Parallel and Distributed Computing
CS3006 (BCS-6C/6D)
Lecture 25

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

02 May, 2023

Previous Lecture

- Programming with Interfaces
- IDL – Interface Definition Language
- Sun RPC Compiler - `rpcgen`
- gRPC HelloWorld Example
  - The protobuf file (.proto)
  - The server side code that implements the rpc
  - The client side code that enables the generation of the client stub
  - The invoking call within the `main()` of the client
  - Source: https://grpc.io/docs/languages/cpp/quickstart/

Web Service

- A *Web Service* is a network accessible interface to application functionality, built using standard Internet protocols

- A Web Service exposes functionality to a consumer
  - Over the Internet or intranet
  - Functions you can call over the Internet

Need for Web Service?

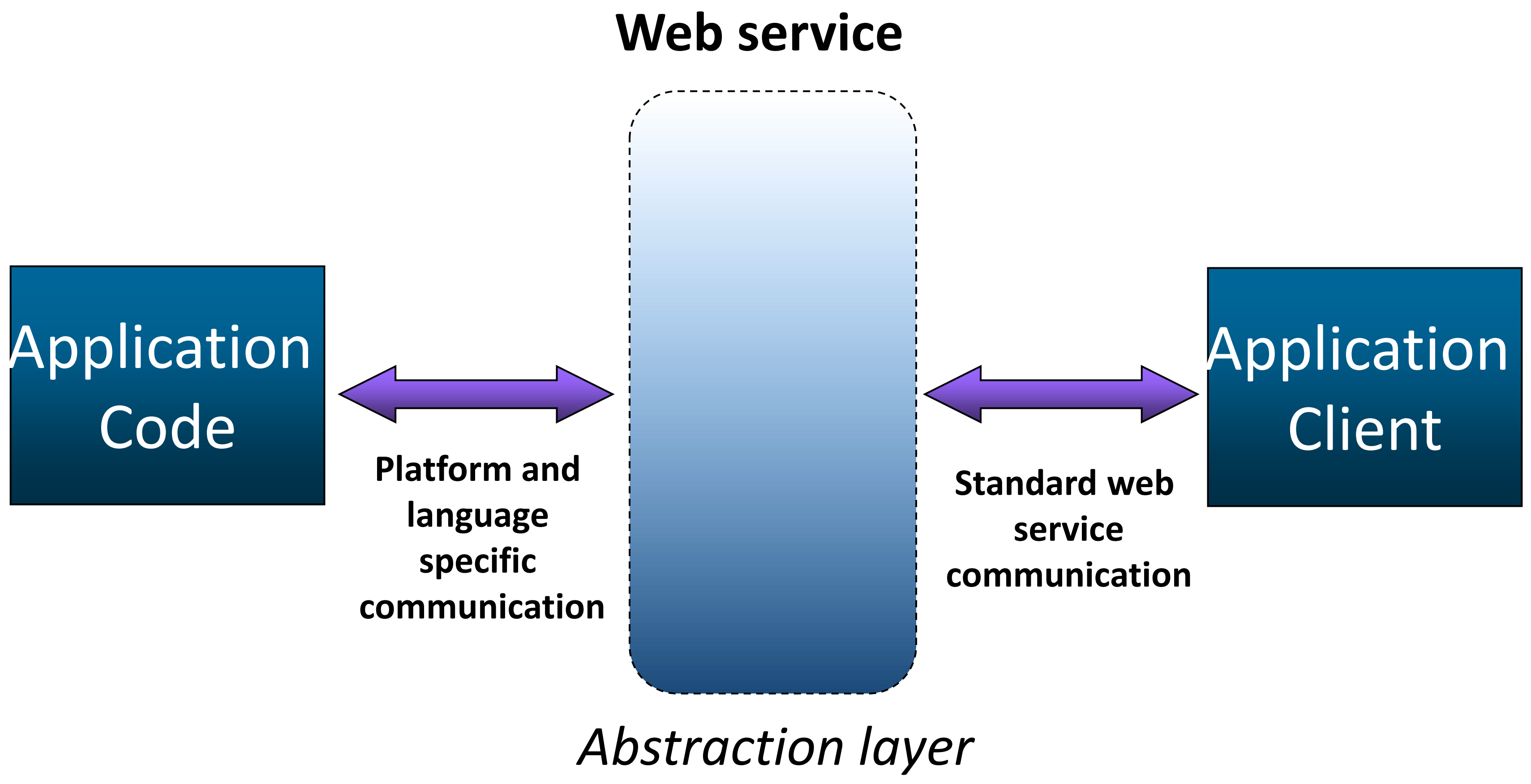
- A variety of programming platforms to develop web-based applications
  - Java
  - .Net
  - Angular JS
  - Node.js
- These heterogeneous applications need to communicate with each other

Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other

- “A Web service is a software system identified by a URI *whose public interfaces and bindings are defined and described using XML*[1]. *Its definition can be discovered by other software systems*[2]. These systems may then interact with the Web service in a manner prescribed by its definition, using *XML based messages*[3] conveyed by Internet protocols[4].”

– W3C

Web Service


Web service
Application Code
Platform and language specific communication
Standard web service communication
Application Client
Abstraction layer

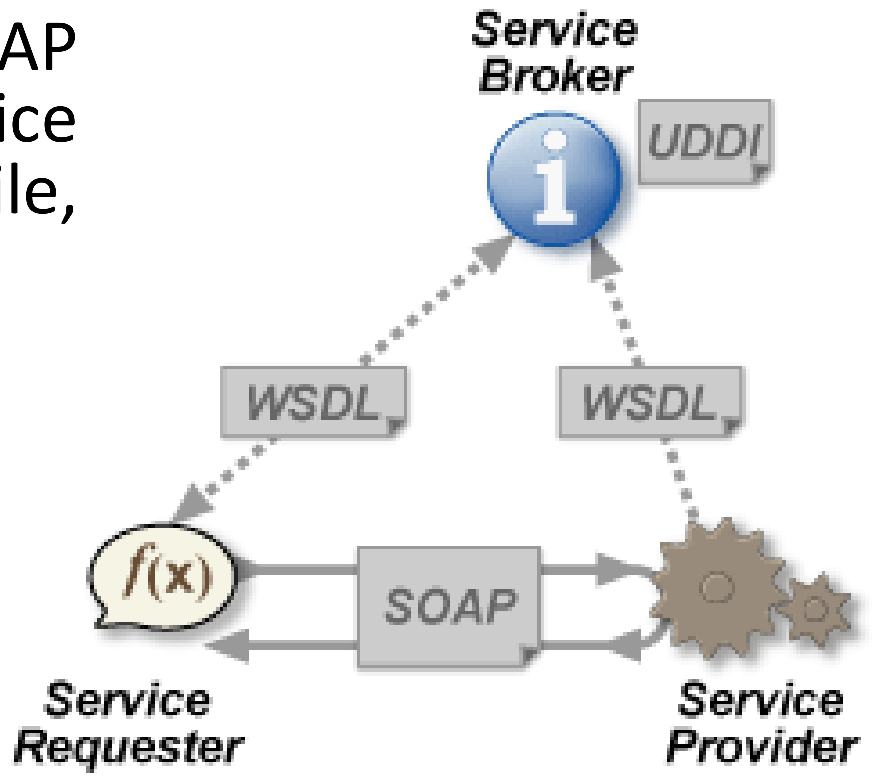
Components of a web service

- A standard way for communication (SOAP)
- A uniform data representation and exchange mechanism (XML)
- A standard meta language to describe the services offered (WSDL)
- A mechanism to register and locate WS based applications (UDDI)

Web services architecture

The service provider sends a WSDL file to UDDI. The service requester contacts UDDI to find out who is the provider for the data it needs, and then it contacts the service provider using the SOAP protocol. The service provider validates the service request and sends structured data in an XML file, using the SOAP protocol.

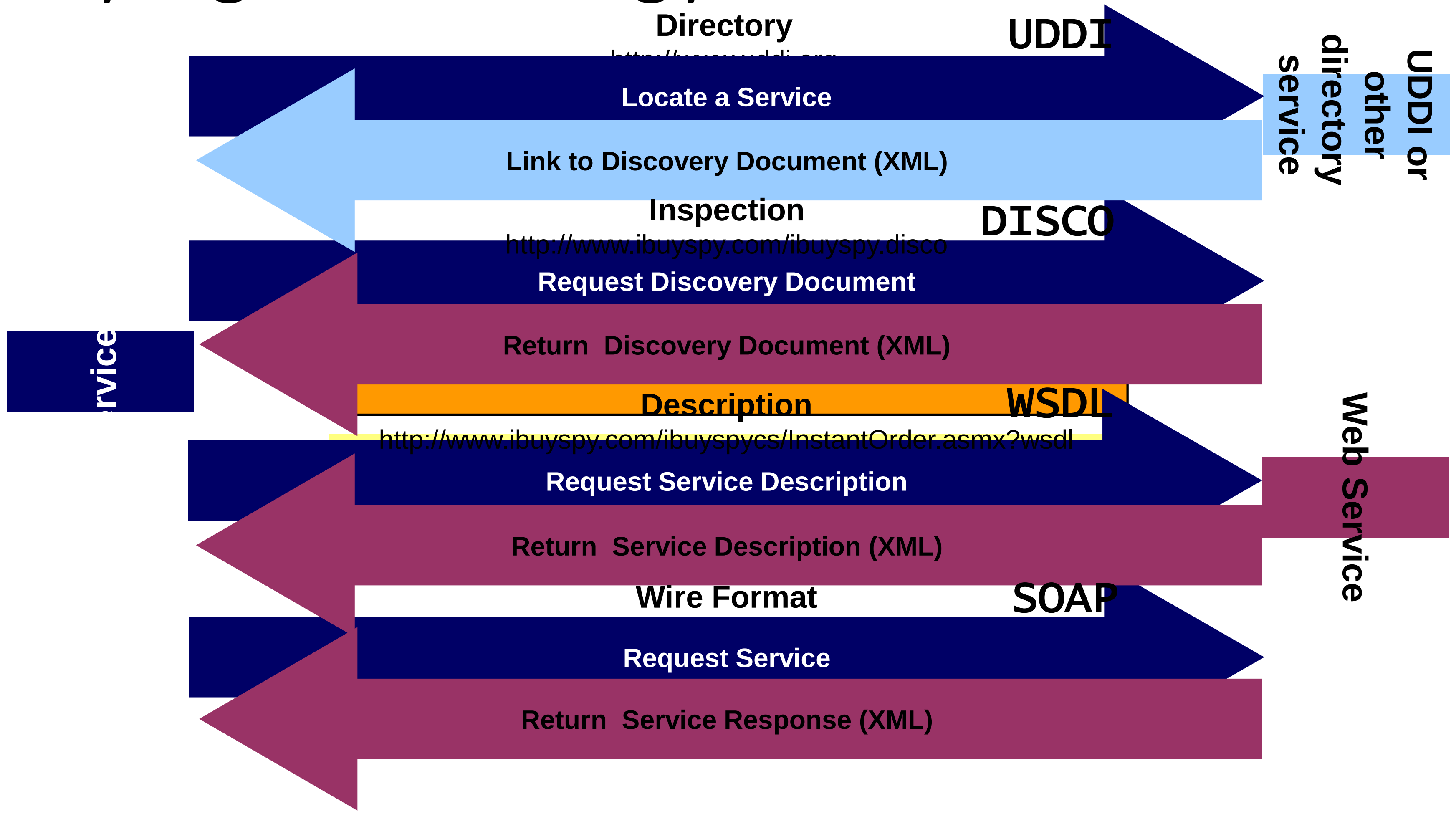
– W3C



Underlying Technology

Directory: Publish & Find Services:	UDDI
Inspection: Find Services on server:	DISCO
Description: Formal Service Descriptions:	WSDL
Wire Format: Service Interactions:	SOAP
Universal Data Format:	XML
Ubiquitous Communications:	Internet

Underlying Technology


Directory
UDDI
http://www.uddi.org
Locate a Service
Link to Discovery Document (XML)
UDDI or other directory service
Inspection
DISCO
http://www.ibuyspy.com/ibuyspy.disco
Request Discovery Document
Return Discovery Document (XML)
rvice
Description
WSDL
http://www.ibuyspy.com/ibuyspycs/InstantOrder.asmx?wsdl
Request Service Description
Web Service
Return Service Description (XML)
Wire Format
SOAP
Request Service
Return Service Response (XML)

Universal Description, Discovery and Integration (UDDI) Protocol

- Enables enterprises to quickly and dynamically discover and invoke Web Services both internally and externally

- Yellow pages of Web services

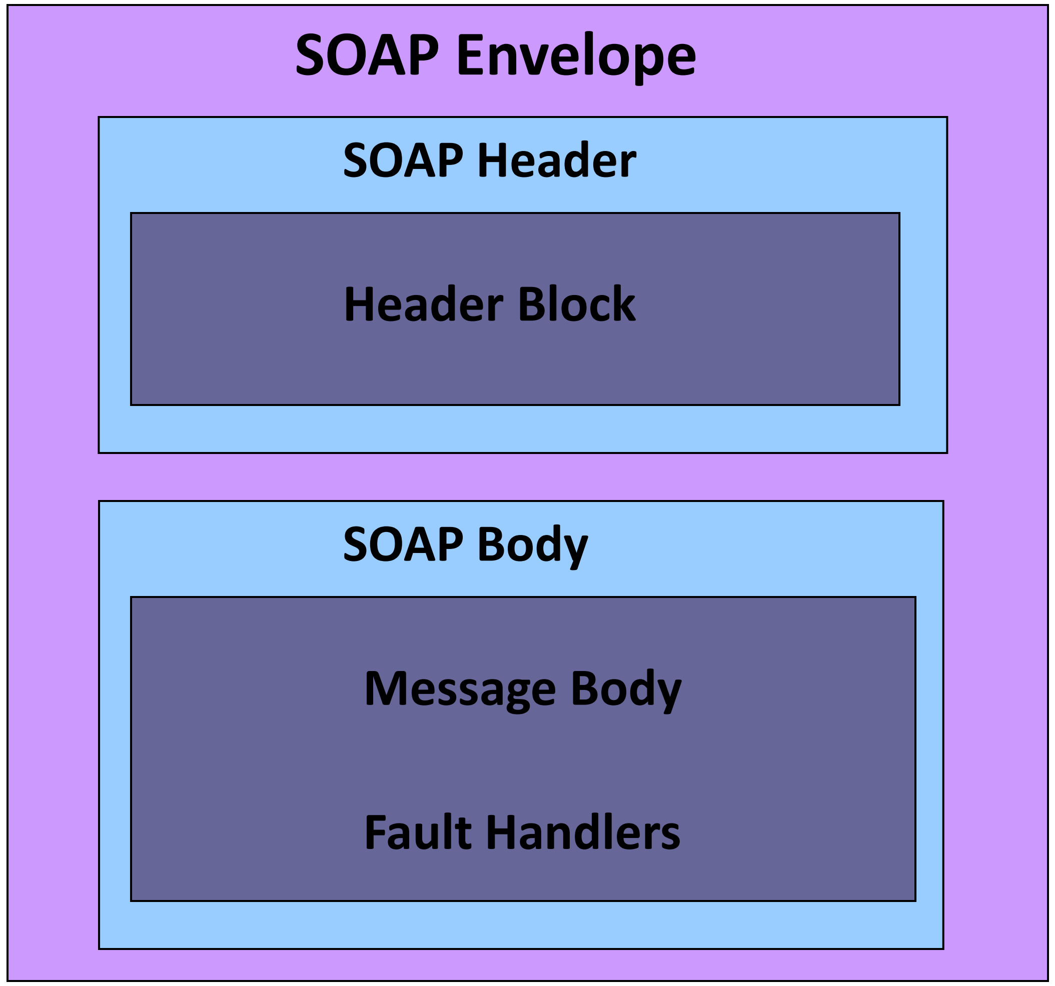
Examples

  - www.uddi.org
  - www.biomoby.org
  - www.xmethods.com

Simple Object Access Protocol (SOAP)

- SOAP is a lightweight (XML-based) protocol for exchange of information in a decentralized, distributed environment. It consists of mainly of:
  - an envelope that defines a framework for describing what is in a message and how to process it
  - relies heavily on XML standards (schemas & name   spaces)

The SOAP message structure



Web Service Definition Language (WSDL)

- WSDL is an XML format for describing network services operating on messages containing either document-oriented or procedure-oriented information.

- It defines Web Service as collection of network endpoints or ports.

XML

- Stands for “Extensible Markup Language”
- Language specification for describing data
  - Syntax rules
  - Syntax & Grammar for creating Document Type Definitions

- Widely used and open standard
  - Defined by the World Wide Web Consortium (W3C)
  - http://www.w3.org/TR/2000/REC-xml-20001006

Advantages of Web Services

- Allow programs written in different languages on different platforms to distribute an application in a standardized manner.
- Adapt the loosely coupled Web programming model for use in applications that are not browser based.
- The goal is to provide a platform for building distributed applications using software
  - running on different operating systems and devices,
  - written using different programming languages and tools from multiple vendors,
  - all potentially developed and deployed independently.

