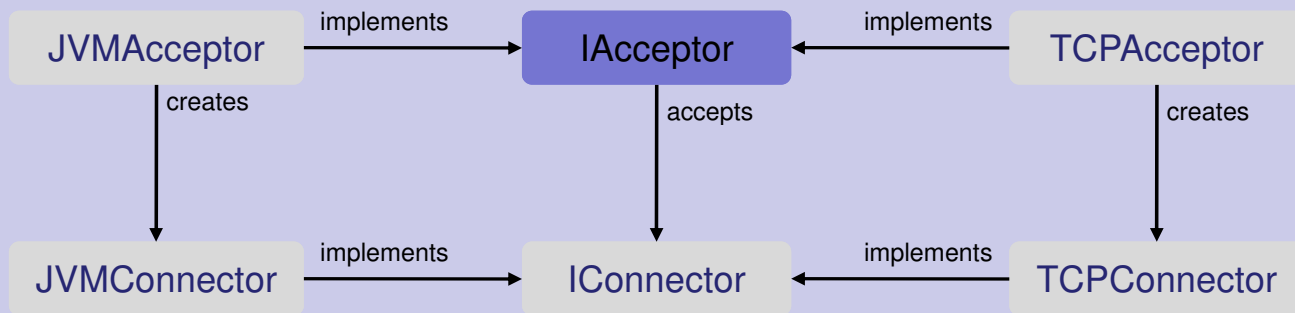
Channels



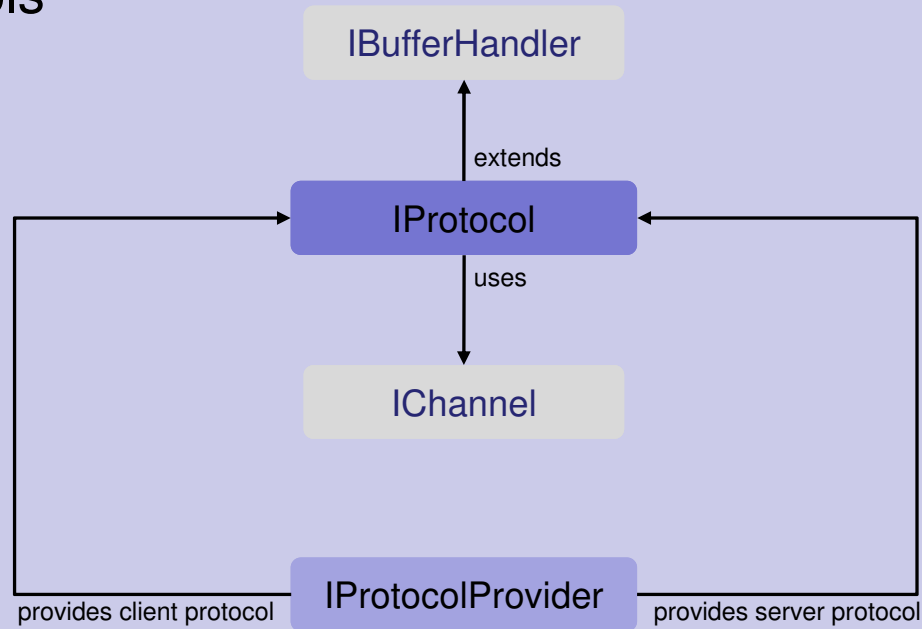
Connectors



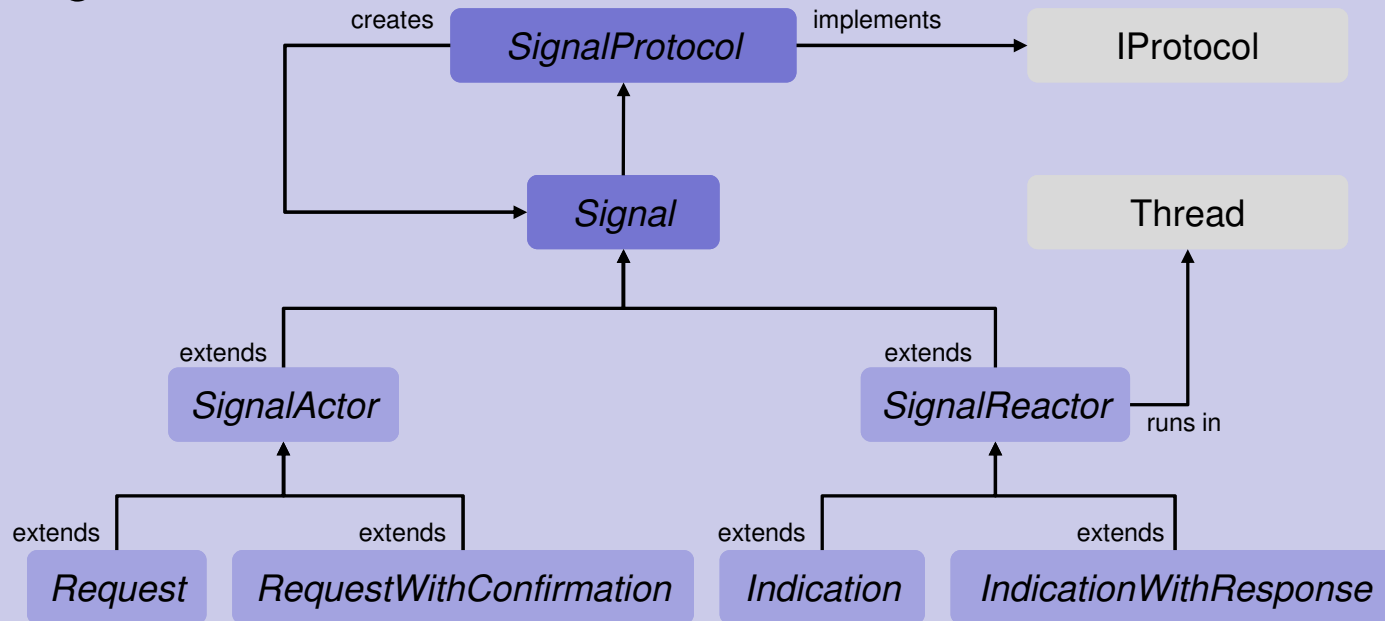
Acceptors



Protocols



Signals



Examples

```
public class JMSLogonRequest extends RequestWithConfirmation<Boolean> {
    private String userName;
    private String password;

    public JMSLogonRequest(IChannel channel, String userName, String password) {
        super(channel);
        this.userName = userName;
        this.password = password;
    }

    @Override
    protected short getSignalID() { return JMSProtocolConstants.SIGNAL_LOGON; }

    @Override
    protected void requesting(ExtendedDataOutputStream out) throws IOException {
        out.writeString(userName);
        out.writeString(password);
    }

    @Override
    protected Boolean confirming(ExtendedDataInputStream in) throws IOException {
        return in.readBoolean();
    }
}
```

Examples

```
public class JMSLogonIndication extends IndicationWithResponse
{
    private boolean ok;

    @Override
    protected short getSignalID()
    {
        return JMSProtocolConstants.SIGNAL_LOGON;
    }

    @Override
    protected void indicating(ExtendedDataInputStream in) throws IOException
    {
        String userName = in.readString();
        String password = in.readString();
        ok = JMSServer.INSTANCE.logon(userName, password);
    }

    @Override
