(Q) Describe: ...

Challenges of Distributed Systems
SCC311

(Q) Describe: Overview of the Session

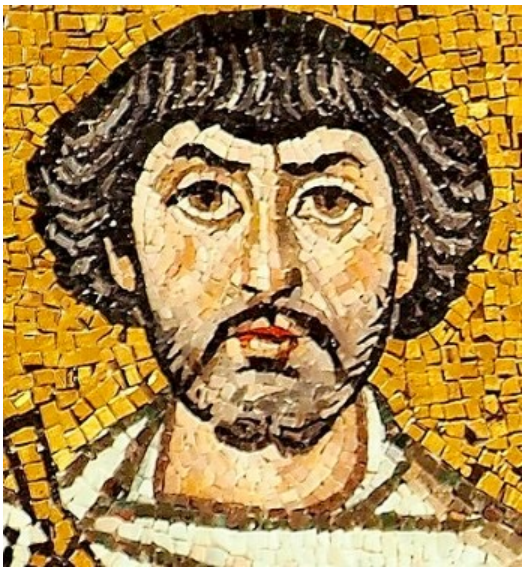
1 Overview of the Session

Two Generals problem
Byzantine Generals problem
Fallacies of distributed computing
Middleware

(Q) Describe: Two Generals Problem

2 Two Generals Problem

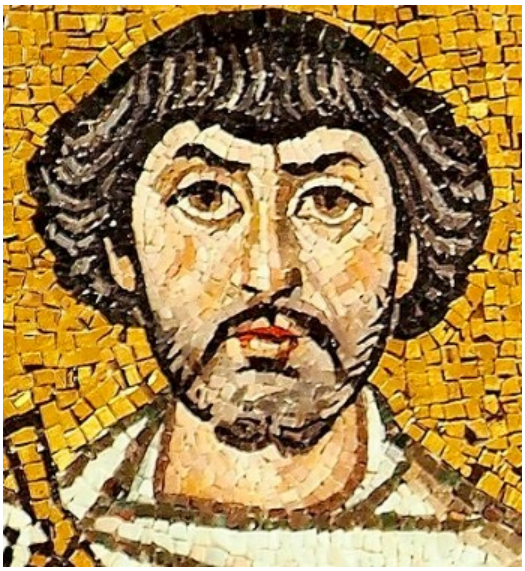
Complication in distributed consensus
Two generals, each leading an army, want to capture a city. The attack is only successful if both armies attack together
Coordination is vital
Generals need to reach consensus in order
to succeed in conquering the city.
Belisarius
Narses
City
General1 General2
messengers
Attack or notAttack or not



(Q) Describe: Two Generals Problem

3 Two Generals Problem

Complication in distributed consensus
Two generals, each leading an army, want to capture a city. The attack is only successful if both armies attack together
Belisarius
Narses
City
General1 General2
messengers
Attack or notAttack or not
Army1 Army2 Result
No Attack No attack Nothing happens
No Attack Attack Army2 defeated
Attack No attack Army1 defeated
Attack Attack City captured



(Q) Describe: Two Generals Problem

4 Two Generals Problem

Important challenges
Faulty network
Receipt of Acknowledgement is necessary
General1 General2
Attack on Oct 14 ?
Acknowledged!
General1 General2
Attack on Oct 14 ?
time time time time

(Q) Describe: How the generals should

5 How the generals should

decide?
Is it possible to come up with an algorithm that works all the time ?
Answer is no.
Possible approaches (none of them fully works):
Approach 1
General1 sends a lot of messages at once, and then attacks
Problem: if all the messengers are captured, army1 is defeated
Approach 2
General1 attacks only if an ACK is received from General 2
Now general2 will be captured if its response is intercepted or delayed
Problem: General2 has no idea if his ACK is received by General1
In the two generals problem, we assume a faulty network but processes (i.e., generals) are not faulty.

(Q) Describe: FLP Impossibility Result

6 FLP Impossibility Result

FLP impossibility result
Fischer, Lynch, and Patterson, Impossibility of Distributed Consensus with One Faulty
Process, 1985
Asynchronous systems:
No bounds on message delivery time and process execution time
Clock drifts are also unbounded
When there is no upper bound on the time a process takes to complete its work and
respond, it's impossible to make the distinction between a process that is crashed and one
that is working (but taking very long to respond).
FLP shows that there is no guarantee for distributed processes to reach consensus in an
asynchronous environment, where it's possible for at least one process to crash.
Equivalently, it's not always possible to detect failure in an asynchronous system with
nodes crashing.
So let's consider the case where messages are delivered reliably (i.e., network is
not faulty) next.

(Q) Describe: Byzantine Generals Problem

7 Byzantine Generals Problem

We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division has its own general. One commanding general (on behalf of the emperor) sends an order to the lieutenant generals. However, some of the generals, including the commanding general, may be traitors. The generals must have an algorithm to guarantee that:

- All loyal lieutenant generals decide upon the same plan of

action.

- A small number of traitors cannot cause the loyal generals to

adopt a bad plan."
L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem", 1982

(Q) Describe: Byzantine Generals Problem

8 Byzantine Generals Problem

The Byzantine Generals Problem
LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International
Re lia ble compute r sys tems mus t ha ndle ma lfunctioning compone nts tha t give conflicting informa tion
to diffe rent pa rts of the sys tem. This s itua tion ca n be expressed a bs tra ctly in te rms of a group of
genera ls of the Byza ntine a rmy ca mpe d with the ir troops a round a n e ne my city. Communica ting only
by messenger, the genera ls mus t agree upon a common ba ttle pla n. How- ever, one or more of the m
ma y be tra itors who will try to confuse the others . The proble m is to find a n a lgorithm to e ns ure tha t
the loya l genera ls will re a ch agreement. It is shown tha t, us ing only ora l messages , this proble m is
solvable if a nd only if more tha n two-thirds of the genera ls a re loya l; so a s ingle tra itor ca n confound
two loya l genera ls . With unforgeable writte n messages , the proble m is solvable for a ny numbe r of
genera ls a nd poss ible tra itors . Applica tions of the s olutions to re lia ble compute r s ys te ms a re the n
discussed.
Ca tegories a nd S ubje ct Descriptors : C.2.4. [C o m p u te r-C o m m u n ic a tio n Ne tworks]: Dis tribute d
S ys tems–ne twork operating systems; D.4.4 [O p e ra tin g S ys te ms]: Communica tions Ma na ge me nt–
network communication; D.4.5 [O p e ra tin g S ys te ms]: Reliability–fault tolerance
Ge ne ra l Te rms : Algorithms, Re lia bility
Additiona l Ke y Words a nd Phrases : Inte ra ctive cons is tency
/
1 . INTR O DUC TIO N
A re lia b le c o m p u t e r s y s t e m m u s t b e a b le to c o p e with th e fa ilu re o f o n e o r m o re
o f its c o m p o n e n t s . A fa ile d c o m p o n e n t m a y e xh ib it a typ e o f b e h a v io r t h a t is

o ft e n o v e r lo o k e d - - n a m e ly , s e n d in g c o n flic t in g in fo r m a t io n to d iffe re n t p a r t s o f
t h e s ys te m . T h e p r o b le m o f c o p in g with th is typ e o f fa ilu re is e xp re s s e d a b s t r a c t ly
a s th e B y z a n t in e G e n e r a ls P ro b le m . W e d e vo te th e m a jo r p a r t o f th e p a p e r to a
d is c u s s io n o f th is a b s t r a c t p r o b le m a n d c o n c lu d e b y in d ic a t in g h o w o u r s o lu t io n s
c a n b e u s e d in im p le m e n t in g a re lia b le c o m p u t e r s ys te m . W e im a g in e t h a t s e ve ra l d iv is io n s o f th e B y z a n t in e a r m y a re c a m p e d o u ts id e
a n e n e m y c ity, e a c h d iv is io n c o m m a n d e d b y its o w n g e n e ra l. T h e g e n e ra ls c a n
c o m m u n ic a t e with o n e a n o t h e r o n ly b y m e s s e n g e r . Afte r o b s e rv in g th e e n e m y ,
t h e y m u s t d e c id e u p o n a c o m m o n p la n o f a c t io n . Ho we ve r , s o m e o f th e g e n e ra ls

This research was s upporte d in pa rt by the Na tiona l Ae rona utics a nd Space Adminis tra tion unde r
contra ct NAS1-15428 Mod. 3, the Ballis tic Miss ile Defense S ys te ms Comma nd unde r contra ct
DASG60-78-C-0046, a nd the Army Research Office unde r contra ct DAAG29-79-C-0102.

