Introducing WCF

Next-generation connected systems on Windows

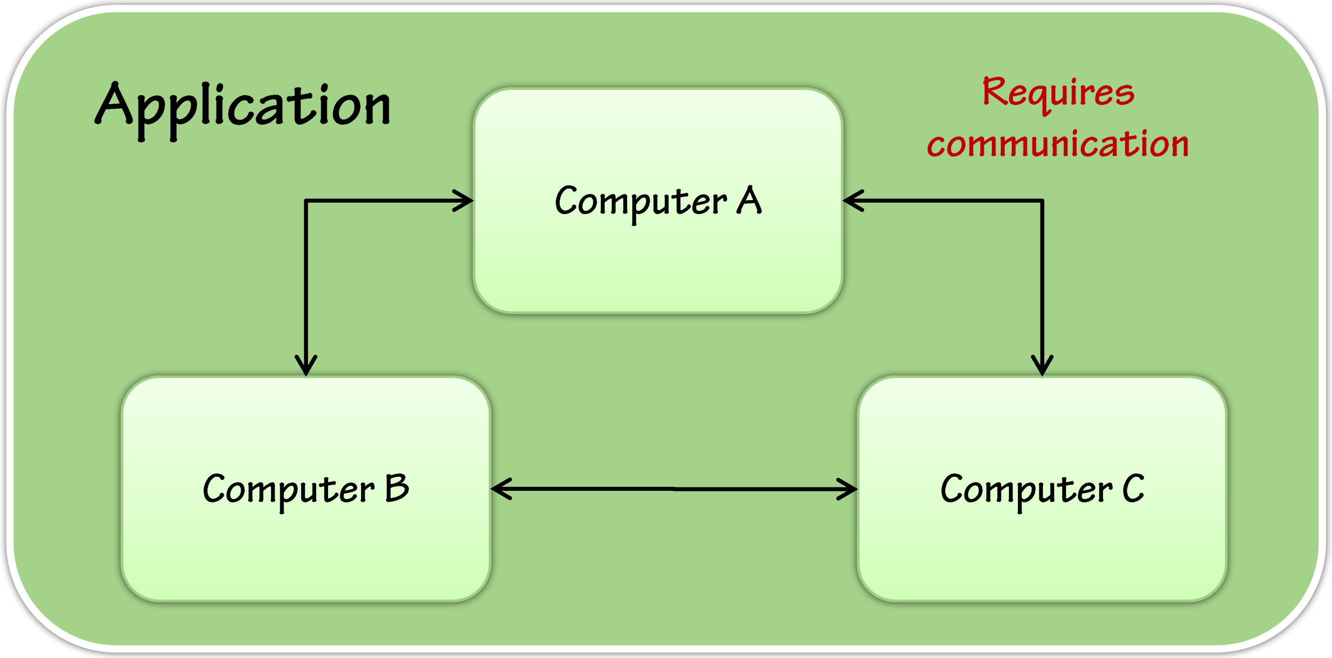
pluralsight
see what you can learn

Outline

- Connected systems overview
- The move towards "services"
- Service-orientation
- Introduction to WCF
- WCF programming model basics
- Common WCF questions

What is a connected system?

- An application that is distributed across multiple computer nodes



Microsoft's new label for distributed applications

Building connected systems on Windows

- MS has shipped many communication frameworks over the years

DCOM/COM+/ES	.NET Remoting	MSMQ
Component-oriented (RPC) Distributed transactions DCOM infrastructure	Component-oriented (RPC) Simple & highly extensible CLR infrastructure	Message-oriented Asynchronous/durable/reliable MSMQ infrastructure

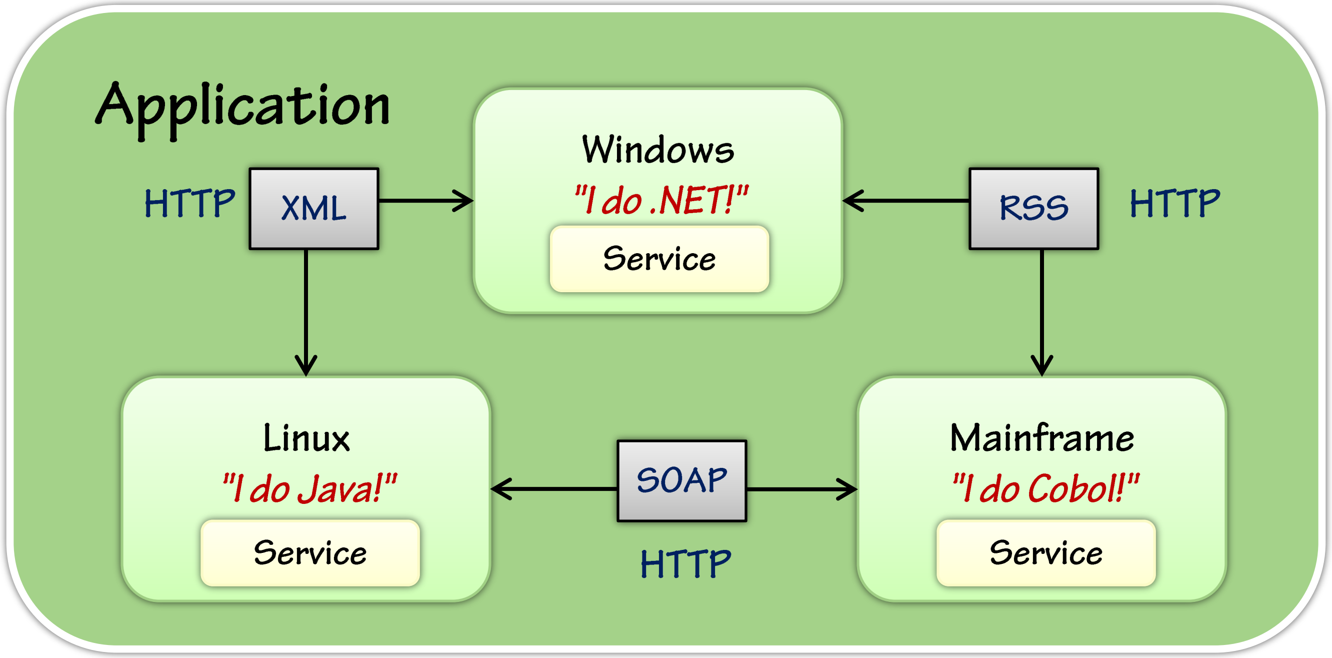
Each framework comes with a unique programming model

