WCF Advanced Routing

Scott Seely
http://www.pluralsight.com/

Outline

- Using Routing Filters to define destinations
- Creating custom filters
- Handling Failover
- One-way Multicast

Routing Filter Types

```
<routing>
  <filters>
    <filter name="HelloWorld" filterType="EndpointName"
        filterData="helloWorld" />
  </filters>
</routing>
```

- **Format is:**
  - name: name of filter. Used to build a routing table.
  - filterType: Type of filter.
  - filterData: String to pass to the constructor of the filter.
  - filter1/filter2: Only used with the 'And' filter (more in a moment!)
  - custom: Type name that implements MessageFilter

EndpointName Filter

- Matches all messages arriving on the RoutingService for a given endpoint
- Fairly useful to map 1:1 from endpoint to service

```
<services>
  <service behaviorConfiguration="routingData"
           name="System.ServiceModel.Routing.RoutingService">
    <endpoint address="/helloWorld" binding="wsHttpBinding" name="helloWorld"
              contract="System.ServiceModel.Routing.IRequestReplyRouter"/>
  </service>
</services>
<filter name="EndpointName" filterType="EndpointName" filterData="helloWorld" />
```

EndpointAddress

- Matches all messages arriving at a hosted URL
- Like EndpointName, only address based.

```
<services>
  <service behaviorConfiguration="routingData"
           name="System.ServiceModel.Routing.RoutingService">
    <endpoint address="/helloWorld" binding="wsHttpBinding" name="helloWorld"
              contract="System.ServiceModel.Routing.IRequestReplyRouter"/>
  </service>
</services>
<filter name="EndpointAddress" filterType="EndpointAddress"
        filterData="http://localhost:30866/router.svc/helloWorld" />
```

PrefixEndpointAddress

- Matches all messages arriving at a base URL
- Useful for mapping a family of prefixes for contracts implementing one or more MEPs to a single location.

```
<filter name="PrefixEndpointAddress" filterType="PrefixEndpointAddress"
           filterData="http://localhost:30866/router.svc/" />
```

XPath

- Matches all messages based on XPath expression
- Useful for mapping messages based on Header information.
- Never get access to message Body, so no CBR on Body.

```
 <namespaceTable>
   <add prefix="soap" namespace="http://www.w3.org/2003/05/soap-envelope"/>
   <add prefix="wsa" namespace="http://www.w3.org/2005/08/addressing"/>
 </namespaceTable>

<filter name="XPath" filterType="XPath"
        filterData=
"/soap:Envelope/soap:Header/wsa:Action = 'http://www.pluralsight.com/WCF/HelloW
orldService/SayHello'"/>
```

Others

- **And: Combine 2 named filters, And-ing them to get a result**

```
<filter name="And" filterType="And" filter1="XPath" filter2="HelloWorld"/>
```

- **MatchAll: Match all messages that come through**

```
<filter name="MatchAll" filterType="MatchAll"/>
```

- **Action: Match when the WS-Addressing Action header has a set value**

```
<filter name="Action" filterType="Action"
   filterData="http://www.pluralsight.com/WCF/HelloWorldService/SayHello" />
```

Filter priority

- Each filter in a filterTable has a priority
- Big priority == earlier evaluation
- Same priority, only one should evaluate to *true*

```
<filterTable name="routingTable">
  <add filterName="Action" endpointName="HelloWorldBackup“
    priority="1" />
  <add filterName="HelloWorld"
    endpointName="HelloWorldPrimary" priority="0" />
</filterTable>
```

What about…?

- **Custom: When WCF doesn't provide the filter you want, write your own!**
- **Derive from System.ServiceModel.Dispatcher.MessageFilter**
- **Implement:**
  - bool Match(Message)
  - bool Match(MessageBuffer) ← only called for Buffered messages, good for MessageLogging, not Routing
- **To dynamically load extra filters, implement CreateFilterTable**
  - Created FilterTable shared by all instances of type in a parent FilterTable
  - Can read from your own datastore
  - Be careful– you may be pushing into territory better served by learning BizTalk

Routing and Failover

- Use when you need redundancy
- If service times out or fails, try another, and another, and …
- Handled by backup lists