Authors ' address : Compute r Science Labora tory, S RI Inte rna tiona l, 333 Ravenswood Avenue , Me nlo
P a rk, CA 94025.

ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, Pages 382-401.

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to find an algorithm to ensure that the loyal generals will reach agreement. It is shown that, using only oral messages, this problem is solvable if and only if more than two-thirds of the generals are loyal; so a single traitor can confound two loyal generals. With unforgeable written messages, the problem is solvable for any number of generals and possible traitors. Applications of the solutions to reliable computer systems are then discussed.

Categories and Subject Descriptors: C.2.4. [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.4 [**Operating Systems**]: Communications Management—*network communication*; D.4.5 [**Operating Systems**]: Reliability—*fault tolerance*

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Interactive consistency

1. INTRODUCTION

A reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked—namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. We devote the major part of the paper to a discussion of this abstract problem and conclude by indicating how our solutions can be used in implementing a reliable computer system.

We imagine that several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. The generals can communicate with one another only by messenger. After observing the enemy, they must decide upon a common plan of action. However, some of the generals

This research was supported in part by the National Aeronautics and Space Administration under contract NAS1-15428 Mod. 3, the Ballistic Missile Defense Systems Command under contract DASG60-78-C-0046, and the Army Research Office under contract DAAG29-79-C-0102.
Authors' address: Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025.

ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, Pages 382–401.

(Q) Describe: Byzantine Generals Problem

9 Byzantine Generals Problem

3 processes, commanding general is disloyal
Lieutenant1 Lieutenant2
Commander
??

(Q) Describe: Byzantine Generals Problem

10 Byzantine Generals Problem

3 processes, commanding general is disloyal
Lieutenant1 Lieutenant2
Commander
Retreat!Attack!

(Q) Describe: Byzantine Generals Problem

11 Byzantine Generals Problem

What should the loyal lieutenants do to follow a unified course of action?
Lieutenant1 Lieutenant2
Commander
Retreat!Attack!

(Q) Describe: Byzantine Generals Problem

12 Byzantine Generals Problem

What should lieutenants do ?
Lieutenants can communicate with one another and compare notes
Lieutenant1 Lieutenant2
Commander
Retreat!Attack!

(Q) Describe: Byzantine Generals Problem

13 Byzantine Generals Problem

Comparing notes (by sending messages to each other)
Remember this time network is not faulty
Lieutenant1 Lieutenant2
I was told
Attack
I was told
Retreat
Commander
Retreat!Attack!

(Q) Describe: Byzantine Generals Problem

14 Byzantine Generals Problem

Lieutenants can't agree on the commander's order so they do the default action which is retreat.
Commander
Lieutenant1 Lieutenant2
Retreat!Attack!
I was told
Attack
I was told
Retreat

(Q) Describe: Byzantine Generals Problem

15 Byzantine Generals Problem

Both scenarios are equivalent from Lieutenant1's point of view
Commander
Lieutenant1 Lieutenant2
Retreat!Attack!
I was
told
Attack
I was
told
Retreat
Commander
Lieutenant1 Lieutenant2
Attack!Attack!
I was
told
Attack
I was
told
Retreat

(Q) Describe: Byzantine Generals Problem

16 Byzantine Generals Problem

Both scenarios are equivalent from Lieutenant2's point of view
Commander
Lieutenant1 Lieutenant2
Retreat!Attack!
I was
told
Attack
I was
told
Retreat
Commander
Lieutenant1 Lieutenant2
Attack!Attack!
I was
told
Retreat
I was
told
Attack

(Q) Describe: Byzantine Generals Problem

17 Byzantine Generals Problem

With d = 1 traitor, the Byzantine Generals problem is not solvable with 3 generals
Consensus is not possible
You need at least 4 nodes! (see next slide)
Commander
Lieutenant1 Lieutenant2
Attack!Attack!
I was told
Attack
I was told
Retreat

(Q) Describe: Byzantine Generals Problem

18 Byzantine Generals Problem

With more than 3 participants, the strategy is still the same
All lieutenants compare notes with each other
Then follow the majority "vote"
If the majority was told "Attack", then attack
If the majority was told "Retreat!", then retreat
Commander
Lieutenant1 Lieutenant2
Attack!Attack!
I was told
Attack
I was told
Retreat

(Q) Describe: Byzantine Generals Problem

19 Byzantine Generals Problem

Communicating notes only work with one traitor
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7

(Q) Describe: Byzantine Generals Problem

20 Byzantine Generals Problem

Let's say all lieutenants compare notes as before
Each lieutenant sends a message to every other lieutenant and
indicates the order it received
Commander
Lieutenant1
Retreat!Attack!
I was told
Attack
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
I was told
Attack
I was told
Attack
I was told
Retreat
I was told
Retreat
I was told
Retreat

(Q) Describe: Byzantine Generals Problem

21 Byzantine Generals Problem

Let's say all lieutenants compare notes as before
Each lieutenant sends a message to every other lieutenant and
indicates the order it received
Commander
Lieutenant1
Retreat!Attack!
I was told
Attack
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7

(Q) Describe: Byzantine Generals Problem

22 Byzantine Generals Problem

How can the traitor lieutenant disrupt the majority vote and
cause chaos ?
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
Attack:3
Retreat: 3
Attack:3
Retreat: 3
Attack:3
Retreat: 3
Attack:3
Retreat: 3
Attack:3
Retreat: 3
Attack:3
Retreat: 3

(Q) Describe: Byzantine Generals Problem

23 Byzantine Generals Problem

How can the traitor lieutenant disrupt the majority vote and
cause chaos ?
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
I was told
Retreat
I was told
Attack

(Q) Describe: Byzantine Generals Problem

24 Byzantine Generals Problem

How can the traitor lieutenant disrupt the majority vote and
cause chaos ?
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
Attack:4
Retreat: 3
Attack:4
Retreat: 3
Attack:4
Retreat: 3
Attack:3
Retreat: 4
Attack:3
Retreat: 4
Attack:3
Retreat: 4

(Q) Describe: Byzantine Generals Problem

25 Byzantine Generals Problem

The solution to this problem is more rounds of communication
between lieutenants!
In addition to what they heard from the commander,
lieutenants also share what they heard from each other.
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
L1 L2 L3 L4 L5 L6 L7
A A A A R R R
L1 L2 L3 L4 L5 L6 L7
R A A A R R R

(Q) Describe: Byzantine Generals Problem

26 Byzantine Generals Problem

For lieutenant2:
For liutenant5:
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
L1 L3 L4 L5 L6 L7
A A A R R R
L1 L2 L3 L4 L6 L7
R A A A R R
I was told
Retreat
I was told
Attack

(Q) Describe: Byzantine Generals Problem

27 Byzantine Generals Problem

Then, the lieutenants exchange their vectors with each other
Each compose a table, where each row is a vector that is
exchanged in the previous round
Commander
Lieutenant1
Retreat!Attack!
Lieutenant2 Lieutenant3 Lieutenant4 Lieutenant5 Lieutenant6 Lieutenant7
L1 L2 L3 L4 L5 L6 L7
L1 X X X X X X
L2 A A A R R R
L3 A A A R R R
L4 A A A R R R
L5 R A A A R R
L6 R A A A R R
L7 R A A A R R
I was told
Retreat
I was told
Attack

(Q) Describe: Byzantine Generals Problem

28 Byzantine Generals Problem

Assumption: there is a default
action in case loyal lieutenants can
not agree on an action
Default = Retreat
Then, the lieutenants exchange their
vectors with each other
Lieutenant5:
L1 L2 L3 L4 L5 L6 L7
L1 X X X X X X
L2 A A A R R R
L3 A A A R R R
L4 A A A R R R
L5 R A A A R R
L6 R A A A R R
L7 R A A A R R
? A A A R R R
Apply majority() to each column

(Q) Describe: Summary of Lamport's Algorithm

29 Summary of Lamport's Algorithm

For d traitors, there has to be more than 3d generals
For d faulty nodes, the algorithm requires a total of d+1 rounds of communication (d rounds of comparing notes + order from the commander)
Lamport's algorithm provides Byzantine Fault tolerance, given that you have more than 3d generals (processes)
Byzantine fault model: a process can behave arbitrarily (even maliciously as in the Byzantine Generals problem)
There are other fault models such as crash-stop (i.e., a process can stop responding), which are less challenging to deal with than the Byzantine fault model
We will cover fault tolerance in detail during week 5
Can we somehow make sure lieutenants don't forge orders that they didn't receive?
Public key cryptography can help with this – signed messages
However, traitors can collaborate and share their keys with each other
We will cover public key cryptography next week

(Q) Describe: Applications of Byzantine

30 Applications of Byzantine

Fault Tolerance
Space flight
Space shuttle had 4 computers for decision making
They used a majoritarian system in case of disagreements
Cryptocurrency: there are financial incentives to act maliciously
Problems stem from having multiple control systems
e.g. air traffic control, airline bookings, critical manufacturing processes, stock exchange, etc.
Lamport's algorithm is expensive! – O(n2) messaging overhead each round.
Alternative: accept uncertainty, just work around it
e.g. send a predetermined number of messages and assume at least one gets to the destination
Assume at most d=1 or 2 simultaneous bad actors

(Q) Describe: Further complications

31 Further complications

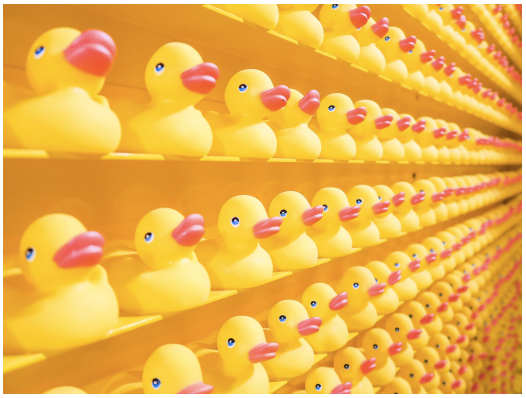
Heterogeneity
Access
Platform
Format
Administration
Changing nature
Increase/decrease in scale
Churn
Relocation
Failure

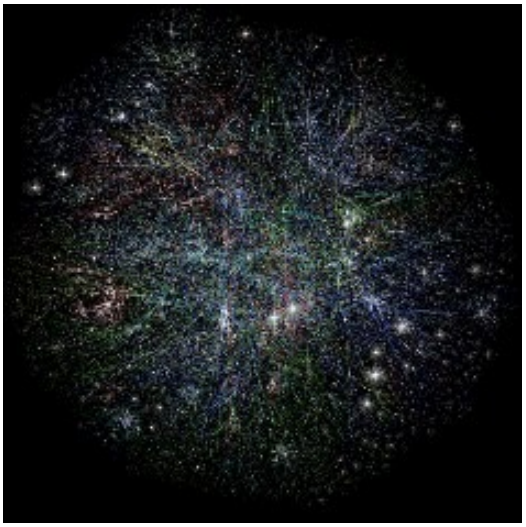
(Q) Describe: Fallacies of distributed computing

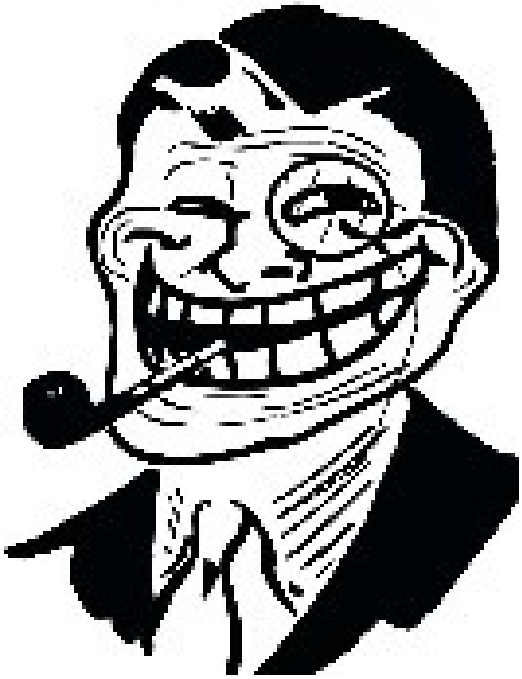
32 Fallacies of distributed computing

Asserted by Peter Deutsch (Sun Microsystems)
1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. There is one administrator
6. Transport cost is zero
7. The network is homogeneous
8. Topology doesn't change


