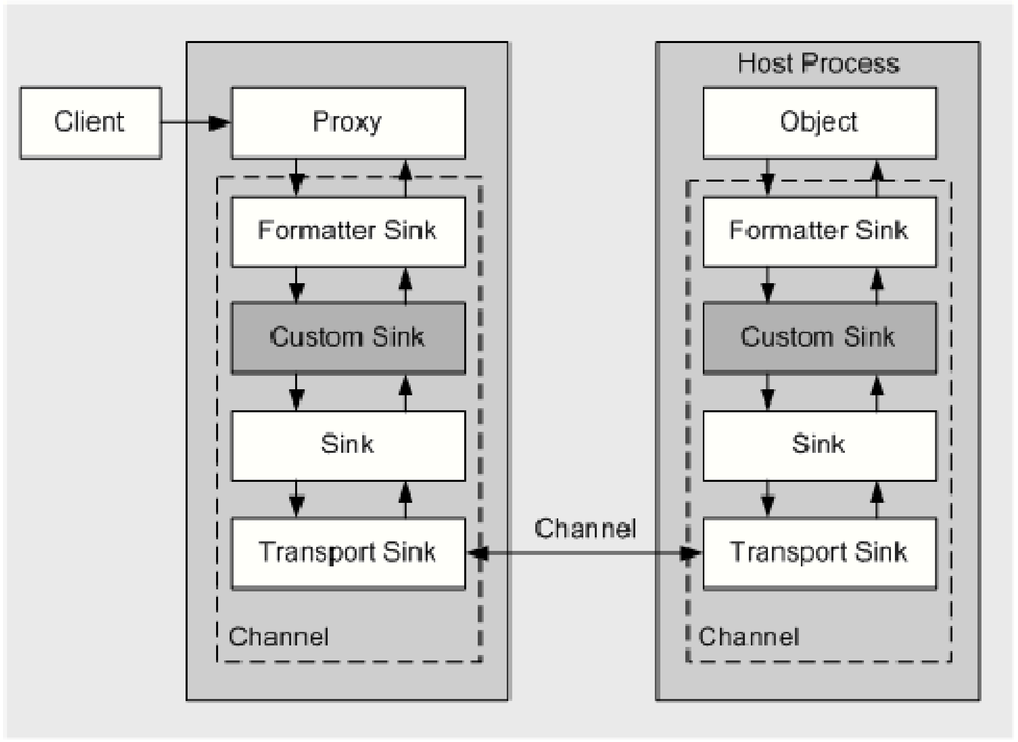
.NET Remoting

IST – MEIC-A, MEIC-T, MERC

.NET Remoting

- Inter-process communication:
  - Method invocation in remote objects
  - Any object can be called remotely
  - Remote objects must derive from `MarshalByRefObject`



Channels

- .Net Remoting communication is done using channels.
- Channels use data streams to:
  - Package information according to a protocol;
  - Send that packet to another computer.
- Channels can be unidireccional or bidireccional
- Two types of channels:
  - **`TcpChannel`** (binary format over TCP)
  - **`HttpChannel`** (XML over HTTP)

Proxies

- Proxies are local objects that point to remote objects.
- When a client receives a reference to a remote object it is in fact receiving a reference to a local proxy object created by .Net Remoting.
- Proxies transform local invocation into network calls to remote objects.

Use of Remote Objects

- Remote objects must be activated:
  - Activation types:
    - **Client activation:** The object on the server is created when the client creates a new instance;
    - **Server activation:** The server objet is created by request. It is created only when a method invocation arrives at the server.

Server Activation

- Two modes of server activation:
  - **`Singleton`**:
    - At a given moment, one instance only;
    - Requires synchronization of shared state.
  - **`SingleCall`**:
    - One instance per call;
    - After a remote call, another instance will be created.

Object Lifetime

- SingleCall lifetime: trivial, one call.
- Singleton lifetime is determined by leases:
  - Leases are timer based;
  - Leases are more reliable than reference counting;
  - When a lease expires, the object memory is reclaimed by the garbage collector.
- Use previously instantiated objects.

Marshalling

- How objects passed are on the Channel:
  - Marshal by Reference (MBR): reference
    - Derive from `MarshalByRefObject`
  - Marshal by Value (MBV): copy
    - `[Serializable]` attribute before class declaration;
    - TCPChannel serializes whole type;
    - HTTPChannel serializes public attributes and properties;
    - Remote call parameters must be MBV.

Summary of Remoting Options

- Channels: `TcpChannel` vs. `HttpChannel`
- Activação: Cliente vs. Servidor, `Singleton` vs. `SingleCall`
- Lifetime: single call vs. Leases
  • MarshallByRef vs. MarshallByValue
  – MBV: Objectos serializáveis, são copiados para o cliente
  – MBR: Cliente obtém proxy que encaminha chamadas

Example: Server

```
class Server {
  static void Main() {
   TcpChannel channel = new TcpChannel(8086);
     ChannelServices.RegisterChannel(channel);
     RemotingConfiguration.RegisterWellKnownServiceType(
        typeof(MyRemoteObject),
        “MyRemoteObjectName",
        WellKnownObjectMode.Singleton);
     System.Console.WriteLine(“Press <enter> to exit...");
     System.Console.ReadLine();
  }
}

public class MyRemoteObject : MarshalByRefObject  {
    public string Hello() {
      return “Hello!”;
    }
  }
```

Example: Cliente

```
class Client {
  static void Main() {
    TcpChannel channel = new TcpChannel();
    ChannelServices.RegisterChannel(channel);
    MyRemoteObject obj = (MyRemoteObject)
   Activator.GetObject(
           typeof(MyRemoteObject),
           "tcp://localhost:8086/MyRemoteObjectName");

    if (obj == null)
      System.Console.WriteLine("Could not locate server");
    else
      Console.WriteLine(obj.Hello());
  }
}
```

Example: Server (preexisting object)

```
class Server {
  static void Main() {
    TcpChannel channel = new TcpChannel(8086);
    ChannelServices.RegisterChannel(channel);
    MyRemoteObject mo = new MyRemoteObject();
    RemotingServices.Marshal(mo,“MyRemoteObjectName”,
      typeof(MyRemoteObject));
    System.Console.WriteLine("<enter> para sair...");
    System.Console.ReadLine();
  }
}

public class MyRemoteObject : MarshalByRefObject  {
  public string Hello() {
    return “Hello!”;
  }
}
```