Each of these framework restricts you to Windows

The move towards "services"

- Demand for technology freedom and interoperability is common now

Services expose units of functionality via messaging



Interop achieved via standard protocols and message formats

Service design philosophies

SOAP

Typical in the enterprise

XML messaging using SOAP as the format, enhanced with the WS-* protocols, can be used with any transport protocol

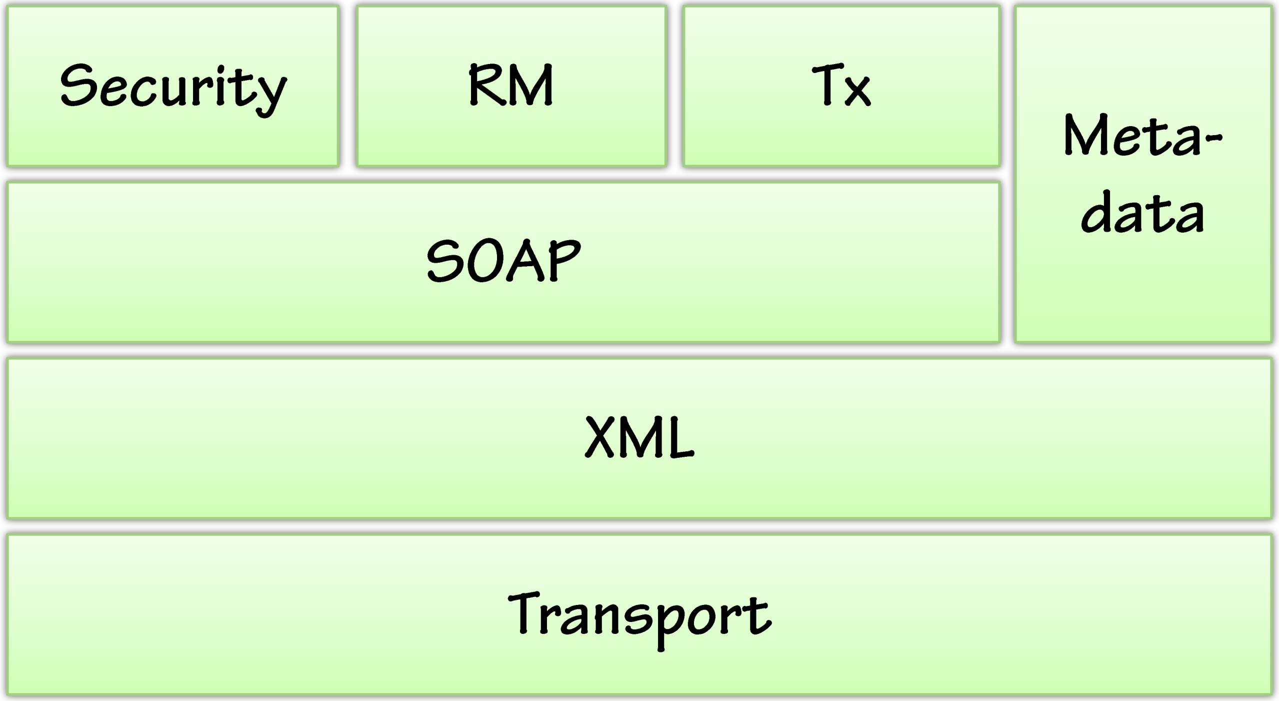
REST

Typical in public-facing Web scenarios

Design paradigm focused on how to identify, represent, and operate on resources through a unified interface (HTTP)

SOAP + WS-* services

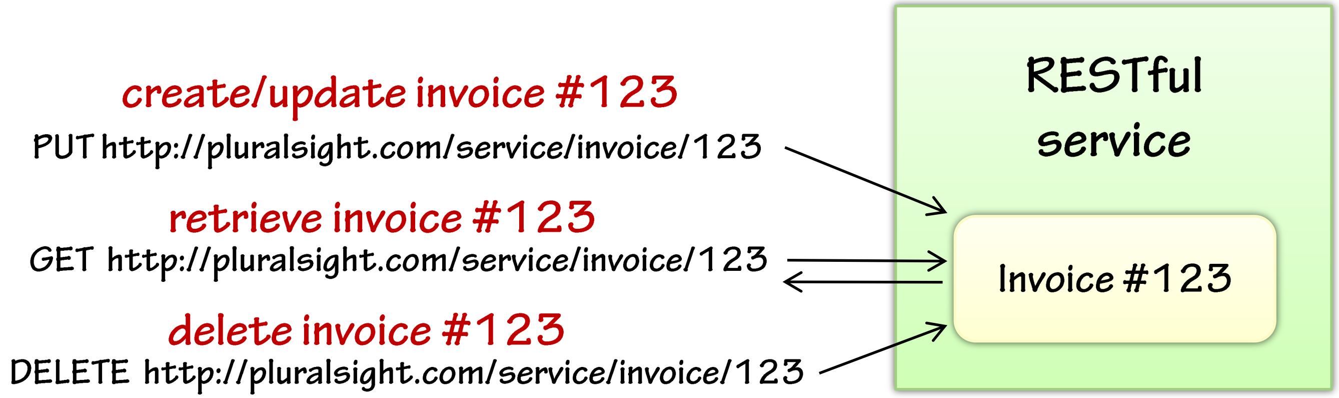
- The industry has defined a complete protocol stack for services