Keiti



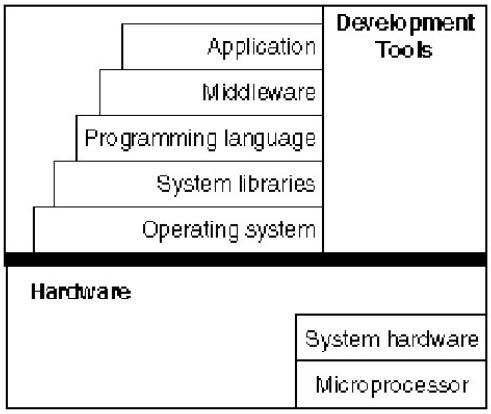




(Q) Describe: The Take-away Message?

33 The Take-away Message?

Challenging
Developing good software is difficult
Developing good distributed software is even harder
To do this you need help!
Very hard to build such systems on bare-bones devices
Strong need for software platforms - >Middleware



(Q) Describe: Middleware

34 Middleware

Rationale
Provide a high level programming abstraction
Hide the complexity associated with distributed systems
(including the underlying forms of heterogeneity)
Machine A Machine B Machine C
Network
Network OS
Services
Middleware
Kernel KernelKernel
Network OS
Services
Network OS
Services
Distributed applications
uted Applic

(Q) Describe: Goals of a Middleware Platform

35 Goals of a Middleware Platform

Middleware provides protocols to support:
Resource sharing
The ability to access and share resources in a distributed environment
Distributed locking
Transparency
The ability to view a distributed system as if it were a single computer
Varying dimensions of transparency incl. location, access, migration, etc.
Openness
The offering of services according to standard rules (syntax and semantics)
Openness provides support for the key properties of:
0 Portability: ability to run across different platforms
0 Interoperability: having separate parts harmoniously work together
Extensibility
The ability to be able to introduce new or modified functionality

(Q) Describe: Goals of a Middleware Platform

36 Goals of a Middleware Platform

Scalability: the ability to grow/shrink with respect to
Size
0 e.g. support massive growth in the number of users of a service
Geography
0 e.g. supporting systems across continents (dealing with latencies, etc.)
Administration
0 e.g. supporting systems spanning many different administrative organisations
Dependability/ Quality of Service
Security
0 Providing secure and authenticated channels, access control, key management, etc.
Fault tolerance
0 Providing highly available and resilient distributed applications and services

(Q) Describe: Styles of Middleware

37 Styles of Middleware

Client-server platforms
e.g. DCE (Distributed Computing Environment) – first!
Distributed object technology
e.g. CORBA, Java RMI
Component-based programming
e.g. Fractal, Enterprise Java Beans, OpenCOM
Microservice architecture
Loosely coupled, testable services using CI/CD
Others styles
Resource discovery platforms (e.g. Jini)
Group communication services (e.g. JGroups)
Publish-subscribe systems (e.g. JMS)
Distributed file systems, distributed transaction services, distributed document-based systems, agent-based systems, message-oriented middleware, P2P technologies, etc.

(Q) Describe: Expected Learning Outcomes

38 Expected Learning Outcomes

At the end of this session, you should:
Have an intuition of why realising good distributed systems is difficult
Byzantine generals problem as an example
Understand what the role of middleware in supporting the development of distributed applications and services
Goals of middleware systems
Some different styles of middleware

(Q) Describe: Additional Reading

39 Additional Reading

CDKB, chapter 1 section 1.5
also chapter 3 for revision
TvS, chapter 1, and sections 8.1-8.2
Lamport, Shostak, Pease, "The Byzantine Generals Problem",
ACM Transactions on Programming Languages and Systems,
July 1982. http://dl.acm.org/citation.cfm?id=357176
Fischer, Lynch, Patterson, "Impossibility of Distributed Consensus
with One Faulty Process", Journal of ACM, April 1985.
https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf
Lamport, "Paxos made simple",
http://www.cs.utexas.edu/users/lorenzo/corsi/cs380d/past/03F/notes/paxos-simple.pdf
Rotem-Gal-Oz, "Fallacies of Distributed Computing Explained",
http://www.rgoarchitects.com/Files/fallacies.pdf (2006).

(Q) Describe: ...

(Q) Describe: ...