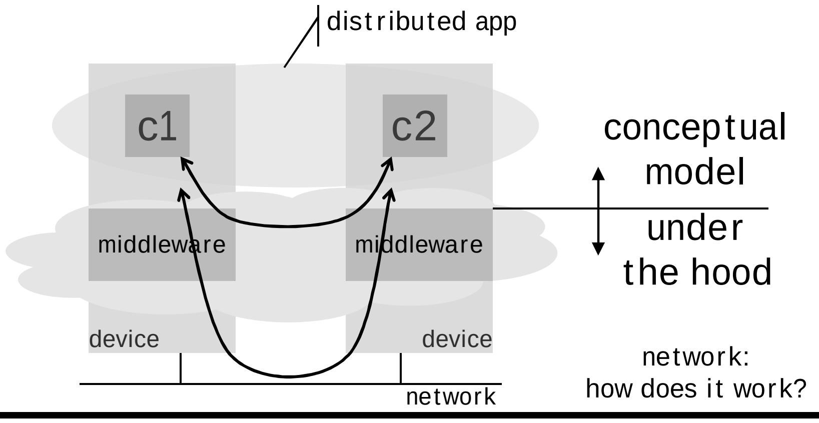
Distributed Software Systems

Lecture 3
Communication: Implementation

João Pedro Sousa
SWE 622
George Mason University

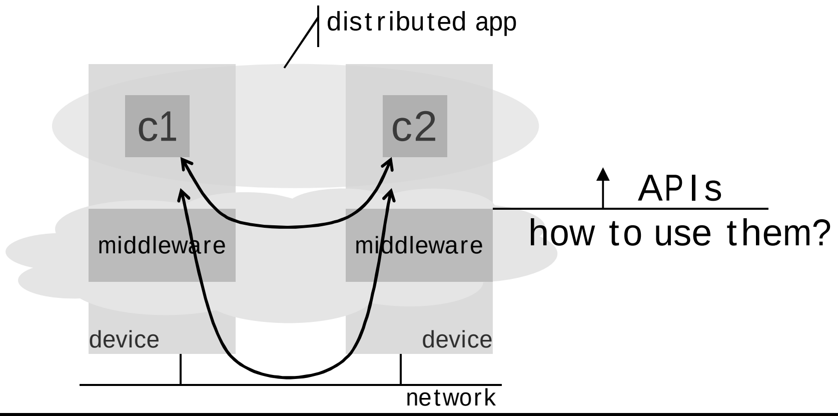
middleware offers conceptual model for communication

previous lecture: communication styles,
their properties and limitations



network:
how does it work?

this lecture:



outline

- toy example
  - using RMI
  - using messages
- case study
  - messages over TCP

lift the design
CalculatePi java tutorial

in-class exercise
using Eclipse

lift the design
simplified CalculatePi

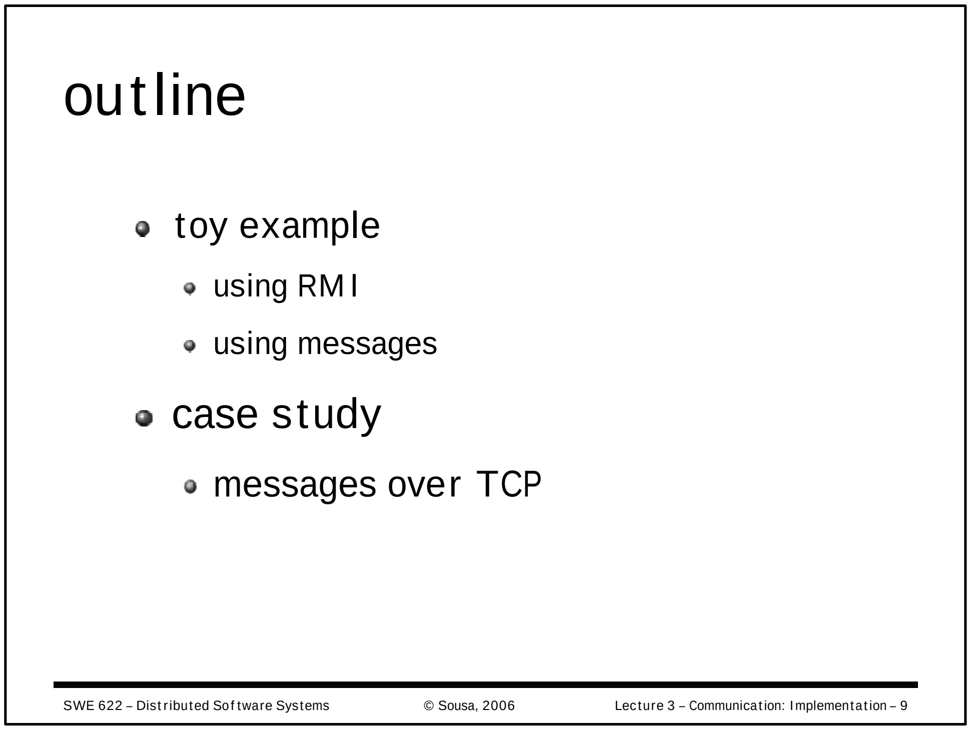
in-class exercise
using Eclipse

outline

- toy example
  - using RMI
  - using messages
- case study
  - messages over TCP

