



Eike Stepper

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

ES-Computersysteme
Berlin, Germany



Net4j Signalling Platform

Developing Pluggable Client/Server Applications



Agenda

1. Requirements

2. Architecture

- § Buffers
- § Channels
- § Connectors
- § Acceptors
- § Protocols
- § Signals

3. Examples

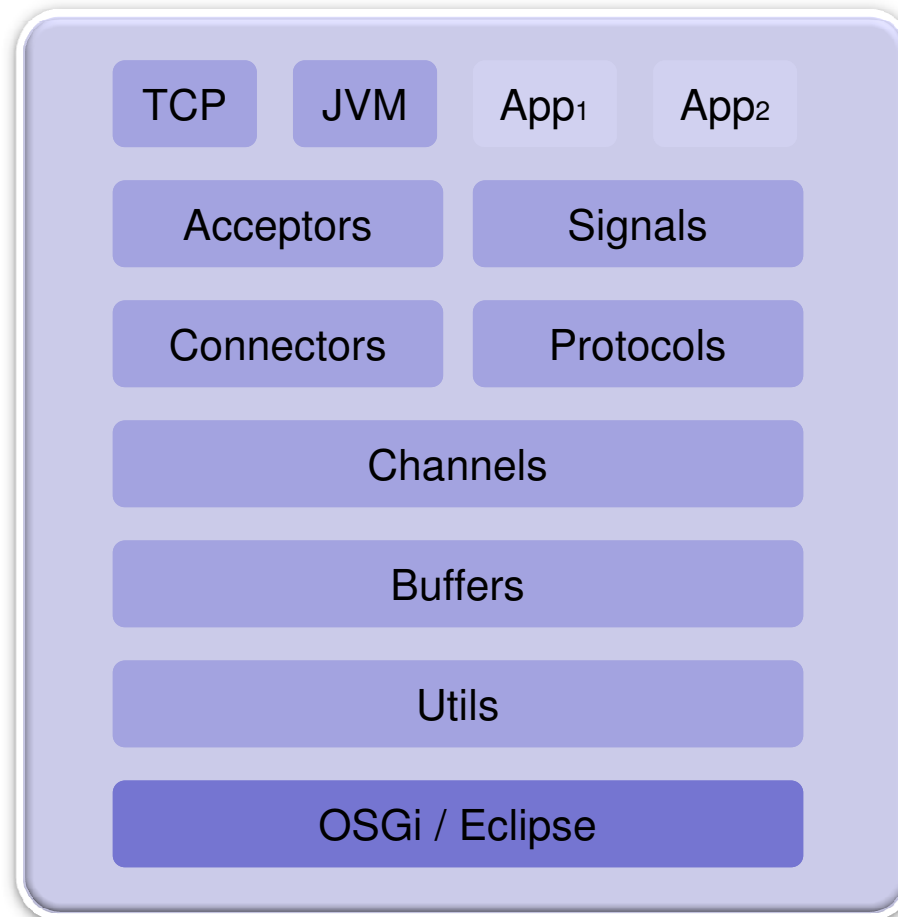
- § Request
- § Indication
- § SignalProtocol
- § Client Usage

