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. In the foreseeable future, a cheap short-range radio chip developed by the Bluetooth consortium or one of their competitors will be built into every consumer electronic device sold. Consequently, there will be an explosion of the number of directly addressable devices as well as a radical change in the types of application demanded and a significant increase in the reliability with which those applications will have to operate.

   a) What are the fundamental problems of distributed systems? Which of these will be made worse by the scale of the environment foreseen above and which will remain unaffected. Are there any new fundamental constraints imposed in this system which are not present in more traditional distributed environments? [11]

   b) In the light of your answer to part (a), what are the technical challenges of developing applications for this environment? [11]

   c) What tools, middleware support, and system services will be necessary before application designers will be able to construct applications with the required degree of scalability and reliability? [11]

2.

   a) State Needham and Schroeder's original protocol for sending symmetric key encrypted e-mail between two parties A and B, ensuring that your answer indicates how necessary keys are obtained. Explain how it works and identify any weaknesses. [10]

   b) Develop this into a protocol where messages are sent to multiple recipients or groups in a single security domain indicating reasons for your choice. [10]

   c) An organisation with 5000 employees wishes to use key escrow to manage their keys.

      Explain what is meant by the term *key escrow,* showing why its use may be controversial. Explain how the necessary keys might be managed where messages may need to be sent encrypted and/or authenticated. [13]

[TURN OVER]

3. Distributed systems constructed using RPC-based mechanisms have been in existence for many years now. The model is one in which servers are essentially passive entities that wait for clients to call them and then respond. A less common approach to communication is based around the notion of event notification. In this, some significant change in an entity is regarded as an event. Examples might include: changes in a stock price; creation of a new file; change to a page in distributed shared memory; or entry into a building by a certain person. Other entities register interest in the occurrence of particular events or classes of events; these are then sent a notification when an event of interest takes place.

   You have been commissioned to design a system that will implement just such an event-based system. Devise an architecture for this system, describing the components necessary, their interfaces, the library support which they require and the implementation techniques and protocols to be used at all levels. Amongst implementation details, you should include the ways in which:

   - event classes are specified
   - interest in events is registered and removed
   - event triggers are dispatched, including the way in which information is packetised for transmission and the ways in which errors are handled.

   Comment on the scaling properties of your system. State any assumptions you make. [33]

4. ACME international wish to establish a stock trading system in which traders around the world either put up blocks of stock for sale or buy blocks put up by others. The system has several properties:

   - It is asynchronous.
   - When a block of stock is bought, it should be allocated to exactly one purchaser, who is decided on the basis of a distributed consensus algorithm in such a way that all available nodes in the system agree who has bought it.

   a) Describe what is meant by the term *asynchronous* above. [4]

   b) Define more rigorously what is meant by the term *consensus* above. [4]

   c) Show that it is impossible to guarantee consensus in the above situation if a single process can fail permanently (fail-stop). State any assumptions you make. [12]

   d) Outline a probabilistic crash-robust protocol and provide an informal argument, understandable by non-specialist, which shows that your protocol will operate as expected. State any assumptions you make. [13]

[CONTINUED]

5.

a) Describe two different meanings for the term *consistency*. [4]

b) In an attempt to save on message transfers, a certain distributed systems designer has proposed a one-phase atomic commitment protocol. In this, the commit process initiator (coordinator) repeatedly sends a commit or abort message, as it deems appropriate, to all of the participant processes until it receives an acknowledgement from all of them. Is this approach adequate? Under what circumstances? [10]

c) Prove that two-phase locking is adequate to ensure serialisability. [6]

d) In a shared whiteboard application it is highly desirable for consistency between different views to be maintained in a timely way, especially in the absence of failures. However, in the presence of link failures it is considered unnecessarily restrictive to prevent changes being made if a subset of the participants is unavailable.

   i) To what forms of failure is this application subject? How are those failures detected?

   ii) For each of the forms of failure identified in (i), specify and justify sensible behaviour for the application. Include details of the consequences of assuming that a failure has happened when it has not, or *vice versa*. You may make reasonable assumptions, provided you state what they are.

   iii) Say how you would implement each of the behaviours you identified in (ii).

   iv) To what sorts of application does your solution to this problem generalise? Are these distinct from the classes of application which are well-supported by transactional mechanisms or can you foresee any use for a hybrid approach? [13]

[END OF PAPER]

D53 answers

1. The key component of this question is the problem of scale; the environment envisaged involves the interconnection of hundreds of millions of devices. Consumers buying electronic goods do not expect to have to reboot them on a regular basis in the way that they expect that of computers. Consequently, there will need to be consideration of how one can make such systems very robust but in the lightest way possible, because the battery lifetime of devices can be relatively low in any case.

   a) Fundamental constraints result from geographical separation and partial failure and include issues like increased latency, loss of/errors in messages, failure of nodes/links, security, heterogeneity, etc. etc. Almost all of these problems are going to be affected by an increase in scale to a greater or lesser extent – any reasonably well argued answer is OK. Battery power is certainly one, and the degree of heterogeneity will dwarf anything to be found at present.

   b) Any reasonably well argued answer but, as above, should place emphasis on reliability with very lightweight protocols. If any other issue will be key is that of security, for which again, we need a very lightweight approach that must be combined with automatic system configuration -- nobody is going to hand-enter an X.509 certificate into a camcorder, but I don’t want my neighbour seeing my home videos.

   c) Asking for speculation. Any reasonable answer accepted, but must be well argued.