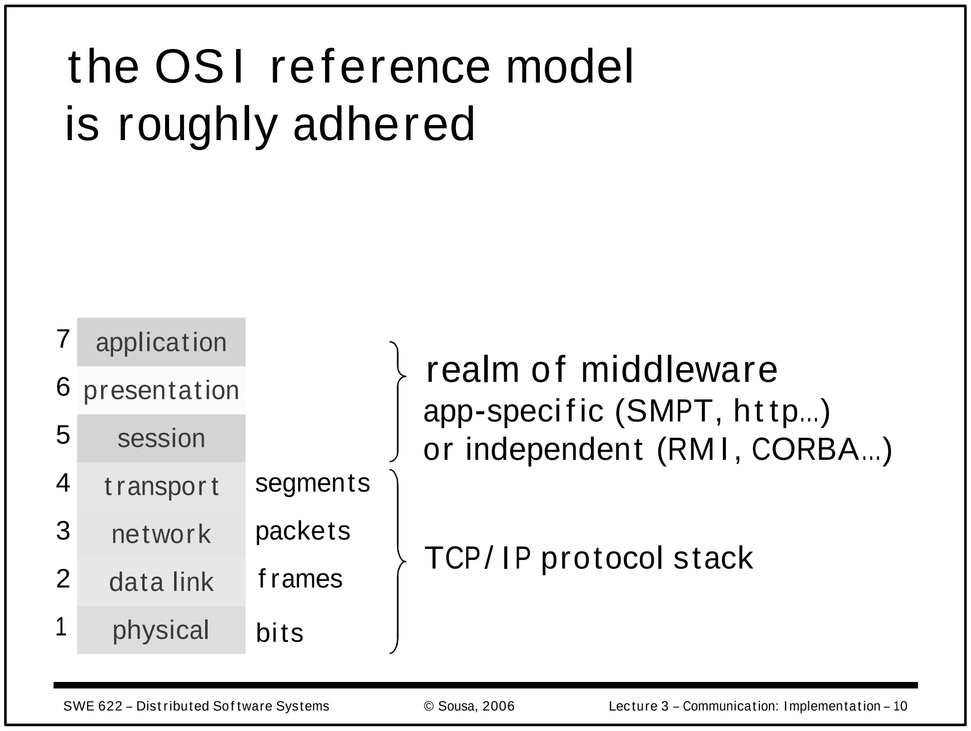
lift the design
simplified CalculatePi using messages

in-class exercise
using Eclipse



outline

- toy example
  - using RMI
  - using messages
- case study
  - messages over TCP



the OSI reference model is roughly adhered

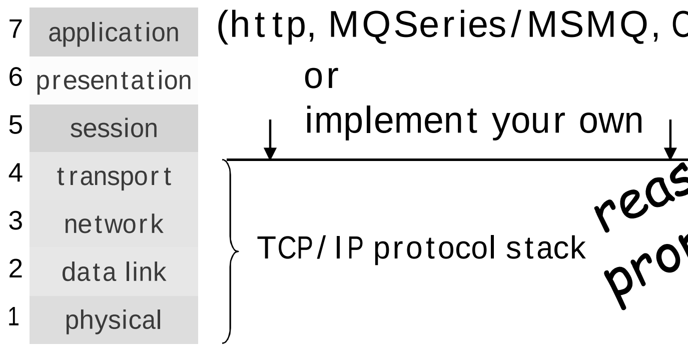
7	application		realm of middleware app-specific (SMPT, http…) or independent (RMI, CORBA…)
6	presentation		
5	session		
4	transport	segments	TCP/IP protocol stack
3	network	packets	
2	data link	frames	
1	physical	bits	

want the messaging style in your app? there are alternatives:

fit your app into existing middleware... (http, MQSeries/MSMQ, CORBA-AMI)

or

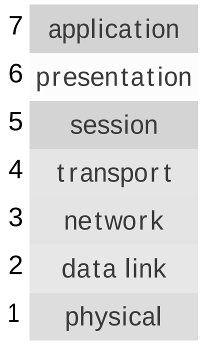
implement your own



reasonable proposition?

