This question paper contains----------01----------no. of printed page

JNIT

JAGANNATH GUPTA INSTITUTE OF ENGINEERING AND TECHNOLOGY

SITAPURA, JAIPUR

First Mid Term Examination Session 2017-18

B.Tech IV Year VIII Semester

Branch : CS | Subject : DIP

Time : 2:00-3:30 p.m. | Subject Code : 8CS2A

Date : 14/02/2018 | Max. Marks : 20

Note :- Attempt any four questions out of five questions.

Q-1 Explain the concept of image representation and differentiate the image compression and representation

Q-2 What is image quantization? Explain

Q-3 What do you mean a Fourier transforms? Explain its properties.

Q-4 Describe the basic principles of image enhancement by frequency domain method.

Q-5 Explain Spatial domain method for image enhancement.

Q-1 Explain the concept of image representation and differentiate the image compression and representation

Ans.

Image Representation

An image as defined on, say a photographic film, is a continuous function of brightness values. Each point on the developed film can be associated with a gray level value representing how bright that particular point is. To store images in a computer we have to sample and quantize (digitize) the image function. Sampling refers to considering the image only at a finite number of points. And quantization refers to the representation of the gray level value at the sampling point using finite number of bits. Each image sample is called a pixel. Your typical desktop image scanner does sampling and quantization for you.

One of the simpler scheme is sampling on a regular grid of squares. We can visualize the process as overlaying an uniform grid on the image and sampling the image function at the center of each grid square. Finer the grid, better the resolution of the image; coarser the grid, the more is the observed ``pixelization'' (see Figure below). At each pixel (or at each grid square) we usually represent the gray level value using an integer ranging from 0 for black to 255 for fully white.

A digital image is essentially a collection of pixel values. There are various formats of storing an image in a file. Essentially they all involve storing the pixels values along with other relevant information about the image. The image format we will be using is called PGM or Portable Gray Map.

Image Compression

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG format and the GIF format.

The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.

Other techniques for image compression include the use of fractals and wavelets. These methods have not gained widespread acceptance for use on the Internet as of this writing. However, both methods offer promise because they offer higher compression ratios than the JPEG or GIF methods for some types of images. Another new method that may in time replace the GIF format is the PNG format.

A text file or program can be compressed without the introduction of errors, but only up to a certain extent. This is called lossless *compression*. Beyond this point, errors are introduced. In text and program files, it is crucial that compression be lossless because a single error can seriously damage the meaning of a text file, or cause a program not to run. In image compression, a small loss in quality is usually not noticeable. There is no "critical point" up to which compression works perfectly, but beyond which it becomes impossible. When there is some tolerance for loss, the compression factor can be greater than it can when there is no loss tolerance. For this reason, graphic images can be compressed more than text files or programs.

Q-2 What is image quantization? Explain

Mostly the output of the image sensors is in the form of analog signal. Now the problem is that we can not apply the digital image processing and its techniques on analog signals This is due the fact that we cannot store the output of image sensors which are in the form of analog signals because it requires infinite memory to store a signal that can infinite values. So we have to convert this analog signal into digital signal. To create a digital image we need to convert the continuous data into digital form. This conversion from analog to digital involves two processes-sampling and quantization,

Sampling:

Sampling is the reduction of a continuous signal to discrete signal. A common example is the conversion of a sound wave to a sequence of samples; a sample refers to a value or set of values at a point in time or space. It is subsystem or operation that extracts samples from continuous signals. Sampling can be done for functions varying in space, time or any other dimensions and similar results are obtained in two or more dimensions.

The equation suggests that a sampled 2-D function defined over a domain size

M▲x X N▲y can be considered as an 2-D array {f (0, 1), f (0, 1),……}

This means that the function is represented as an M X N uniformly spaced sample. Apart from the sampling interval in x and y direction, another important choice relevant to the image sampling is the spatial arrangement of the sample points called as Tessellation.