Typically implemented with RPC-based toolkits, feels a lot like COM+

RESTful services

- **RESTful services typically embrace HTTP, the "Web" transport**
  - Services are modeled as "resources" with unique identifiers (URI's)
  - HTTP defines a uniform service contract: GET, POST, PUT, DELETE, HEAD
  - Resources can be represented as XML, RSS, JSON, etc
- **HTTP provides the necessary features and scalability**
  - A successful design pattern used throughout the Web today


create/update invoice #123
PUT http://pluralsight.com/service/invoice/123
retrieve invoice #123
GET http://pluralsight.com/service/invoice/123
delete invoice #123
DELETE http://pluralsight.com/service/invoice/123
RESTful service
Invoice #123

Service-orientation

- **Service-orientation is a design paradigm for separation of concerns**
  - Focused on autonomy, explicit boundaries, contracts & policies
  - Design principles help achieve a Service Oriented Architecture (SOA)
  - SOA says nothing about technology – room for both SOAP & REST

"SOA" as defined by OASIS

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities…

Microsoft's first attempts at "services"

- Microsoft has shipped a few different Web services frameworks

ASP.NET Web Services (ASMX)

Basic XML over HTTP
Simple support for SOAP 1.1/1.2

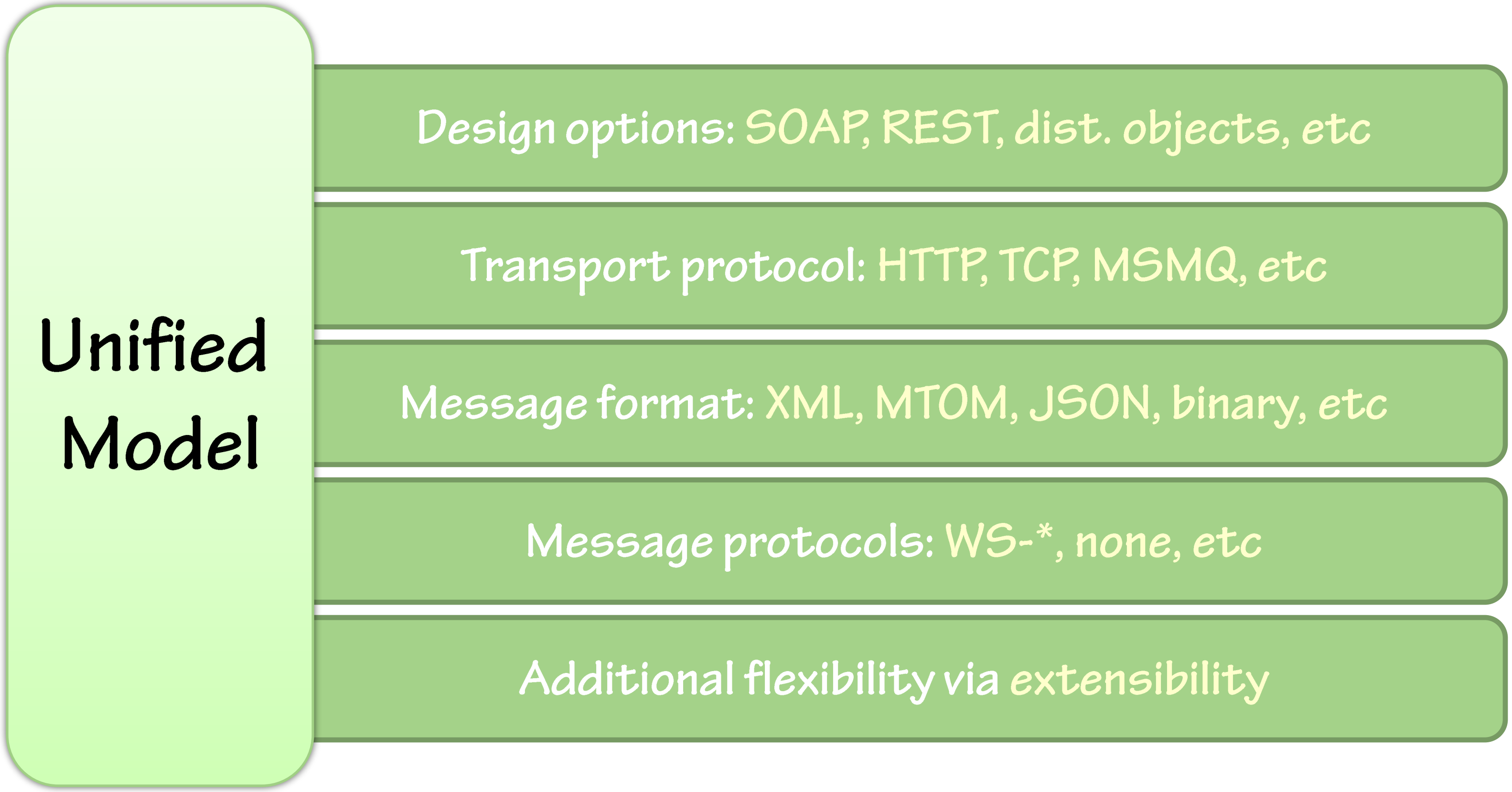
Both very limited

Web Services Enhancements (WSE)

