*Where an algorithm is asked for, you may write in any suitable pseudocode. Correct syntax for any computer language is not expected.*

*Answer 3 questions.*

1. According to supporters of the object-based approach to distributed systems design, decomposition of systems into sets of communicating objects is both desirable and easily possible. Indeed, many would claim that there is a single natural object-oriented design for a given distributed application that can be produced without having to consider:

   - the environment in which the application will be deployed (e.g. whether on a LAN or globally distributed)
   - failure and performance issues.

   The alternative view says that the differences between systems which are centralised or locally distributed and those which are globally distributed are so fundamental that the scale of the application must be considered at all stages of the design as well as in the implementation.

   Produce an argument that outlines, with justification, the extent to which you agree with either of these views. Give examples where appropriate. [33]

2.
   a) State the two major benefits of replicating data. [4]

   b) What is meant by *replication transparency*? [2]

   c) Replication transparency is often used as a definition of correctness for the behaviour of a replicated system. Is this always necessarily a good definition? Illustrate your answer with an example. [4]

   d) Describe the operation of a replicated system based on weighted voting. Argue that it has both the benefits you identified in part (a). State the conditions of failure under which such a scheme is replication transparent and justify your answer. [18]

   e) How practicable is weighted voting for general-purpose use? [5]

[TURN OVER]

3.

a) Provide and explain pseudocode that illustrates how a distributed approach to mutual exclusion can be implemented using a token passing algorithm for an arbitrarily connected graph. Calculate the upper bound on how long a process must wait between desiring to enter its critical section and actually achieving it. You should assume that nodes do not fail and that messages are not lost. State any further assumptions you make. [12]

b) Now identify the issues in addressing the following and say how you would adapt your answer to (a) to take account of

   i) the failure of a token holding client.

   ii) the failure of a client that has requested but not yet received the token.

   iii) the dynamic addition of a new client wishing to access the shared resource to the graph.

   Again, state any assumptions you make. [18]

c) Does your combined solution guarantee to provide mutually exclusive access to the shared resource under all conditions? Justify your answer. [3]

4.

a) What is a mobile system? [2]

b) To what constraints must application designers pay particular attention when constructing applications for mobile systems? Explain the points you make. [10]

c) It has been argued that mobile systems are prime candidates for the incorporation of intelligent/adaptive behaviour. Why might this be so, and in what areas do you think that adaptation is particularly important for mobile systems? [15]

d) Lucent have recently developed a plastic transistor which will potentially allow the production of flexible screens and much more shock resistant hardware. Speculate as to whether and how this development will impact the nature of distributed computing as we know it today. [6]

[CONTINUED]

5.

a) Explain the purpose of each of the steps in the following protocol which allows two processes in the same security domain to exchange encrypted data. Your answer should clearly indicate the meaning of any notations used:

Step 1 $A \rightarrow AS$: $A, B, I_a$

Step 2 $AS \rightarrow A$: $\{I_a, B, K_S, \{K_S, A\}_{K_B}\}_{K_A}$

Step 3 $A \rightarrow B$: $\{K_S, A\}_{K_B}$

Step 4 $B \rightarrow A$: $\{I_B\}_{K_S}$

Step 5 $A \rightarrow B$: $\{f(I_b)\}_{K_S}$

Step 6 DATA TRANSFER ....

[10]

b) Show how the protocol can be extended to deal with entities in different security domains. [3]

c) Define the terms *block cipher* and *cipher block chaining*. Identify the main problems in encrypting electronic messages (e.g. business transactions) with block ciphers and suggest appropriate solutions. [8]

d) A company with 5000 employees is planning to use public key cryptography for its internal purposes for authentication and secrecy of e-mail.

Explain what is meant by the term *public key certificate*. Suggest how the company might deal with the production, storage and revocation of public key certificates. Your answer should indicate potential areas of weakness and where possible any safeguards that can be implemented. [12]

[END OF PAPER]

# C328 Answers

1. Any reasonable answer accepted here, provided that it is well argued. The question is quite a deep one and will require considerable thought to answer well. I would expect that answers would examine the differences that result from the transition centralised – small scale – large scale. Weaker answers will probably omit the large-scale aspect and simply address basic centralised/distributed differences.

2.
   a) Fault tolerance and performance improvement.

   b) Programmer/user should be unaware of how many replicas form up a replicated object and of where they are located. In other words, a replicated object should have single-copy consistency.

   c) No, not always necessary. In some situations it is better to allow a system to become inconsistent and fix that when time/resources and connectivity allow rather than holding out for replication transparency.

   d) Weighted voting [Gifford79] is a generalisation of unanimous update. Each replica is assigned some number of votes. In order to perform a read, a process must collect a *read quorum* of *r* votes, and to do a write it must have a *write quorum* of *w* votes. If *n* is the total of the number of votes assigned to replicas in the system, then the constraints on *r* and *w* are that:

   - $r+w > n$
   - $w > n/2$

   The first of these constraints ensures that there is a non-null intersection between every read quorum and every write quorum; that is, they must have at least one replica in common. When a write is done, the value written is tagged with a version number higher than any that has been used before (to allow a subsequent read to determine which is the current value). Since we need to collect a read quorum to determine what this is, it is usually deemed convenient to make the write quorum the largest of the two, hence the second of the constraints.