Quantization:

The values obtained by sampling a continuous function usually comprise of an infinite set or real numbers ranging from a minimum to maximum depending upon the sensors calibration. These values must be represented by a finite number of bits usually used in computer to store or process any data. In practice the sampled signal values are represented by a finite set of integer values. This is known as quantization. Every image that is seen on the monitor is actually this matrix. Each element of the matrix is called as pixel. Whenever we see the image on the screen of computer it is actual matrix consisting of N X M pixels and each pixel is considered to be a sample. Hence more the pixels, more the sample, higher the sampling rate hence better the spatial resolution. The value of each pixel is known as the grey level. Computer understands only ones and zeros. Hence these grey levels needs to be represented in terms of zeros and ones. If we have two bits to represent the grey levels, only 4 different grey levels (2) can be identified viz. 00,01 10, 11, where 00 is back, 11 is white and other two are different shades of grey. Similarly, if we have 8 bits, to represent the grey levels, we will have 256 grey levels (2). Hence more the bits, more are the grey levels and better is the total clarity (quantization). The total size of the image is N X M X m, where m is the number of bits used. As we know, more the samples and the bits, better is the image. This answer will vary from image to image. As the sampling and the quantization increase, the number of bits required to store the image increases tremendously. The clarity increases, but storage space required increases too.

Q-3 What do you mean a Fourier transforms? Explain its properties.

Ans.

The Fourier transform is a mathematical function that decomposes a waveform, which is a function of time, into the frequencies that make it up. The result produced by the Fourier transform is a complex valued function of frequency. The absolute value of the Fourier transform represents the frequency value present in the original function and its complex argument represents the phase offset of the basic sinusoidal in that frequency.

The Fourier transform is also called a generalization of the Fourier series. This term can also be applied to both the frequency domain representation and the mathematical function used. The Fourier transform helps in extending the Fourier series to non-periodic functions, which allows viewing any function as a sum of simple sinusoids.

The Fourier transform of a function f(x) is given by:

$$f(x) = \int_{-\infty}^{\infty} F(k)\, e^{2\pi i k x}\, dk$$

$$F(k) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i k x}\, dx$$

Where F(k) can be obtained using inverse Fourier transform.

Some of the properties of Fourier transform include:

- It is a linear transform – If g(t) and h(t) are two Fourier transforms given by G(f) and H(f) respectively, then the Fourier transform of the linear combination of g and t can be easily calculated.
- Time shift property – The Fourier transform of g(t–a) where a is a real number that shifts the original function has the same amount of shift in the magnitude of the spectrum.
- Modulation property – A function is modulated by another function when it is multiplied in time.
- Parseval's theorem – Fourier transform is unitary, i.e., the sum of square of a function g(t) equals the sum of the square of its Fourier transform, G(f).
- Duality – If g(t) has the Fourier transform G(f), then the Fourier transform of G(t) is g(-f).

Q-4 Describe the basic principles of image enhancement by frequency domain method.

Ans.

Techniques are based on modifying the Fourier transform of the image.

Image enhancement in the frequency domain is straightforward. We simply compute the Fourier transform of the image to be enhanced, multiply the result by a filter (rather than convolve in the spatial domain), and take the inverse transform to produce the enhanced image. i.e.

Frequency Domain is nothing more than the space defined by values of FT & frequency variables (u, v)

a. Compute DFT of the image, F(u, v)

b. Multiply F(u, v) by filter function, H(u, v)

c. Compute inverse DFT

The frequency domain methods of image enhancement are based on convolution theorem. This is represented as,

$$g(x, y) = h(x, y) * f(x, y)$$

Where. g(x, y) = Resultant image

h(x, y) = Position invariant operator

f(x, y)= Input image

The Fourier transform representation of equation above is,

$$G(u, v) = H(u, v)\, F(u, v)$$

The function H (u, v) in equation is called transfer function. It is used to boost the edges of input image f (x, y) to emphasize the high frequency components.

The different frequency domain methods for image enhancement are as follows.

1. Contrast stretching.
2. Clipping and thresholding.
3. Digital negative.
4. Intensity level slicing and
5. Bit extraction.

Q-5 Explain Spatial domain method for image enhancement.

Ans.

Filtering is a technique for modifying or enhancing an image. Spatial domain operation or filtering (the processed value for the current pixel processed value for the current pixel depends on both itself and surrounding pixels). Hence Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel.

Spatial domain Techniques are based on direct manipulation of pixels in an image

As we know that the term *spatial domain* refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression:

$$g(x,y) = \boldsymbol{T}[f(x,y)]$$

Where f(x,y) in the input image, g(x,y) is the processed image and ***T*** is as operator on f, defined over some neighborhood of (x,y)

In addition, ***T*** can operate on a set of input images.