Added support for WS-Security et al and TCP services

Each framework comes with a unique programming model

The ideal communication framework

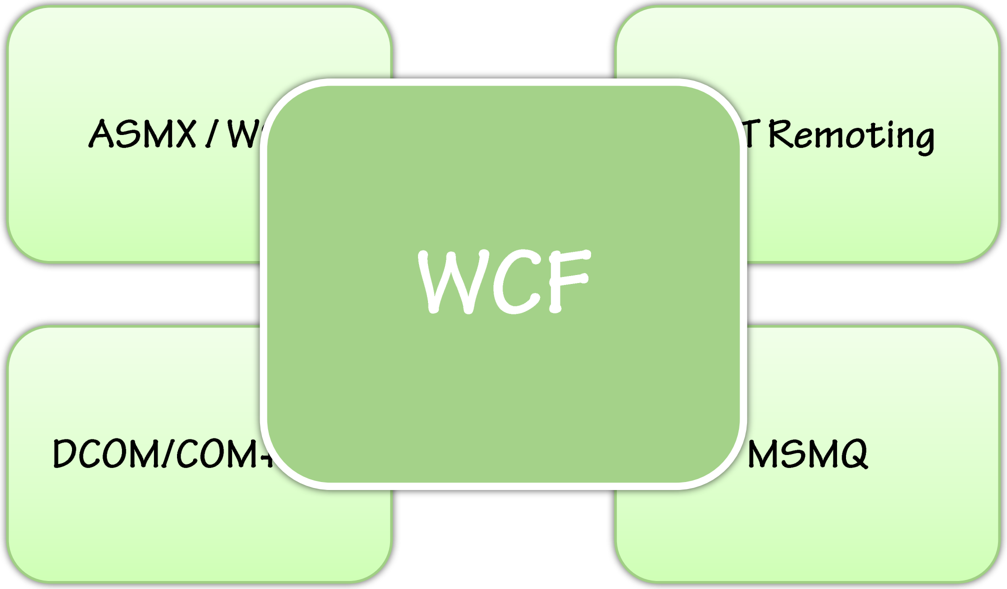

Unified Model
Design options: SOAP, REST, dist. objects, etc
Transport protocol: HTTP, TCP, MSMQ, etc
Message format: XML, MTOM, JSON, binary, etc
Message protocols: WS-*, none, etc
Additional flexibility via extensibility

Enter Windows Communication Foundation

Introducing WCF

- WCF is the new unified "communications" framework for Windows

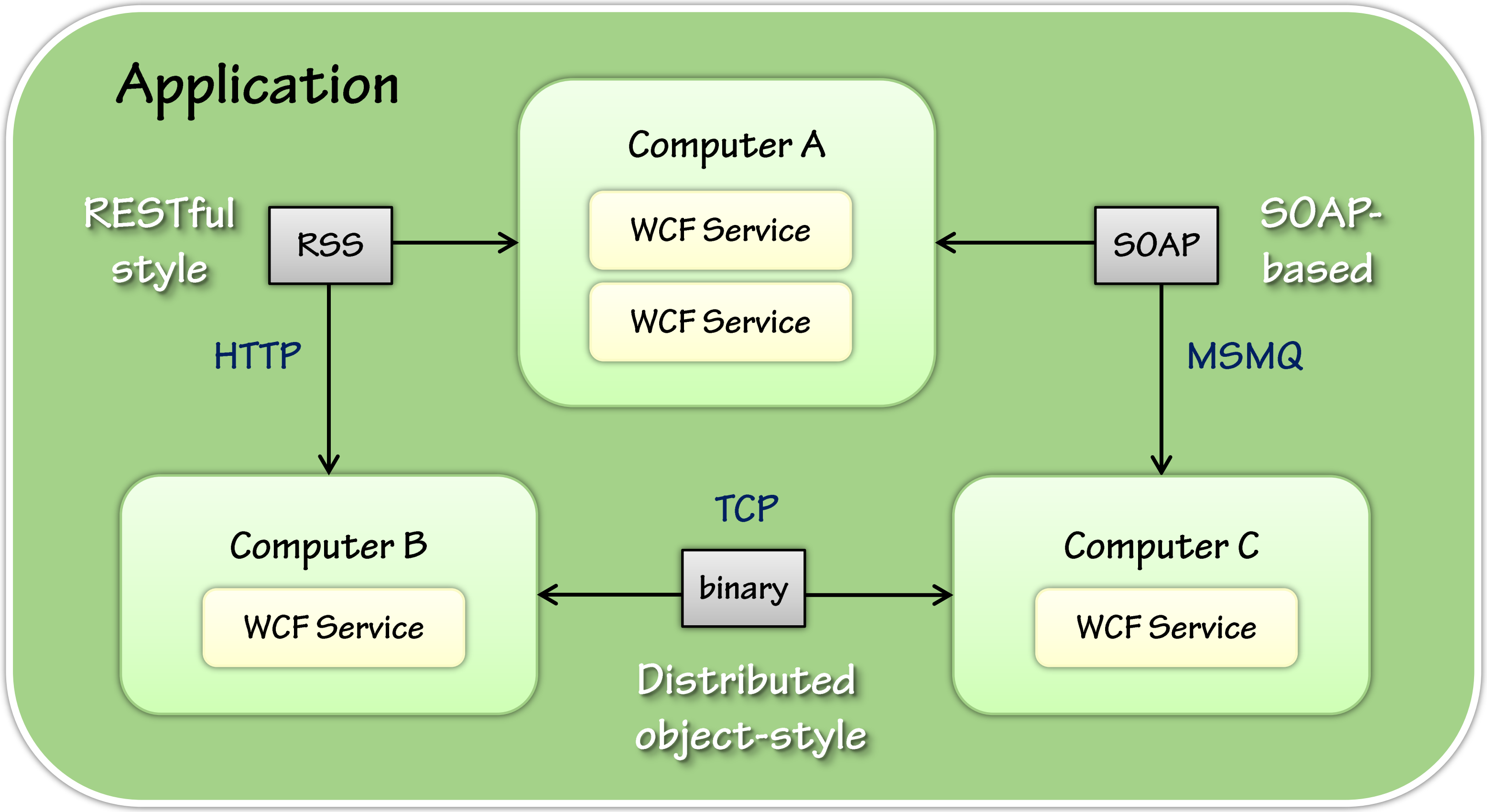
Becomes the default choice for connecting apps today