2.

   a) The scheme suggested by Needham and Schroeder is not the only possible one, but it is workable. It uses *self-authenticating messages* and timestamps. When A wishes to send to B it asks its AS to create a conversation key and wrap it up in B's key, i.e.:

      Token = {Kc, A}Kb

      The body of the message is then sent along with this token, but encrypted with Kc in the following way:

      Seg1 = Token, {TS, S1, Mess1)Kc

      Seg2 = {S2, Mess2}Kc

      Seg3 = (S3, Mess3}Kc

etc .

where:

TS is a unique timestamp added by A,

Mess1, Mess2 etc are the fragments of the message,

Seg1, Seg2, etc. are transport service packets, and, S1, S2, etc. are sequence numbers.

All distinct messages originating from the source are given a unique TS, and the recipient keeps a table of TS values and corresponding sources for a limited time related to the expected transit time of messages through the network. Any very old or replayed (duplicate) messages can thus be discarded.

b) Here I expect the above to be combined with the standard N&S authentication protocol for SKC. Thus, they would get a token for the group as a whole, or for each individual member. This would be used to encrypt the message or multiple copies resp.. The best answers will briefly discuss these two approaches indicating the benefits of lower network traffic but greater security weaknesses in the first case.

c) Escrow is where an organisation seeks to reduce risks to its business by keeping copies (possibly via an impartial third party) of information on which its business relies. This may include IPR of a vital supplier in case they go out of business or keys of employees in case they die in service so that company files can be recovered. This can be controversial where governments wish to automatically be able to escrow keys of citizens.

Second part of question relates to the difference between the escrow of encryption and signing keys.

3. This is tough. They haven't been taught directly about events, though they should have read at least one paper on it. Thus they're constructing this largely *ab initio*, and that requires a good feel for the sort of elements that it is possible to construct. Stuff like marshalling and so forth comes out of the material they have seen on RPC. However, the rest requires an ability to abstract. A good basic paper on events is 'Events in an RPC-based distributed system' by Jim Waldo et al. Sun tech report SMLI TR-95-47 (available from the Sun labs website).

4. Whilst this question is largely bookwork, it is hard bookwork which they will have had to think about in some detail in order to assimilate and understand. The question is aimed at testing their ability to understand the nature of proof and to present it in a

sensible way without being so tough that it is impossible for a graduate to undertake. The actual proof that there are no asynchronous deterministic 1-crash-resistant consensus protocols is due to Fischer, Lynch, and Paterson [JACM 32 (1985) p374—382]. The algorithm asked for is that of Bracha and Toueg [JACM 32 (1985) p824—840] or anything else that works.

5.

a) Absolute consistency/one copy serialisability vs something like eventual consistency.

b) No of course it isn't adequate in general, because it doesn't allow for unilateral abort/failure of some component processes. In some circumstances it could be good enough where either this can't (or is v. unlikely to) happen and/or we can't do much about it if it does. Likewise, some applications can tolerate inconsistency for periods, provided that they synchronise from time to time in a different way – eventual consistency.

c) Proof is by contradiction. Assume that there is a serialisable schedule that is not 2PL, then prove that no such schedule exists.

   Then there must be some schedules where T_i is ordered before T_j on one site and T_j before T_i on another (maybe transitively), or T_i is before T_j and T_j before T_i on the same site. But no such conflicting operations would be permitted by 2PL since, for them to occur without deadlock, one of T_I or T_j would have had to release a lock and then acquire another => contradiction. So 2PL ensures serialisability.

d)

   i) Node, link failures (esp. partition), lost messages and, possibly, byzantine failures.

   ii) – iv)
   The rest of this question comes under the heading of 'any reasonable answer, provided well justified'.

   What I'm after here is a logical analysis of a problem that they probably haven't thought about and that isn't in my notes. It requires them to adapt what they have learnt based on the constraints in the problem, which requires thought about what those constraints are. This is non-trivial, hence the breaking down of the question into several parts.

   Note that answers that regurgitate algorithms that would halt in the presence of partitions will not be considered favourably. However, there must be some degree of reliability in the protocol that communicates changes. Likewise, there has to be some action to restore consistent views when a partition mends, even if that is to refer back to the users a question of what to do. Hopefully, these shouldn’t pester the user too much if a partition occurs and no updates have been made on one side.