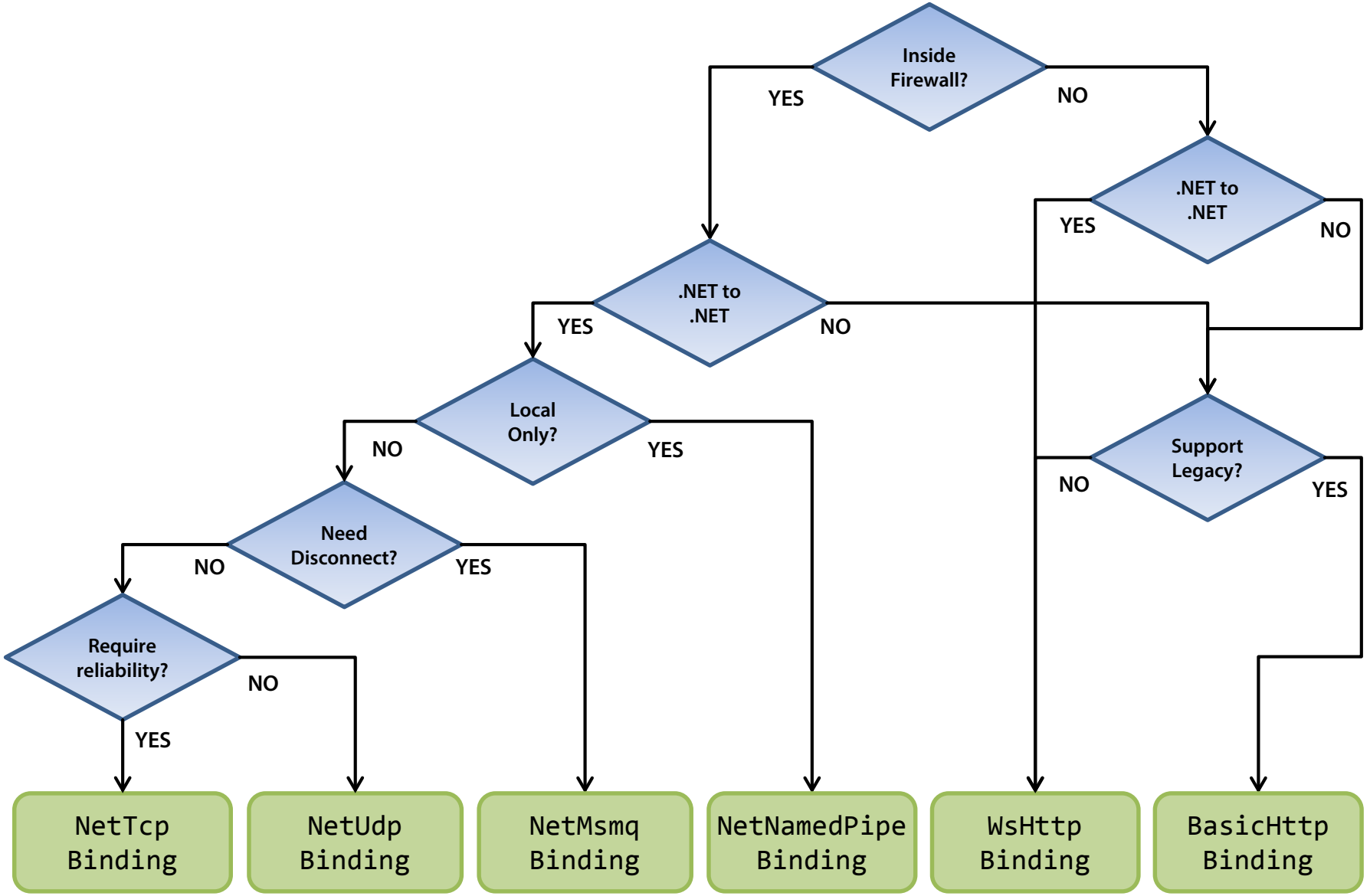
Bindings and Behaviors

Miguel A. Castro
@miguelcastro67

What Are Bindings

- **Bindings define the transport mechanism for messages**
- **Different bindings have different characteristics and uses**
- **HTTP bindings**
  - Can travel on port 80 and through a firewall
  - Are secured and encrypted using certificates
- **TCP bindings**
  - Typically used inside firewall
  - Fast and secure
- **IPC bindings**
  - Fastest bindings but are limited to single machine
- **UDP bindings**
  - Non-reliable but faster than TCP
  - Limited to subnet without bridging
- **MSMQ bindings**
  - Use MSMQ as transport using queuing so can work disconnected