How to create a web service

- Following link provides a useful example to create and consume a web service in Visual Studio!
    - https://www.c-sharpcorner.com/UploadFile/4d9083/create-simple-web-service-in-visual-studio-2008-2010-2012/

Time: A major issue in distributed systems

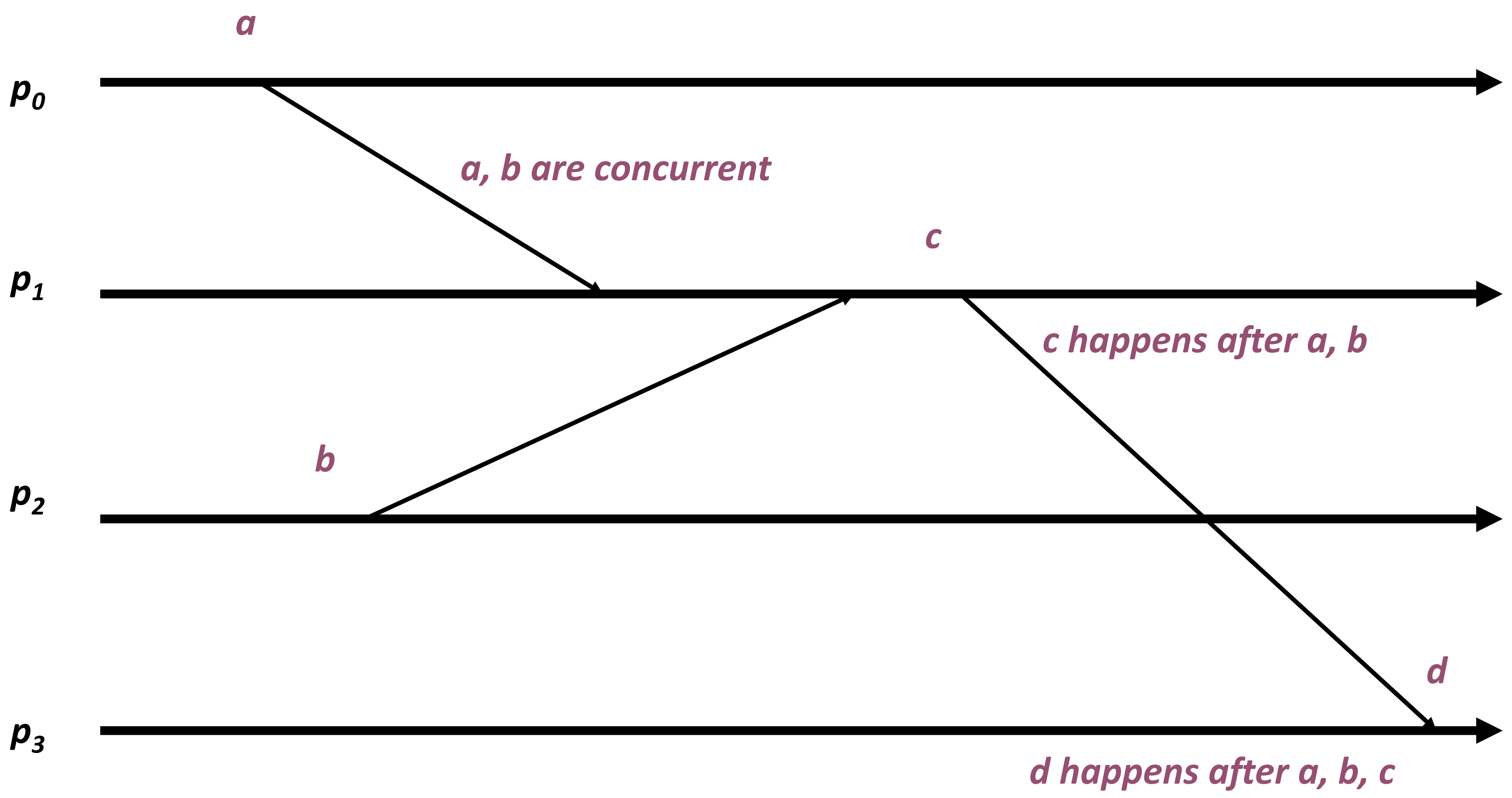
- Measuring time can be problematic due to the existence of multiple frames of reference
  - Time seen by external observer. A global clock of perfect accuracy
  - Time seen on clocks of individual processes. Each has its own clock, and clocks may drift out of sync.
  - Logical notion of time: event *a* occurs before event *b* and this is detectable because information about *a* may have reached *b*.
- Clock synchronization is the big problem: clocks can drift apart and resynchronization, in software, is inaccurate
- Unpredictable speeds a feature of all computing systems, hence can’t predict how long events will take (e.g. how long it will take to send a message and be sure it was delivered to the destination)