Functionality mostly found in System.ServiceModel.dll

The WCF experience

Just one way to write the code



But many ways to connect-the-dots

Just one way to write the code

```
[DataContract]
public class Invoice {
    [DataMember]
    public string CustomerId;
    [DataMember]
    public string InvoiceDate;
    [DataMember]
    public double Amount; ...
}
```

Defines communication contracts via attributes

```
[ServiceContract]
public interface IInvoiceService {
    [OperationContract]
    void SubmitInvoice(Invoice invoice);
}
```

Defines business logic

```
public class InvoiceService : IInvoiceService {

   public void SubmitInvoice(Invoice invoice) {
      ... // implementation omitted
   }
}
```

But many ways to connect-the-dots

- You configure endpoints to define different communication options

REST over HTTP →

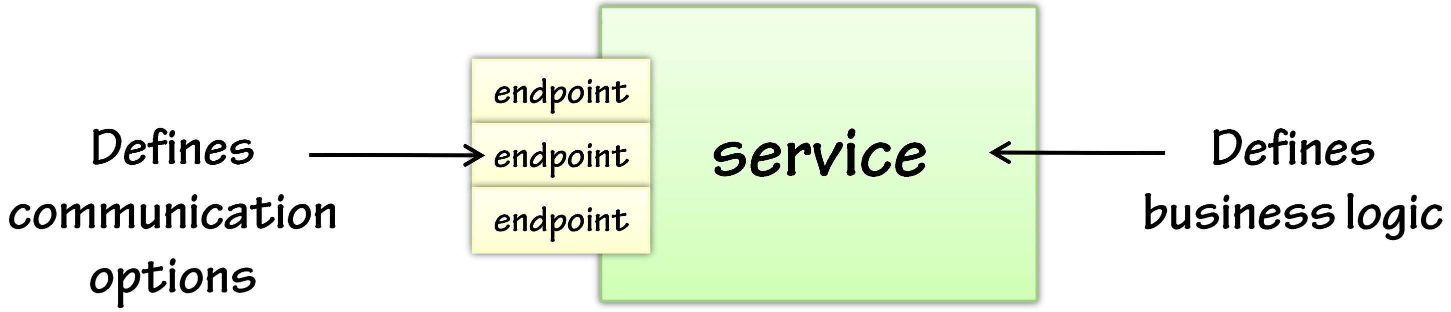
SOAP + WS-* over MSMQ →

SOAP + WS-* over TCP →

```
<configuration>
  <system.serviceModel>
    <services>
      <service name="InvoiceService">
         <endpoint
            address="http://server/invoiceservice"
            binding="webHttpBinding"
            contract="IInvoiceService"/>
         <endpoint