cost-benefit analysis of using a generic messaging middleware

fit your app into existing middleware... (http, MQSeries/MSMQ, CORBA-AMI)

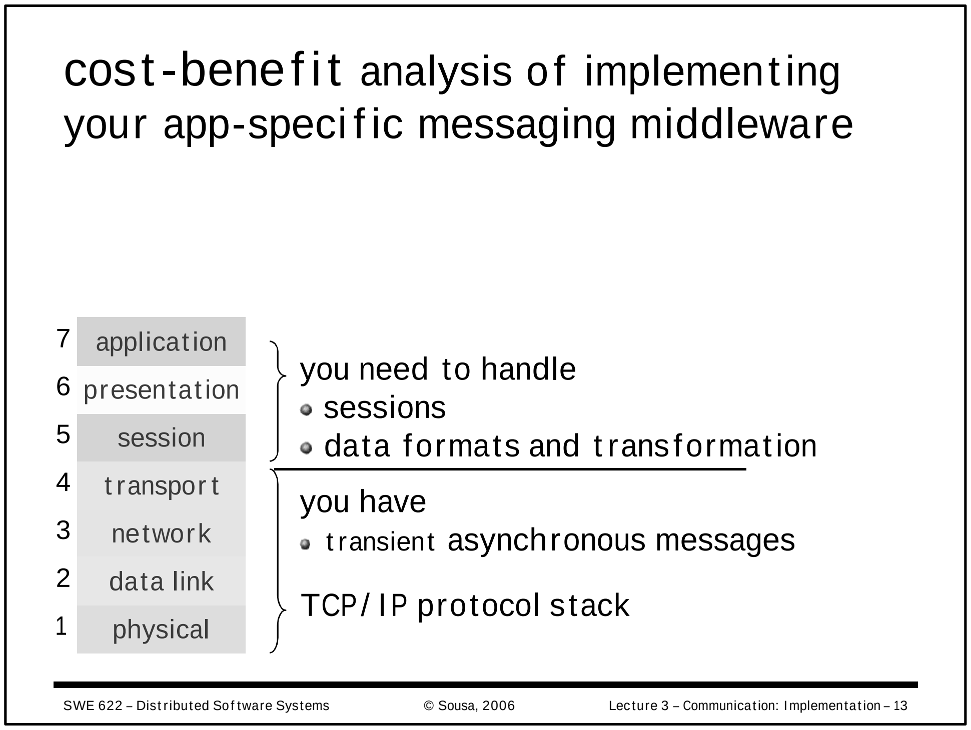


reason to do it

- gives you a lot of functionality

reasons not to do it

- maybe a lot more than what you need
- maybe not exactly what you need
- cost



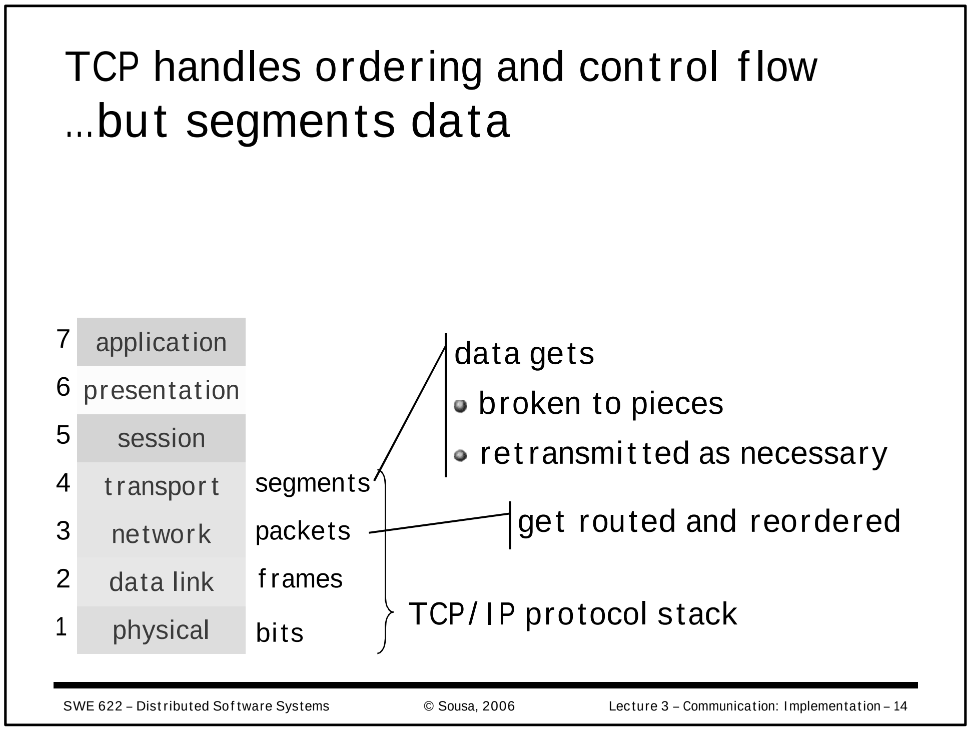
cost-benefit analysis of implementing your app-specific messaging middleware