    protected void responding(ExtendedDataOutputStream out) throws IOException
    {
        out.writeBoolean(ok);
    }
}
```

Examples

```
public class JMSServerProtocol extends SignalProtocol
{
    public String getType()
    {
        return JMSProtocolConstants.PROTOCOL_NAME;
    }

    @Override
    protected SignalReactor doCreateSignalReactor(short signalID)
    {
        switch (signalID)
        {
            case JMSProtocolConstants.SIGNAL_SYNC:
                return new JMSSyncIndication();

            case JMSProtocolConstants.SIGNAL_LOGON:
                return new JMSLogonIndication();
        }

        return null;
    }
}
```

Examples

```
// Start a TCP acceptor that is configured through extension points
IAcceptor acceptor = TCPUtil.getAcceptor(IPluginContainer.INSTANCE, "0.0.0.0:2036");

// Open a TCP connection that is configured through extension points
IConnector connector = TCPUtil.getConnector(IPluginContainer.INSTANCE, "localhost:2036");

// Open a channel with the JMS protocol
IChannel channel = connector.openChannel(JMSProtocolConstants.PROTOCOL_NAME);

try
{
    // Create a logon request and send it through the channel
    JMSLogonRequest request = new JMSLogonRequest(channel, "stepper", "secret");
    boolean ok = request.send();
}
catch (Exception ex)
{
    OM.LOG.error("Problem during logon", ex);
}
finally
{
    channel.close();
}
```

FileShare Demo