   Works under all conditions of failure in which nodes/links fail/stop (i.e. not Byzantine).

   e) OK, but in general, there are too many parameters to choose and automatic mechanisms for choosing them are still not particularly good.

3.

a) Any reasonable answer that has the following properties:

- Assume we have a group of processes; the first step is to establish a logical communication ring such that each process knows who is ‘next’ in the ring. This need not bear any relationship to the physical locations of the processes, but will work best if the links are chosen in such a way as the communication delay around the ring is least.
- A single token circulates around the ring.
- If a process has the token then it can enter its critical section.
- When a process has finished its critical section, it passes the token on to the next process in the ring.
- If a process receives a token and does not wish to enter its critical section, then it simply passes the token on to the next process in the ring, possibly after a delay.

b) This is hard, since they haven’t seen this before.

i) Requires devising an election protocol of some form which, again, has only been touched on briefly. Good answers may well point out what happens in the event of failure during this process.

ii) Catching communication failure in passing on the token, then reconfiguring – information on how to reconfigure needs to be held somewhere.

iii) A reconfiguration problem again; depends on where they stored the configuration in the first place. Since each participant stores next process in ring, must set new process up with the next process pointer before changing an existing participant to point to the new one. Good answers may well spot that there’s a problem here if no concurrency control is employed.

Since this is outside their experience, marking will be relatively generous for decent attempts at the problem, particularly those that bring out the issues.

c) Almost certainly not; there is a possibility of having multiple tokens in existence.

4.

a) A mobile system has components in it which are mobile in the sense that their connectivity relative to some fixed infrastructure changes over time. In general, to differentiate them from nomadic systems they are considered to include self-powered, hand-held, mobile computers with wireless connections though this is not an all-encompassing definition.

b) Lots of things to write about here, but should include issues like:

- Mobile elements resource poor compared to fixed hosts of same size/weight
  - Weight, power, size constraints
  - Affects computational ability as well as UI
- Mobility affects security + robustness
- Mobility gives communication uncertainty
- Power management is a concern
- It costs real money to connect.

c) Again loads of stuff – should address issues identified in (b), but things like QoS adaptation, adaptation to take account of battery power constraints, use of info on movement patterns, replication vs caching, etc.

d) An attempt to get them to think ‘outside their boxes’ about wider issues. Any reasonable answer is acceptable provided reasonably argued.

5.

a) Principals are A and B; AS is authentication server. I is a nonce – once only message tag. Ka, Kb are private keys of A and B resp. and Ks is session key – used just for this set of data exchanges between A and B.

| | |
|---|---|
| Step 1: | A sends request to AS for session key. I acts to identify response. |
| Step 2: | Response encrypted with A’s key so only A can decrypt it to reveal Ks. It also includes a token encrypted with Kb for B’s benefit, which A forwards unchanged. |
| Step 3: | Token forwarded to B – in practice would have other info too, e.g. request secure communication |
| Step 4: | B can decrypt token to discover Ks, but also sees that AS believes communication is with A (bound with Ks) because only A could have extracted token at Step 2 (needed Ka)<br><br>However, it could be a replayed message so B sends message to A with nonce Ib, encrypted with Ks |
| Step 5: | The real A can decrypt the previous message and apply the pre-arranged function (e.g. add 1) to Ib, re-encrypt it and send it back |
| Step 6: | A and B have both established liveness and encrypted communication using Ks can take place |

b) Assume two authentication servers ASa and ASb, where secure communication has been established between them with a session key Kas. Need to introduce two messages after step 1

| | | |
|---|---|---|
| Step 1a: | ASa-> ASb: | {Ks, A, B, Ia2, A}Kas |
| Step 1b: | ASb-> ASa: | { {Ks, A}Kb, Ia2, A}Kas |

ASa chooses a session key, passes it together with the identities of the principals to B, together with a nonce Ia2, encrypted with Kas. ASb creates the token for B and passes it back to ASa. This means that Asa and ASb do not have to disclose the secret keys of entities in their domain to other parties.

c) Public Key certificate contains the PK of an individual (and parameters e.g. length, type of algorithm), their identity, start and end validity dates, the identity of the issuer. The details are bound together by a strong cryptographic sumcheck that is encrypted with the secret key of the issuer. Thus anyone receiving the PK Certificate can be validated by anyone with access to the issuer's public key.

Rest of question expects students to briefly discuss risks of using software for PK/SK pair production, storage on computer systems that may be hacked or stolen, and how revocation can be dealt with particularly using directories.