4. Discussion

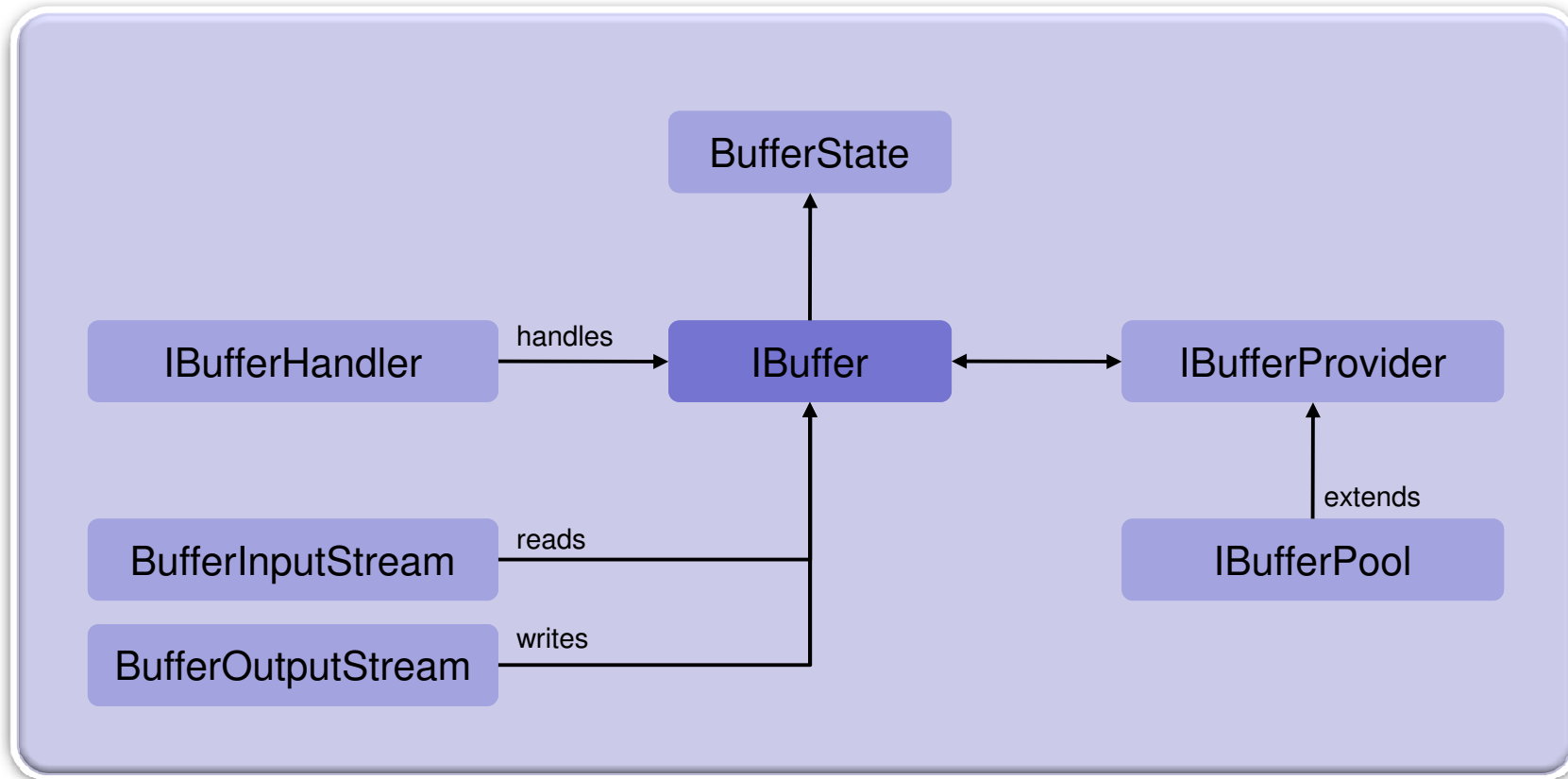
Requirements

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

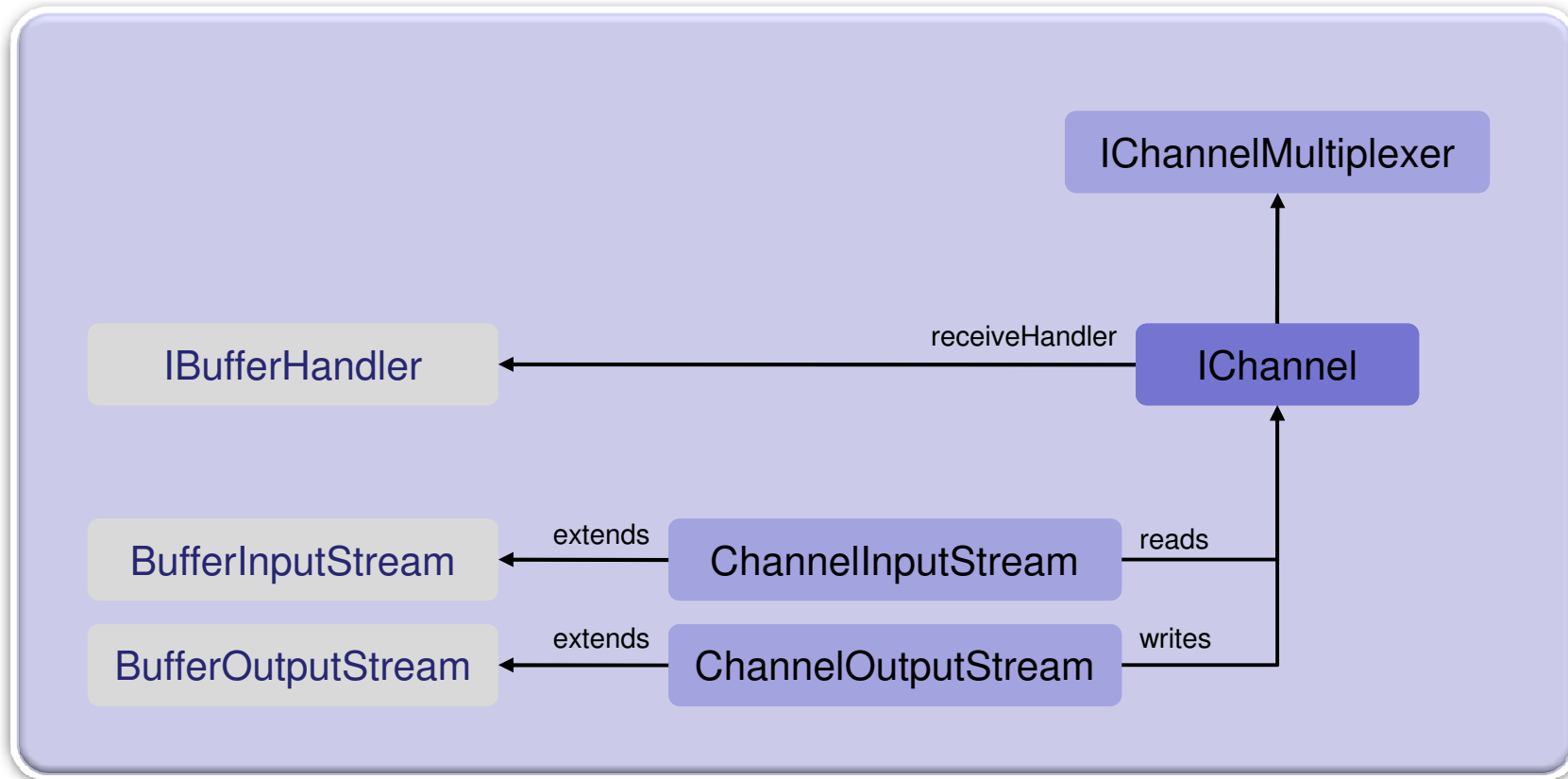
Architecture



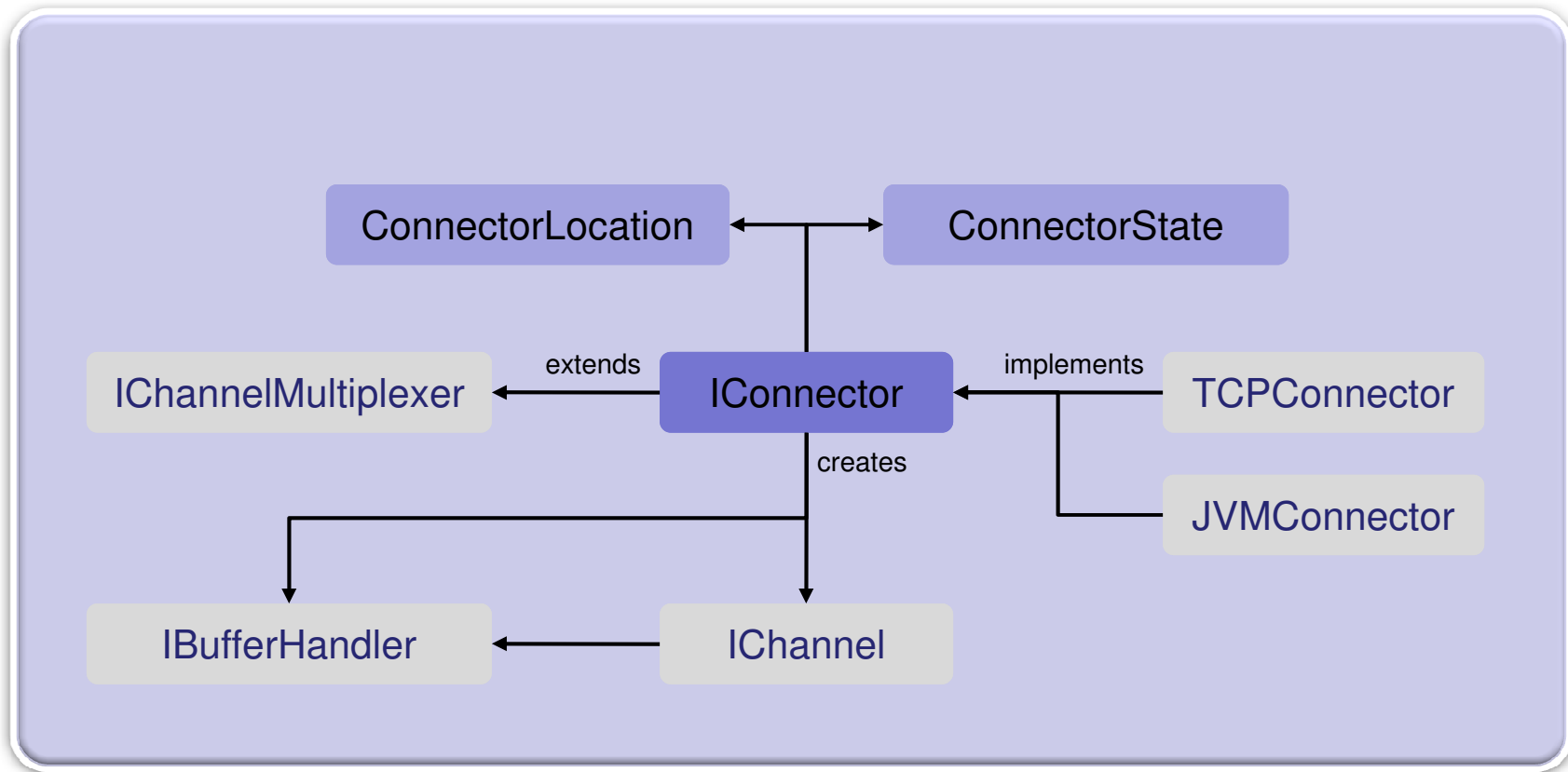
Buffers



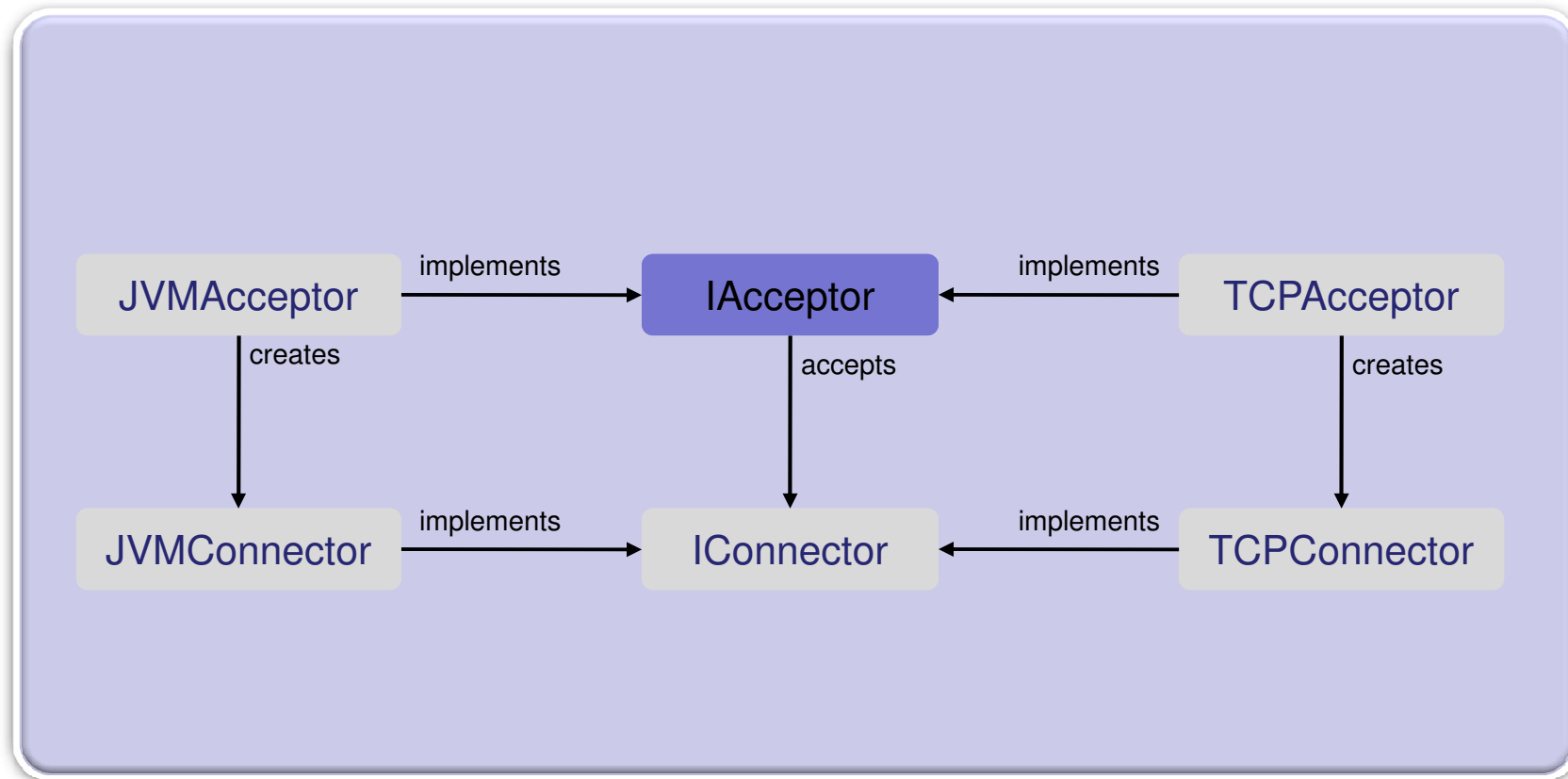