Logical notion of time

- Has no clock in the sense of “real-time”

- Focus is on definition of the “*happens before*” relationship: *“a happens before b” if:*
  - both occur at same place and *a* finished before *b* started, or
  - *a* is the send of message *m, b* is the delivery of *m*, or
  - *a* and *b* are linked by a chain of such events

Logical time as a time-space picture



Notation

- Use “arrow” to represent happens-before relation

- For previous slide:
  - $a \rightarrow c$, $b \rightarrow c$, $c \rightarrow d$
  - *hence,* $a \rightarrow d$, $b \rightarrow d$
  - *a, b are concurrent*

- Also called the “potential causality” relation

Logical clocks

- Proposed by Lamport to represent causal order
- Write: LT(e) to denote logical timestamp of an event e, LT(m) for a timestamp on a message, LT(p) for the timestamp associated with process p
- Algorithm ensures that if $a \rightarrow b$, then

$$\mathrm{LT}(a) < \mathrm{LT}(b)$$

Algorithm

- Each process maintains a counter, LT(p)

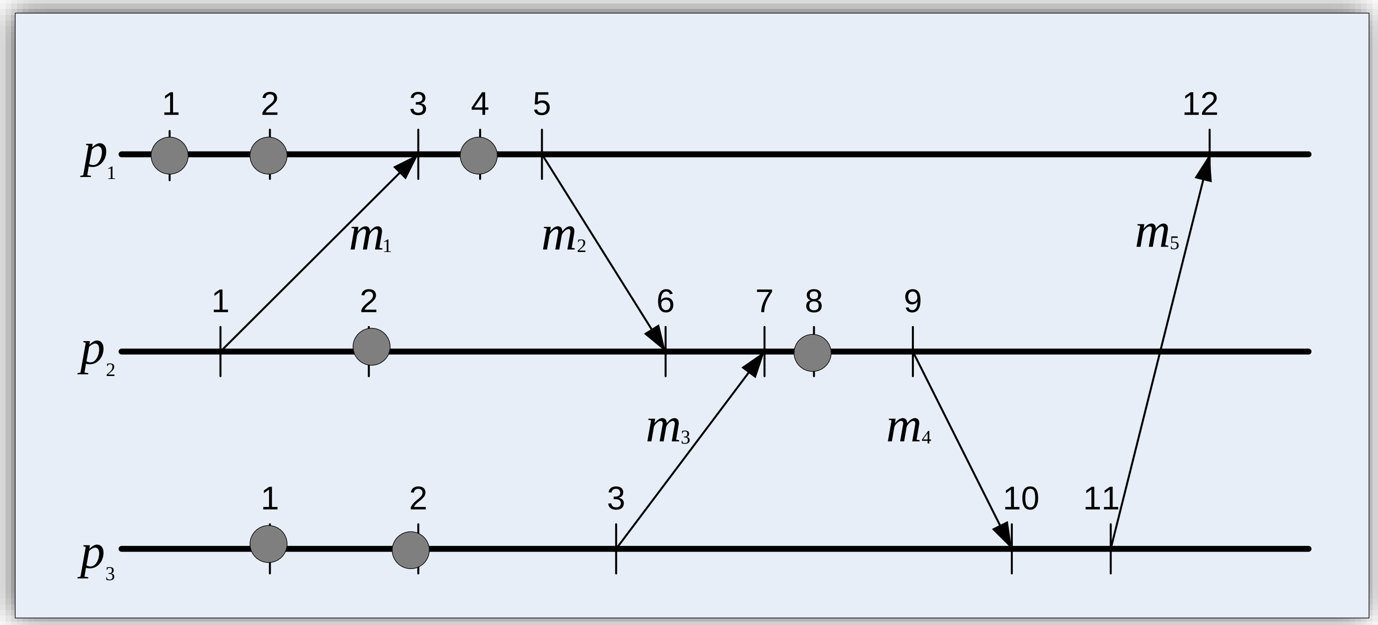
- For each event other than message delivery:

  set LT(p) = LT(p)+1

- When sending message m,
  set LT(m) = LT(p)