            address="net.msmq://server/invoicequeue"
            binding="netMsmqBinding"
            contract="IInvoiceService"/>
         <endpoint
            address="net.tcp://server:8081/invoiceservice"
            binding="netTcpBinding"
            contract="IInvoiceService"/>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

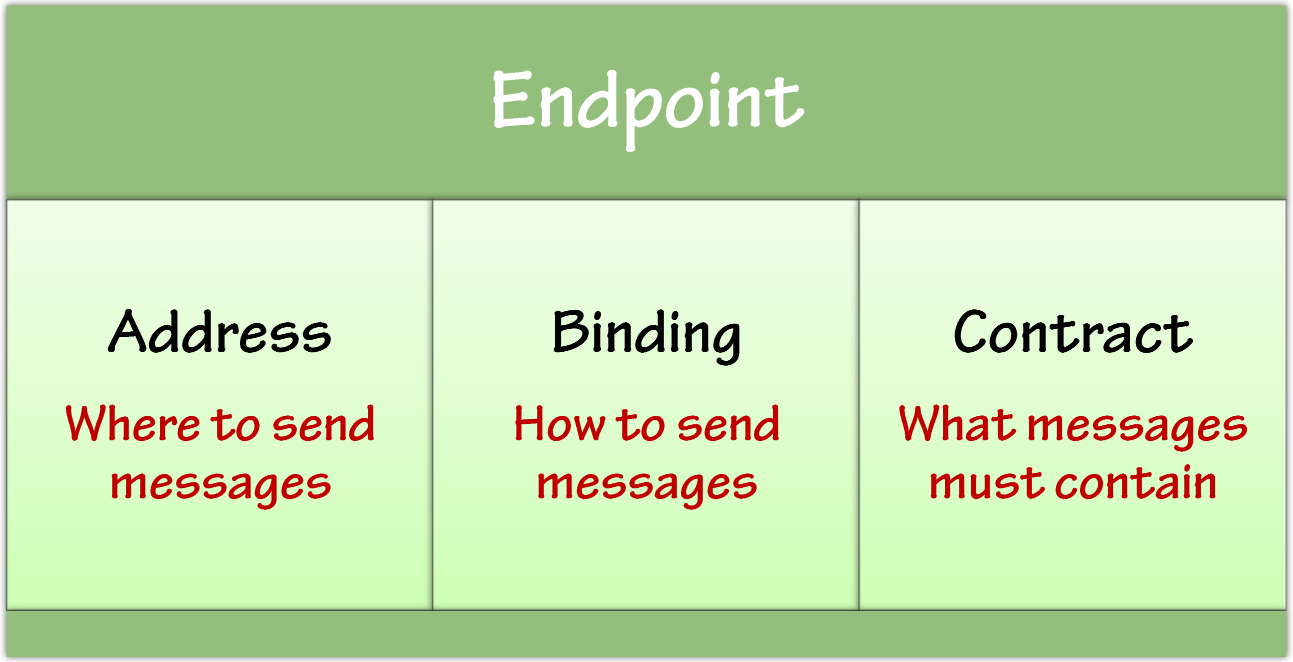
Services and endpoints

- **With WCF, you write services that expose endpoints to the world**
  - Service implementation defines business logic
  - Endpoints define the communication options
  - Services can expose multiple endpoints for consumers



What is an endpoint?

- Endpoints tell WCF how to build the runtime communication channels



Services *expose* endpoints while clients *consume* them

WCF's built-in bindings

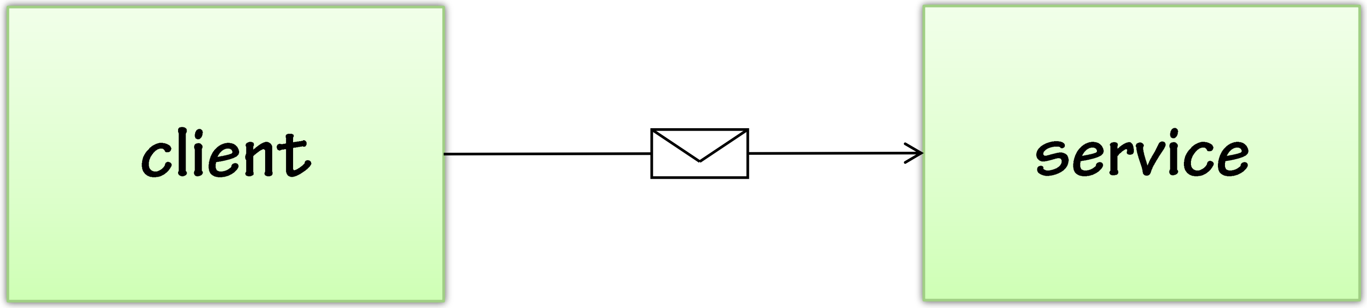
- WCF provides built-in bindings for common communication scenarios

Binding Name	Communication Scenario
WebHttpBinding	Interoperable RESTful communication via HTTP
BasicHttpBinding	Interoperable SOAP communication via HTTP, offering only the "basic" protocols conforming to WS-I Basic Profile