Backup List Configuration

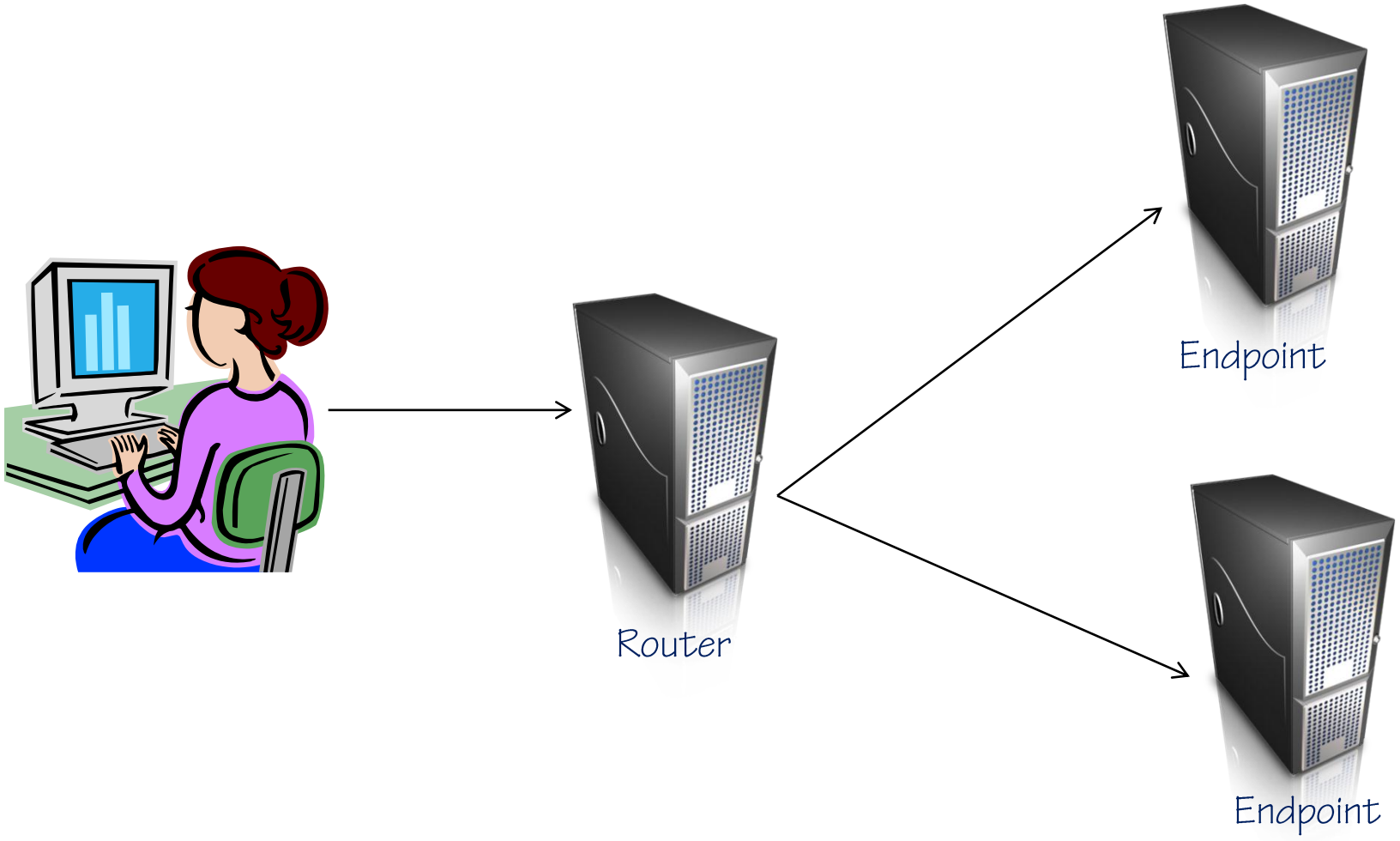
```
<filterTables>
  <filterTable name="routingTable">
    <add filterName="HelloWorld"
      endpointName="HelloWorldPrimary"
      backupList="HelloBackups" />
  </filterTable>
</filterTables>
<backupLists>
  <backupList name="HelloBackups">
    <add endpointName="HelloWorldBackup"/>
  </backupList>
</backupLists>
```

Multicast With Routing

- You can 'multicast' for messages received ISimplexDatagramRouter endpoints
- For each filter in the filter table that matches, the target endpoint gets a message
- Priority still in play!

```
<filterTable name="routingTable">
  <add filterName="OneWayEndpointName" endpointName="OneWayPrimary"
       priority="2" />
  <add filterName="OneWayEndpointName" endpointName="OneWayBackup"
       priority="2" />
</filterTable>
```

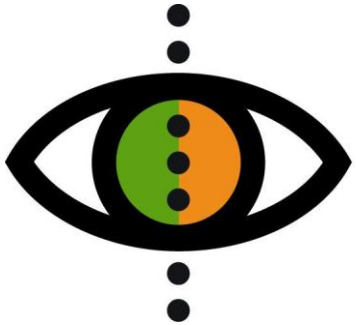
Why multicast works on ISimplexDatagramRouter



Summary

- WCF ships with many built in filters.
- You can build custom filters, create your own message filter tables in code.
- To support failover, create backup lists pointing to other implementations of the service(s)
- For one-way datagram contracts, you can multi-cast messages

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust