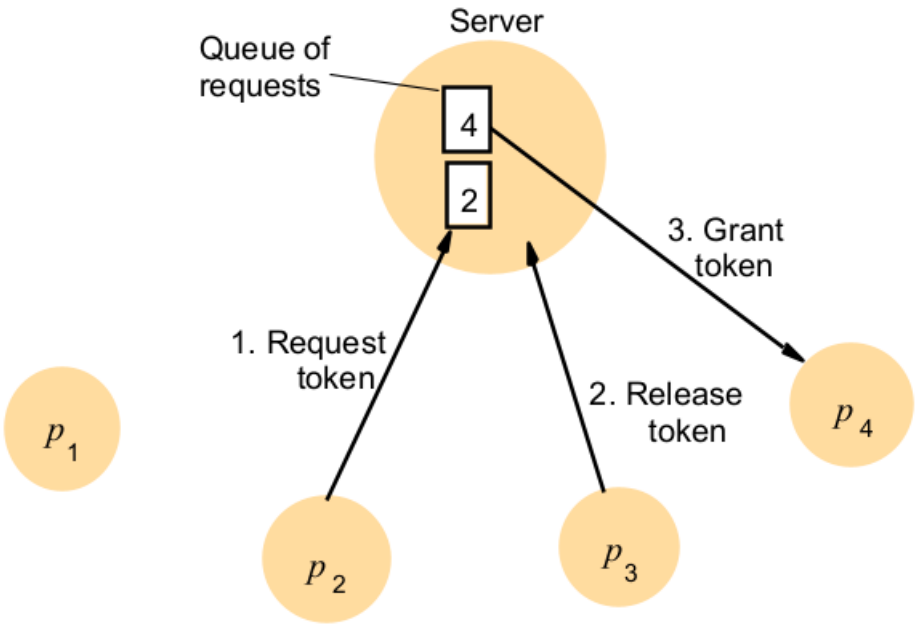
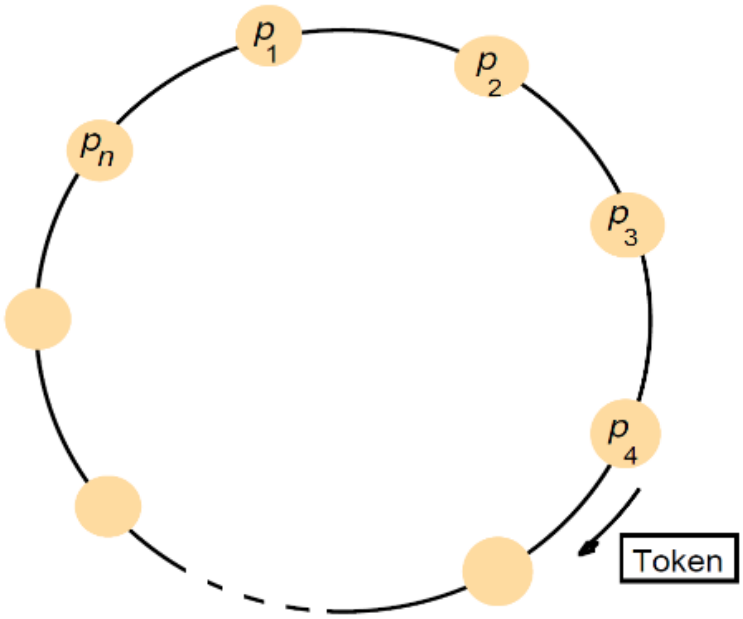
Co-ordination and Agreement

1. Explain with a neat diagram Central Server Algorithm.



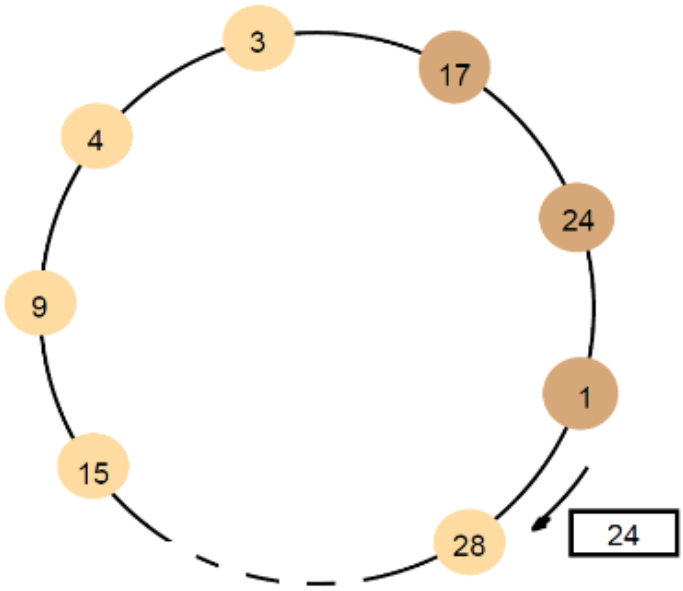
The simplest way to achieve mutual exclusion is to employ a server that grants permission to enter the critical section. The figure shows the use of this server. To enter a critical section, a process sends a request message to the server and awaits a reply from it. Conceptually, the reply constitutes a token specifying permission to enter the critical section. If no other process has the token at the time of the request, then the server replies immediately, granting the token. If the token is currently held by another process, then the server does not reply, but queues the request. When a process exits the critical section, it sends a message to the server, giving it back the token. If the queue of waiting processes is not empty, then the server chooses the oldest entry in the queue, removes it and replies to the corresponding process.

2. With a neat diagram explain Ring based Algorithm w.r.t. mutual exclusion.



One of the simplest ways to arrange mutual exclusion between the N processes without requiring an additional process is to arrange them in a logical ring. This requires only that each process p_i has a communication channel to the next process in the ring, $p_{(i+1)modN}$. The idea is that exclusion is conferred by obtaining a token in the form of a message passed from process to process in a single direction, ex: clockwise or around the ring. The ring topology may be unrelated to the physical interconnections between the underlying computers. If a process does not required to enter the critical section when it receives the token, then it immediately forwards the token to its neighbour. A process that requires the token waits until it receives it, but retain it. To exit the critical section, the process sends the token on to its neighbour.

3. With a neat diagram explain Ring based Election Algorithm.



Note: The election was started by process 17.
The highest process identifier encountered so far is 24.
Participant processes are shown in a darker colour

The algorithm of Chang and Roberts is suitable for a collection of processes arranged in a logical ring. Each process p_i has a communication channel to the next process in the ring, $p_{(i+1)modN}$, and all the the messages are sent clockwise around the ring. We assume that no failures occur, and that the system is asynchronous. The goal of this algorithm is to elect a single process called the *coordinator*, which is the process with the largest identifier.

Initially, every process is marked as non-participant in an election. Any process can begin an election. It proceeds by marking itself as a participant, placing its identifier in an election message and sending it to its clockwise neighbour.

When a process receives an election message, it compares the identifier in the message with its own. If the arrived identifier is greater, then it forwards the message to its neighbour. If the arrived identifier is smaller and the receiver is not a participant, then its substitute its own identifier in the message and forwards it. If, however, the received identifier is that of the receiver itself, then this process's identifier must be the greatest, and it becomes the coordinator. The coordinator marks itself as a non-participant once more and sends an elected message to its neighbour, announcing its election and enclosing its identity.