WSHttpBinding	Interoperable SOAP communication via HTTP, offering the full range of SOAP + WS-* protocols
NetTcpBinding	Cross-machine WCF communication via TCP
NetPeerTcpBinding	Cross-machine WCF communication via P2P
NetNamedPipesBinding	Same-machine WCF communication via IPC
NetMsmqBinding	Disconnected/asynchronous WCF communication via MSMQ

NetXXX bindings designed for .NET-to-.NET communication

Consuming services with WCF

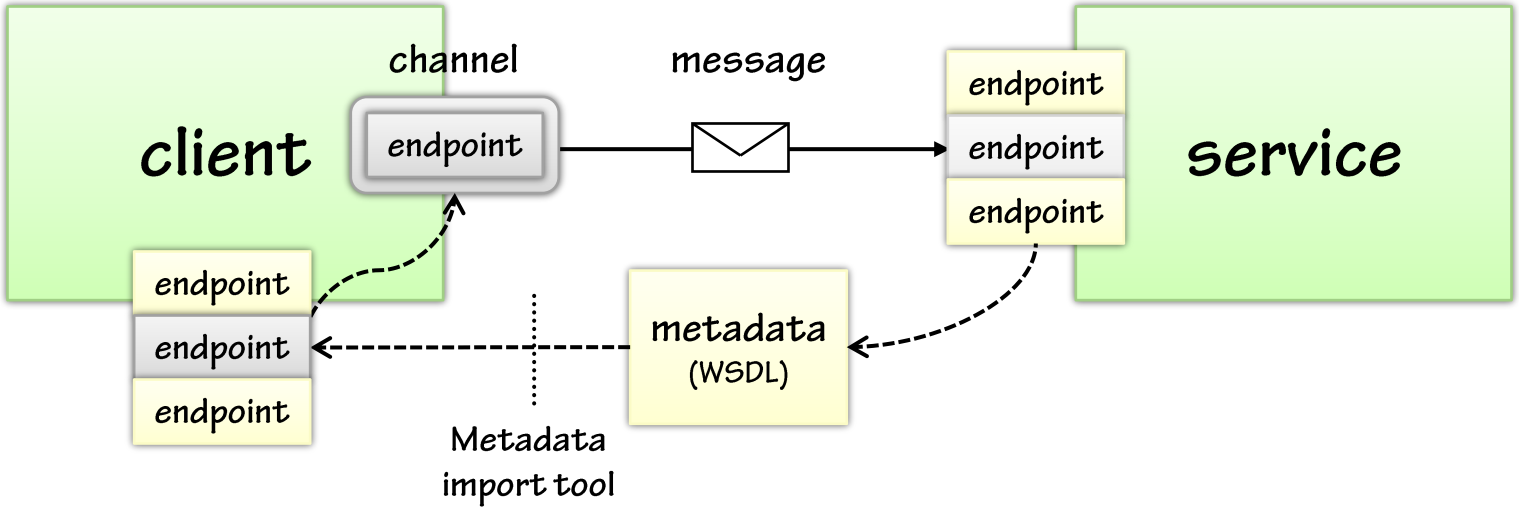
- **Clients need to know several things in order to consume a service**
  - Where to send the message (address)
  - How to send the message, such as what transport/protocols to use (binding)
  - What the messages should contain (contract)



WCF answers these questions via *endpoints*

WCF clients

- **With WCF, you consume services via channels based on endpoints**
  - Clients retrieve endpoint definitions from service metadata

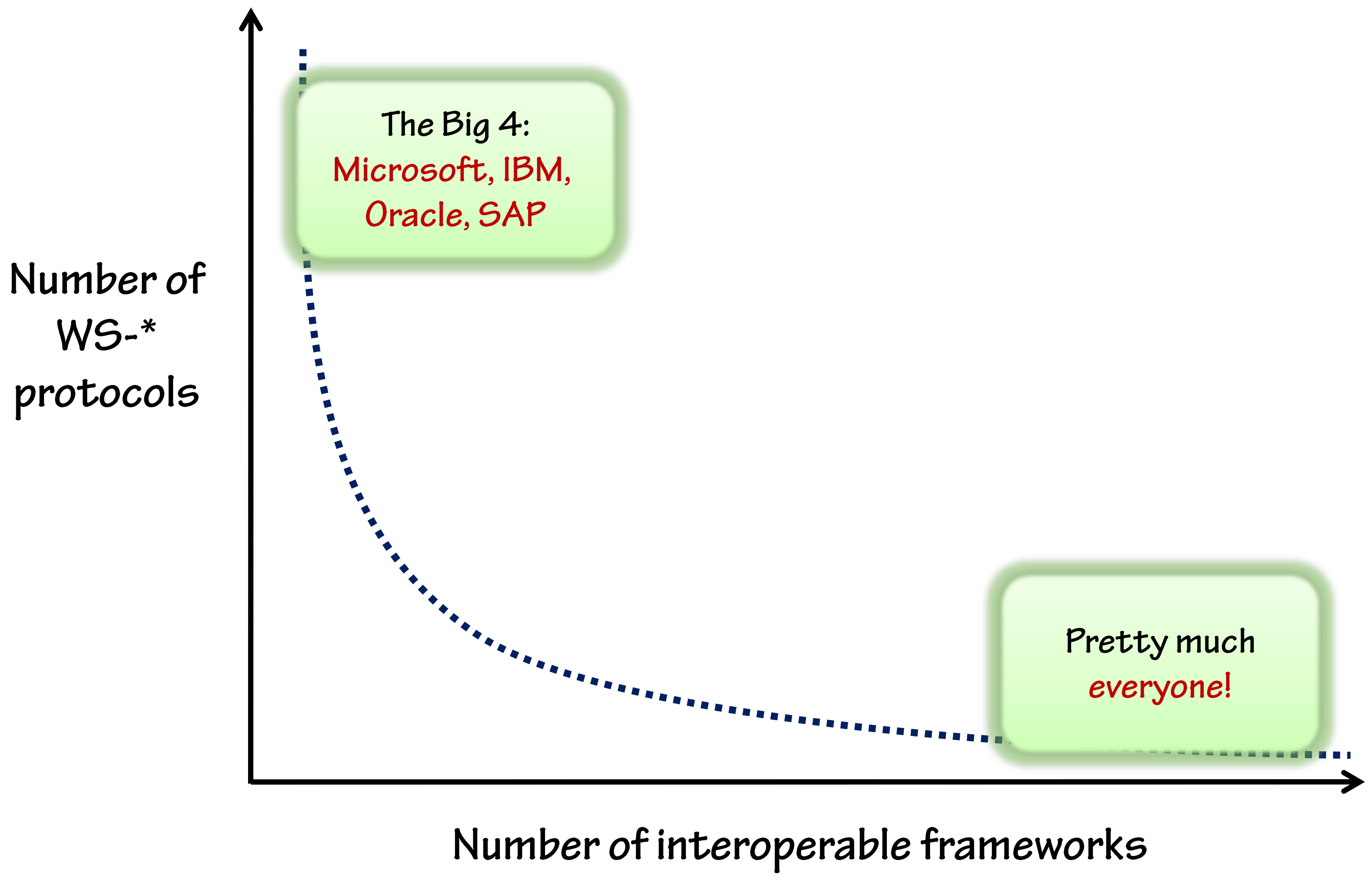


Endpoints provide symmetry across clients/services

Some common WCF questions

- How far can WCF reach?
- What about my existing code?
- What does WCF run on?
- Why should I move towards WCF?

How far can WCF reach?

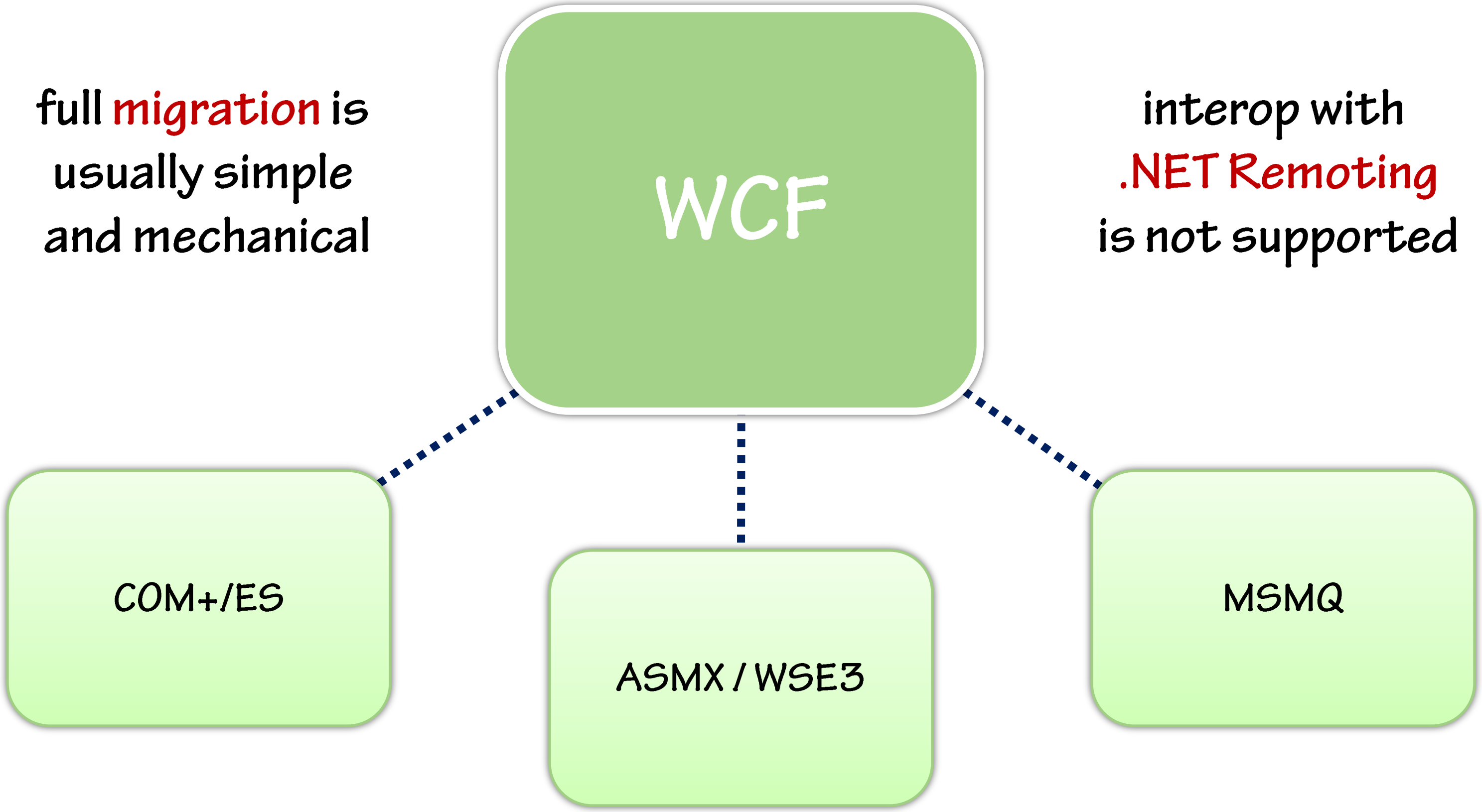