Channels



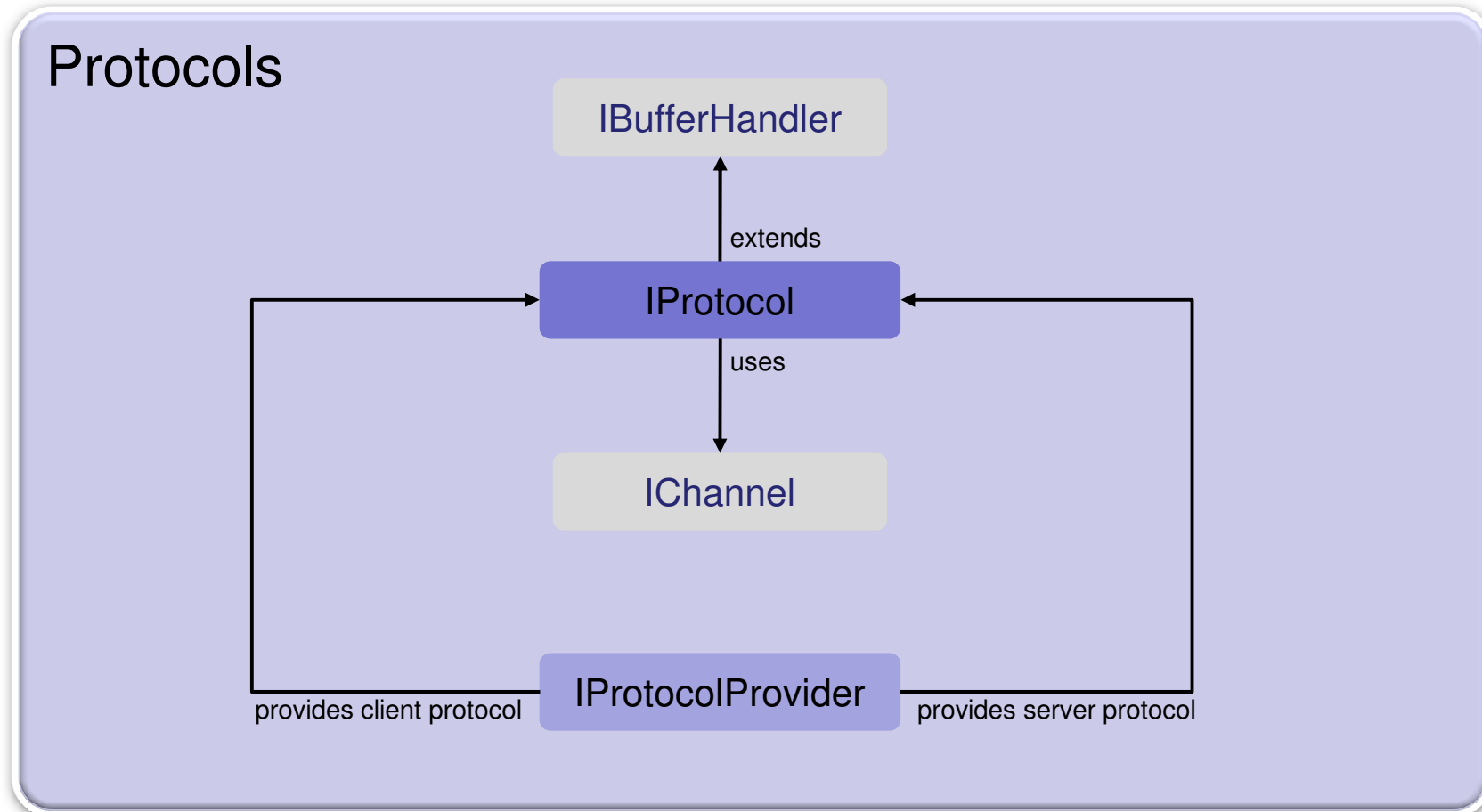
Connectors



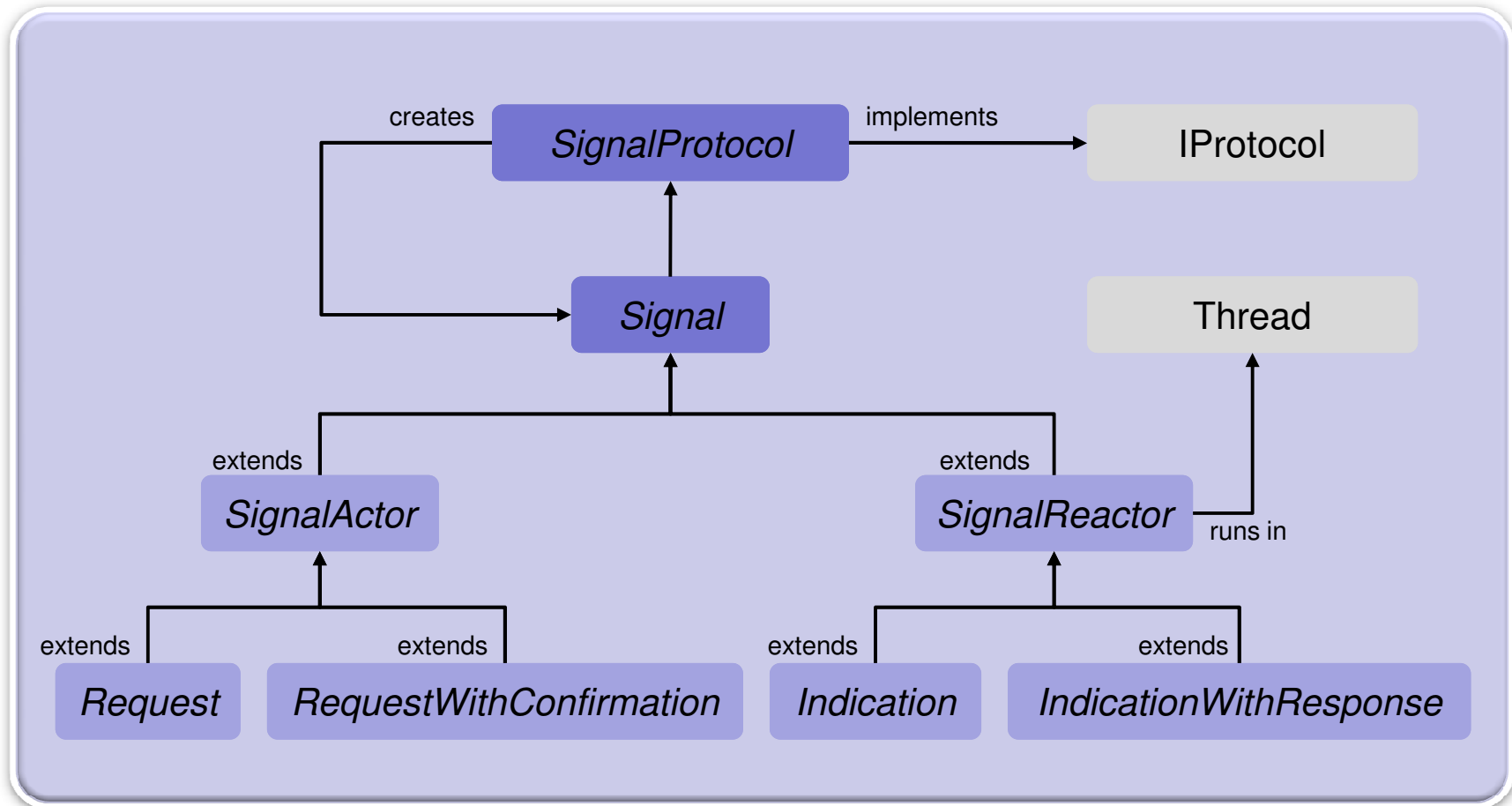
Acceptors



Protocols



Signals



Client Example

```
// 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
JMSClientProtocol protocol = new JMSClientProtocol(infraStructure);
IChannel channel = protocol.open(connector);
channel.addListener(channelListener);

// Create a logon request and send it through the channel
JMSLogonRequest request = new JMSLogonRequest(protocol, "user", "pw");
boolean ok = request.send();
```

Request Example

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

    public JMSLogonRequest(JMSClientProtocol protocol, String userName, String password) {
        super(protocol);
        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();
    }
}
```

Indication Example

```
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);
    }
}
```

SignalProtocol Example

```
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;
    }
}
```

ProtocolFactory Example

```
public final class JMSServerProtocolFactory extends ServerProtocolFactory
{
    public static final String TYPE = JMSProtocolConstants.PROTOCOL_NAME;

    public JMSServerProtocolFactory()
    {
        super(TYPE);
    }

    public JMSServerProtocol create(String description)
    {
        return new JMSServerProtocol();
    }
}
```