7	application
6	presentation
5	session
4	transport
3	network
2	data link
1	physical

you need to handle
- sessions
- data formats and transformation

you have
- transient asynchronous messages

TCP/IP protocol stack



TCP handles ordering and control flow …but segments data

7	application	
6	presentation	
5	session	
4	transport	segments
3	network	packets
2	data link	frames
1	physical	bits

data gets
- broken to pieces
- retransmitted as necessary

get routed and reordered

TCP/IP protocol stack

lift the design
PtoP library

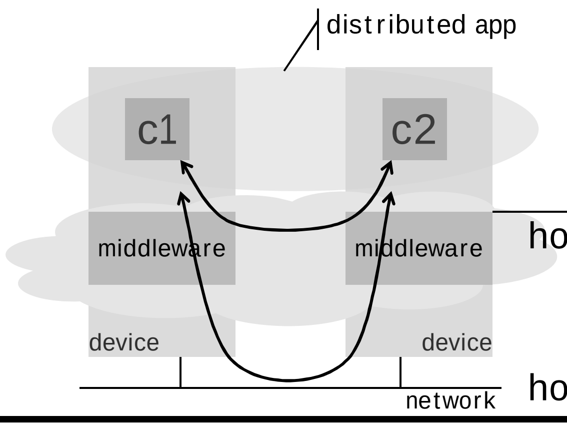
in-class exercise
using Eclipse

choosing a binding solution

if component deployment:

- is determined at startup/design time
  - read endpoints off a config file
- changes at run-time
  - use a registry/service discovery (topic of Lecture 4)

in summary: we saw how to use RMI and messaging APIs

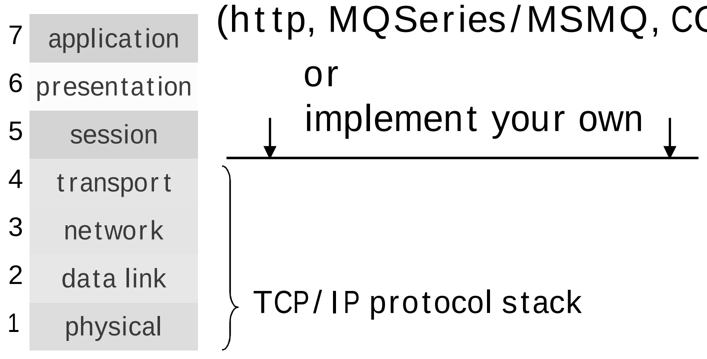


APIs
how to use them?

network:
how does it work?

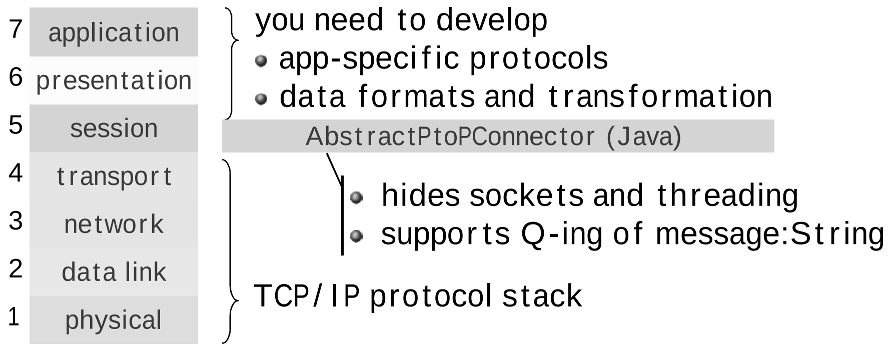
what if you want to use the messaging style in your app?

adopt the conceptual model
of existing middleware…
(http, MQSeries/MSMQ, CORBA-AMI)
or
implement your own



TCP/IP protocol stack

not that hard to implement your own: you may use AbstractPtoPConnector as a library for point-to-point messaging



you need to develop
- app-specific protocols
- data formats and transformation

AbstractPtoPConnector (Java)
- hides sockets and threading
- supports Q-ing of message:String

TCP/IP protocol stack