Choosing a Binding



Transport-Level Sessions

- **Transport sessions allow service to identify its clients**
  - Kind of like a hot, open connection between service and client
- **Some bindings support transport sessions**
  - netTcpBinding
  - netNamedPipeBinding
  - wsHttpBinding
    - With Reliability or Security turned on – *to be explained, stand by*
- **Later, during Instancing, will discuss Transport Session further**

Binding Configurations

- **Characteristics of the transport**
- **Set on both sides of the wire**
- **Possible settings**
  - Reliability
  - Ordered messaging
  - Inactivity timeout
  - Message timeout
  - Message size
  - Transaction flow

Binding Configuration Syntax (Declarative)

```
<services>
   <service name="MyService">
      <endpoint address="net.tcp://localhost:8008/MyService
                binding="netTcpBinding"
                contract="IMyContract"
                bindingConfiguration="binding1" />
   </service>
</services>
<bindings>
   <netTcpBinding>
      <binding name="binding1" additional attributes here>
            <additional tags here />
      </binding>
   </netTcpBinding>
</bindings>
```

Note the attributes with binding name establishing the link.

Each binding type carries its own static configuration tag.

Binding Configuration Syntax (Procedural)

```
NetTcpBinding binding = new NetTcpBinding();
binding.{property} = {value};

Binding object used in programmatic endpoint
```

Useful Binding Configurations

- **Reliability**
  - End-to-end message transfer reliability. Overcomes potential transport failures across some networks
  - Provides for non-TCP bindings what TCP offers out-of-the-box
- **Ordered messaging**
  - Ensures order of message servicing equals order of calls. Useful in one-way calls across Internet
- **Inactivity timeout**
  - Sliding value specifying time reliable transport session will remain open after not receiving messages
    - Supports Infinite
  - Transport session only available under certain conditions
    - TCP, IPC, WS-HTTP (with Security or Reliability turned on)

Useful Binding Configurations

- **Receive timeout**
  - Sliding value specifying time non-reliable session will remain open after not receiving messages
    - Supports Infinite
- **Send timeout**
  - Specifies time a call will wait for message to be processed
- **Message size**
  - Specifies maximum size of SOAP message

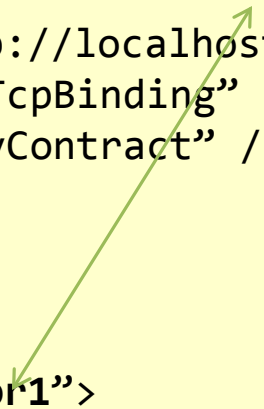
Demo Time

Behavior Configurations

- **Characteristics of a service**
- **Set only on the service side**
- **Client unaware**
  - Exception is client developer should know what to expect
- **Some set declaratively and some inline**
- **Possible settings**
  - Exception details
  - Metadata exposure
  - Instancing
  - Concurrency
  - Throttling

Behavior Configuration Syntax *(Config)*

```
<service name="MyService"
            behaviorConfiguration="behavior1">
   <endpoint address="net.tcp://localhost:8008/MyService
               binding="netTcpBinding"
               contract="IMyContract" />
</service>
<behaviors>
   <serviceBehaviors>
      <behavior name="behavior1">
            <additional tags here />
      </behavior>
   </serviceBehaviors>
</bindings>
```



Behavior Configuration Syntax *(Inline)*

```
[ServiceBehavior(IncludeExceptionDetailsInFaults = True)]
   public class MyService : IMyService
{
   … implementation code here …
}
```

Behavior Configuration Syntax *(Procedural)*

```
Behavior behavior = host.Description.Behaviors.Find<{type}>();
if(behavior == null)
{
   behavior = new {behavior type};
   behavior.{property} = {value};
   host.Description.Behaviors.Add(behavior);
}
```

Useful Behavior Configurations

- **Exception details**
  - Provides a little more detail on unhandled SOAP faults
- **Throttling**
  - Allows manipulation of values that prevent server snap and memory overload
- **Instancing & Concurrency**
  - Determines how the service is instantiated and how it will handle thread locking
  - This will covered in its entirety in upcoming module

Demo Time