What about my existing code?

- WCF also integrates (on the wire) with some key MS frameworks

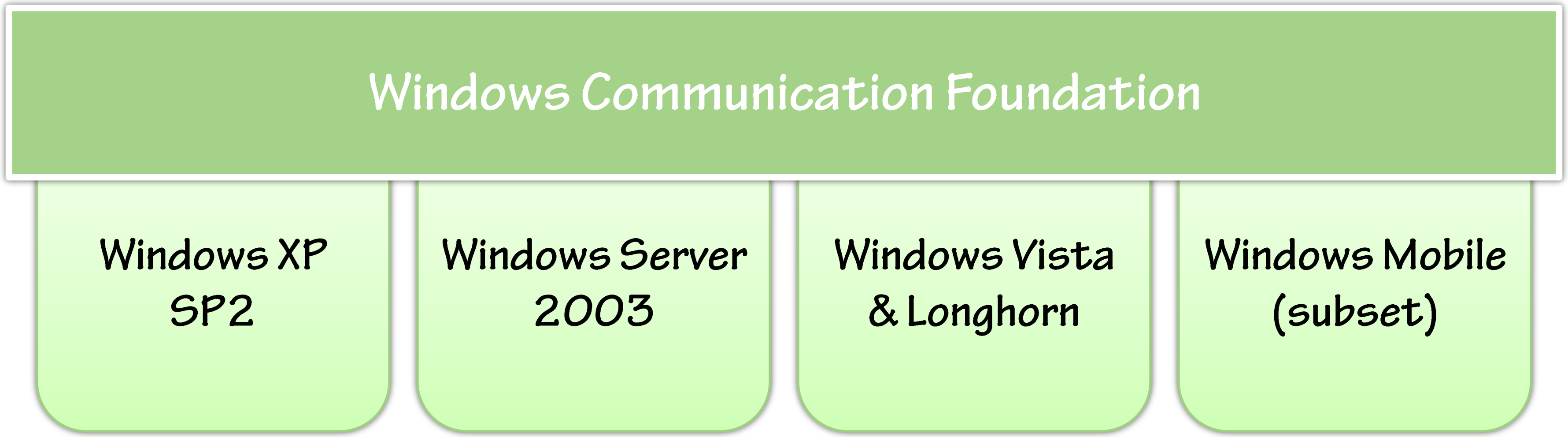
full migration is usually simple and mechanical

interop with .NET Remoting is not supported



What does WCF run on?

Today's modern Windows platforms



Why should I move towards WCF?

Increase productivity

Increase interoperability

Increase flexibility

You can adopt piecemeal

Microsoft's future work focused here

Summary

- **WCF provides a unified model with flexibility in communications**
  - Your choice of architecture, transport, message format, protocols, etc
  - Replaces the need for the preceding Windows frameworks
- **There are several good reasons to begin moving towards WCF**
  - It provides a simpler model that will increase productivity/reach
  - Microsoft has positioned it as the "DCOM" of the next decade

References

- **Introducing WCF, Chappell**
  - http://www.davidchappell.com/IntroducingWCFv1.2.1.pdf
- **What is Windows Communication Foundation?**
  - http://msdn2.microsoft.com/en-us/library/ms731082.aspx
- **Pluralsight's WCF Wiki**
  - http://pluralsight.com/wiki/default.aspx/Aaron/WindowsCommunicationFoundationWiki.html