Chapter 2
Inter-Process Communication

Mastering Cloud Computing

Coleman Kane

(based on materials from Paul Talaga)

UNIVERSITY OF Cincinnati

Interprocess Communication (IPC)

- How different programs/process can communicate
- Use a client/server model
- Built using network sockets
  - Allow traversal of unknown networks due to network stack

UNIVERSITY OF Cincinnati

Types of Message Passing

- Message Passing: pass plain messages
  - Ex: Message-passing interface (MPI) and OpenMP - Ohio Supercomputer
- Remote Procedure Call (RPC): Run procedure/function on remote process
  - stateless
  - Ex: many many examples
  - Uses marshaling of parameters/return values

More Types of Message Passing

- Distributed objects: RPC for OO
  - remote objects have state!
  - Ex: CORBA, COM/DCOM/COM+, RMI (Java), .NET Remoting, RPyC
- Distributed agents and active objects
  - Like distributed objects, but remote instances can operate independently
- Web Services: RPC over HTTP
  - More interoperable than typical RPC
  - Ex: SOAP, REST

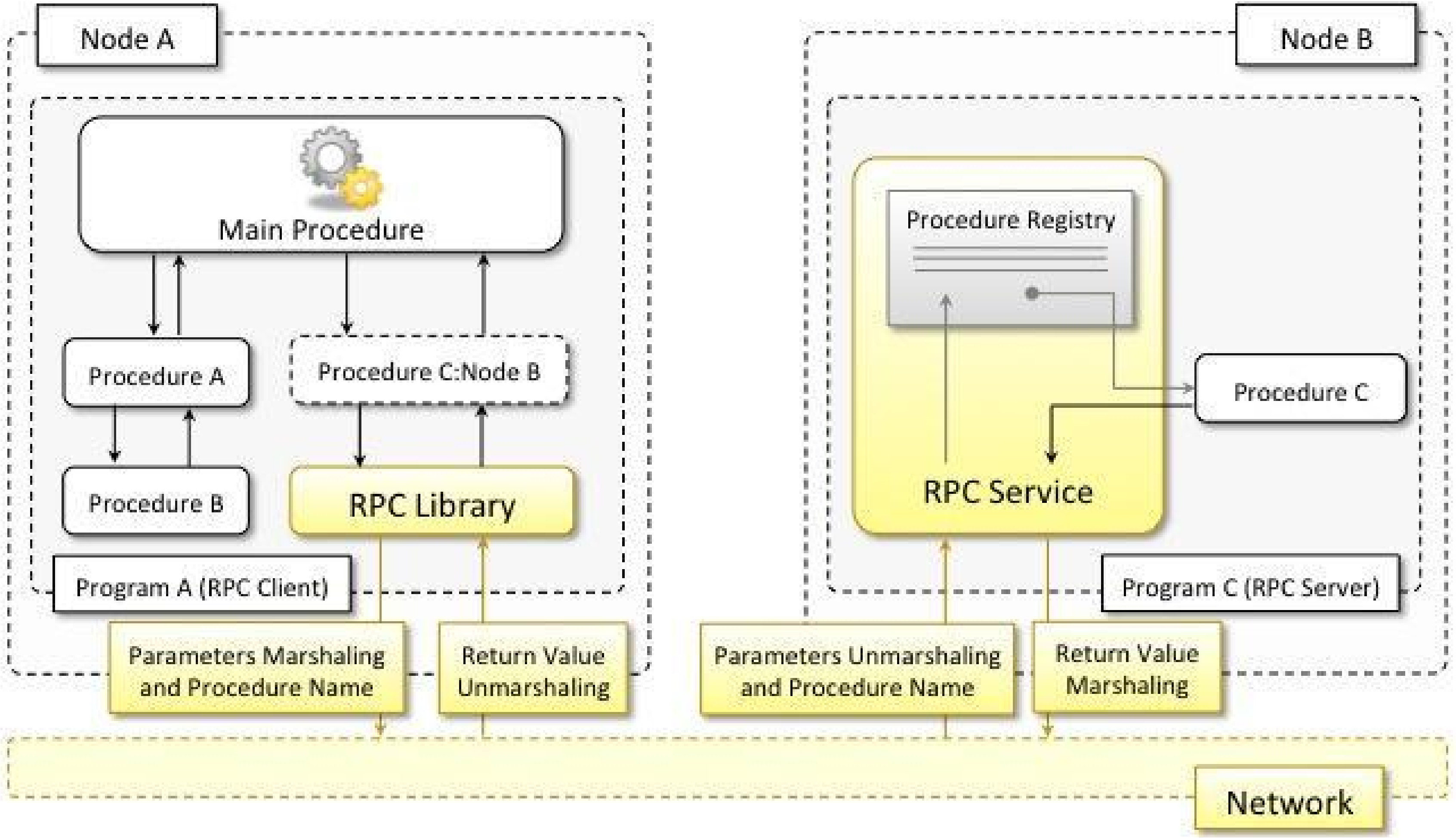
UNIVERSITY OF Cincinnati

Models for Message Communication

- Point-to-point
  - Unique sender and receivers
  - No central infrastructure
  - request-reply model
  - 2 subcategories:
    - direct - request processed when received
    - queued - delay processing
- Publish-and-Subscribe
  - Register for notification of event
    - Push strategy - publisher notifies
    - Pull strategy - subscriber checks in for events

UNIVERSITY OF Cincinnati

Technologies - RPC



RPC - Continued

- Old idea/technology (1980+)
- Caller blocks
- Marshaling/unmarshaling important: convert parameters/return values to be moved on network
- Pass by reference/pointers not suitable

UNIVERSITY OF Cincinnati

RPyC unbounded computing

http://rpyc.readthedocs.org/en/latest/

- RPC system for Python
- Video demo:
  http://showmedo.com/videotutorials/video?name=2780000;fromSeriesID=278
- Classic and Service modes (we'll cover service)
- Any class method prefixed with 'exposed' is available remotely
- First class language/functions possible!

UNIVERSITY OF Cincinnati

Distributed Object Frameworks

- Add state and OO to RPC
- Issues… Object activation & lifetime
  - Who activates an object?
    - Server (on startup)
    - Client (on connection)
  - When does an object die?
    - Server initiated - programmer decides
    - Client - ??? Depends!

Examples of Distributed object frameworks

- Common object request broker architecture (CORBA)
    - cross-platform, cross-language
    - complex, not very popular
- RPyC for Python, platform independent: XML-RPC, JSON-RPC
- Thrift (Facebook) - cross-platform RPC
- Protocol Buffers (Google)

UNIVERSITY OF Cincinnati

Examples of Distributed object frameworks (cont)

- DCOM/COM+
  - Microsoft technology before .NET
  - Losing popularity quickly to .NET
- Java remote method invocation (RMI)
  - *stub-skeleton* concept: define (and publish) interface extending *java.rmi.Remote*
- .NET Remoting
  - Very flexible RPC (can specify transport system, marshaling method, lifetime, etc)
  - Uses TCP or *HTTP*

UNIVERSITY OF Cincinnati

Service-oriented Computing

- ***Boundaries are explicit*** - remote interaction explicit - minimal interface
- ***Services are autonomous*** - general/used anywhere, not for specific thing - user must deal with errors
- ***Services share schemas and contracts, not class or interface definitions*** - not bound to language - common schemas (JSON, XML)
- ***Service compatibility is determined based on policy*** - structural compatibility and semantic compatibility

Abstract away specific implementation technology!

Service-Oriented Architecture (SOA)

Software system designed as a collection of interacting services.