- When delivering message m to process q,
  set LT(q) = max(LT(m), LT(q))+1

Logical Clocks example

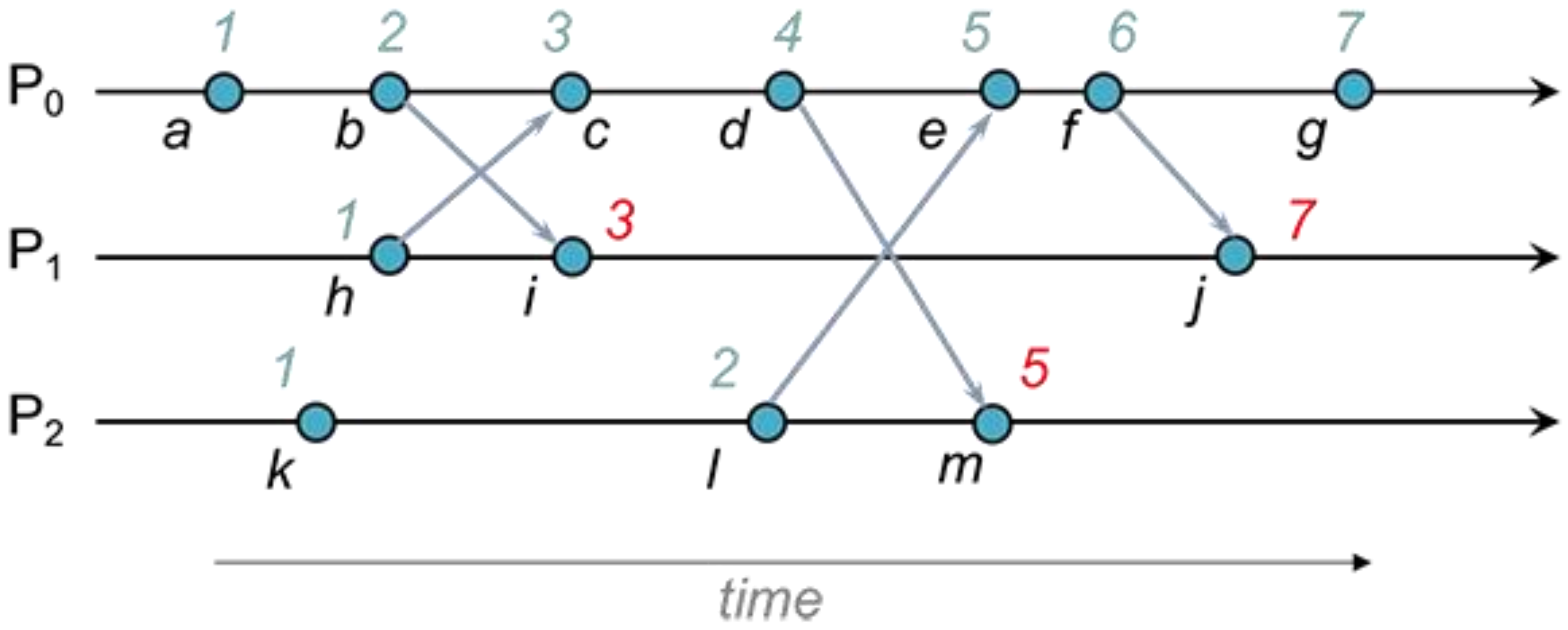


Three processes and their logical clocks. The usual labeling of events as $e_1^1, e_1^2, e_1^3, \ldots$ is omitted to avoid overloading the figure; only the logical clock values for the local and communication events are marked. The correspondence between the events and the logical clock values is obvious: $e_1^1, e_2^1, e_3^1 \rightarrow 1$, $e_1^5 \rightarrow 5$, $e_2^4 \rightarrow 7$, $e_3^4 \rightarrow 10$, $e_1^6 \rightarrow 12$, and so on. Global ordering of all events is not possible; there is no way to establish the ordering of events e_1^1, e_2^1 and e_3^1.

$LC(e) = LC + 1 \rightarrow$ *if e is a local event or a send(m) event*

$LC(e) = max(LC + 1, TS(m) + 1) \rightarrow$ *if e = receive(m)*

Another Example: Logical Clocks



References

1. Slides of Dr. Haroon Mahmood

Helpful Links:

1. http://lamport.azurewebsites.net/pubs/time-clocks.pdf
2. https://en.wikipedia.org/wiki/Lamport_timestamp