2 Roles:

- Service provider
- Service consumer

UNIVERSITY OF Cincinnati

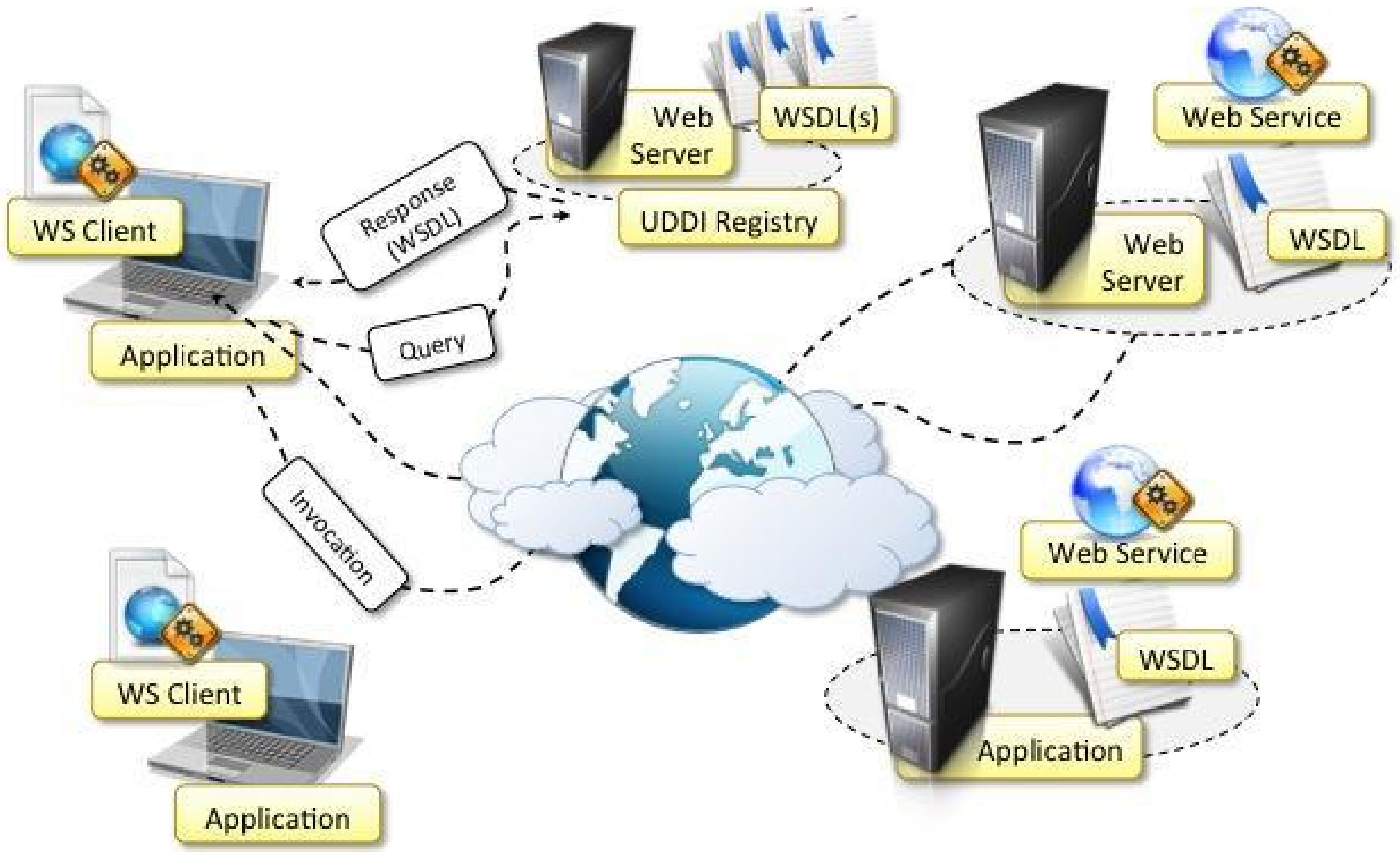
SOA Enterprise Features

- Standardized service contract
- Loose coupling - minimize dependencies
- Abstraction - hide logic
- Reusability
- Autonomy
- Lack of state - easier to use
- Discoverability
- Composability

UNIVERSITY OF Cincinnati

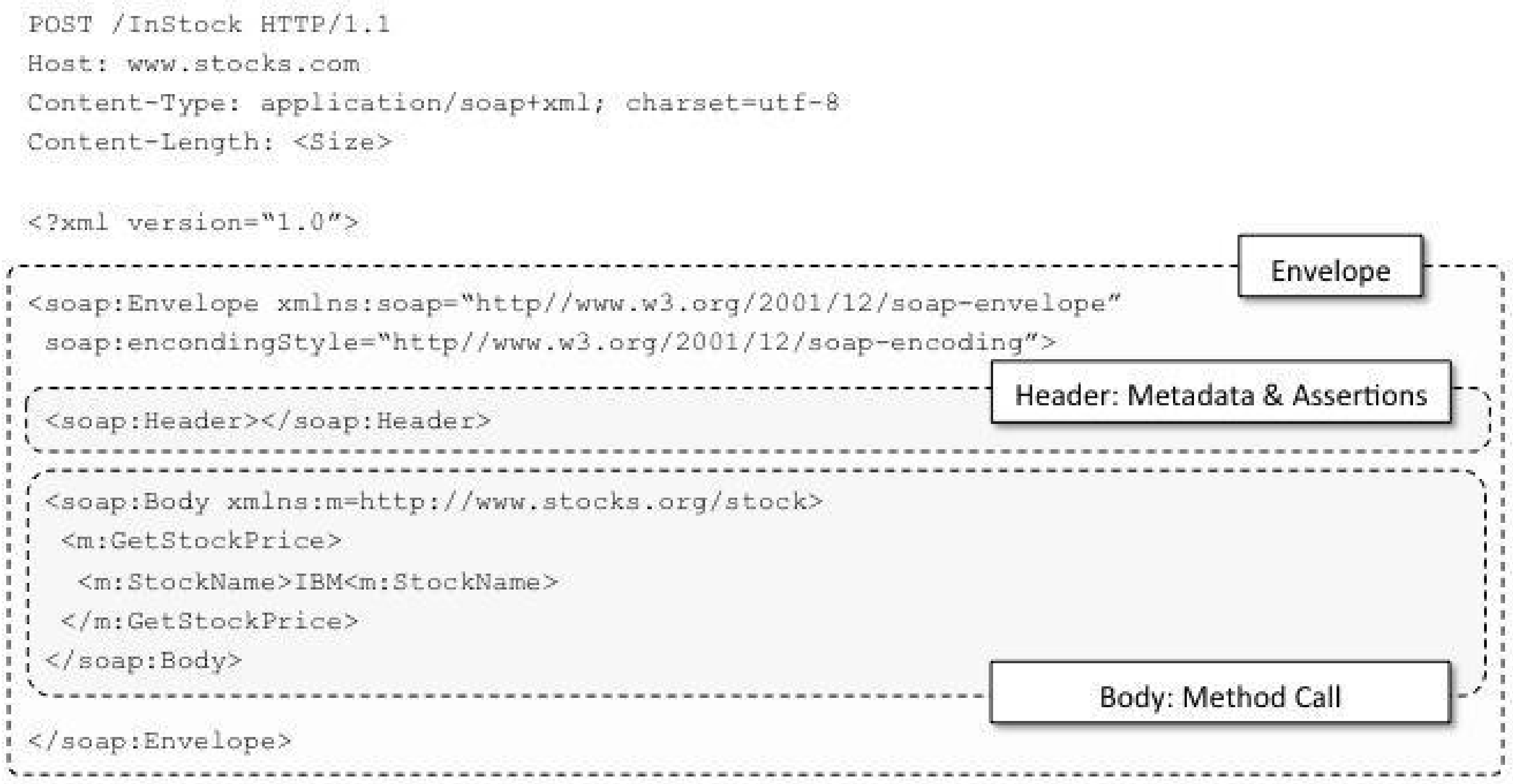
Web Services

SOA using HTTP, SOAP, XML, WSDL



Simple Object Access Protocol (SOAP)

XML language for exchanging information



```
POST /InStock HTTP/1.1
Host: www.stocks.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: <Size>

<?xml version="1.0">
<soap:Envelope xmlns:soap="http//www.w3.org/2001/12/soap-envelope"
 soap:encondingStyle="http//www.w3.org/2001/12/soap-encoding">
<soap:Header></soap:Header>
<soap:Body xmlns:m=http://www.stocks.org/stock>
 <m:GetStockPrice>
  <m:StockName>IBM<m:StockName>
 </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

Representational State Transfer (REST)

- Rely on HTTP `PUT, GET, POST, DELETE` for ‘action’ rather than embedding it in XML
- Can use XML or JSON for encoding
- *Web Service Description Language (WSDL) - document for describing web services*
  - *Designed for SOAP, though falling out of favor - we use API documentation (manual) or libraries*

UNIVERSITY OF Cincinnati

Summary

- Parallel vs distributed computing
- Parallel architectures
- IPC, RPC, Distributed Objects, Active Objects, web services
- Client/Server, Peer-to-peer
- Point-to-point, publish-and-subscribe
- SOA, SOAP, REST, XML, JSON

UNIVERSITY OF Cincinnati