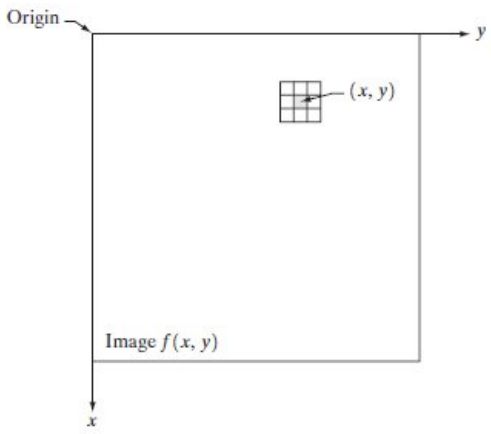
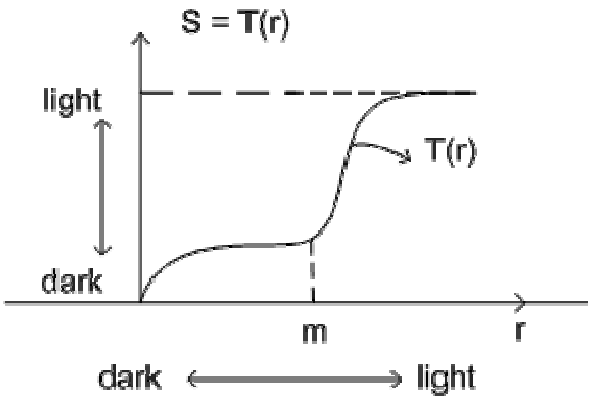


FIGURE 3 × 3 neighborhood about a point (x, y) in an image.

These techniques are based on gray level mappings, where the type of mapping used depends on the criterion chosen for enhancement. As an eg. consider the problem of enhancing the contrast of an image. Let *r* and *s* denote any gray level in the original and enhanced image respectively. Suppose that for every pixel with level *r* in original image we create a pixel in the enhanced image with level $S = T(r)$. If $T(r)$ has the form as shown



Figure

The effect of this transformation will be to produce an image of higher contrast than the original by darkening the levels below a value *m* and brightening the levels above *m* in the original pixel spectrum. The technique is refered to as contrast stretching. The values of *r* below *m* are compressed by the transformation function into a narrow range of *S*towards the dark end of the spectrum; the opposite effect takes place for values of *r* above *m* .

In the limiting case shown in figure, $T(r)$ produces a 2-level (binary) image. This is also referred to as image thresholding. Many powerful enhancement processing techniques can be formulated in the spatial domain of an image.

JNIT JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I-Mid Term Examination Session 2018

B.Tech IV Year VIII Semester

Branch: CS	**Subject: Distributed system**
Time:	**Subject Code: 8CS3A**
Date:	**Max. Marks: 20**

Questions with solution

Q1. What is distributed system? Explain its features.

Solution:

In distributed database system, the database is shared on several computers. The computers in a distributed system communicate with one another through various communication media, such as high-speed networks or telephone lines. They do not share main memory or disks.

The computers in distributed system may vary in size and function, ranging from workstations up to mainframe systems.

The computers in distributed system are referred to by a number of different names, such as Sites or Nodes depending on the context in which they are mentioned.

A distributed database system consists of single logical database which is split into different fragments. Each fragment is stored on one or more computers under the control of separate DBMS with computers connected by communication network. Each site is capable of independently processing user requests that require access to local data or file system and is also capable of performing processing on remote machines in the network.

Characteristics of Distributed System:

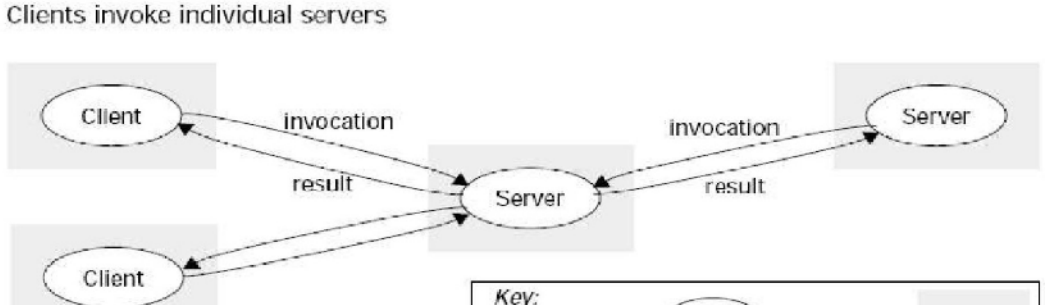
- Data set can be split in to fragments and can be distributed across different nodes within network.
- Individual data fragments can be replicated and allocated across different nodes.
- Data at each site is under control of a DBMS.
- DBMS at each site can handle local applications autonomously.
- Each DBMS site will participate in at least one global application.

Q2 Explain:

(1) Architecture model

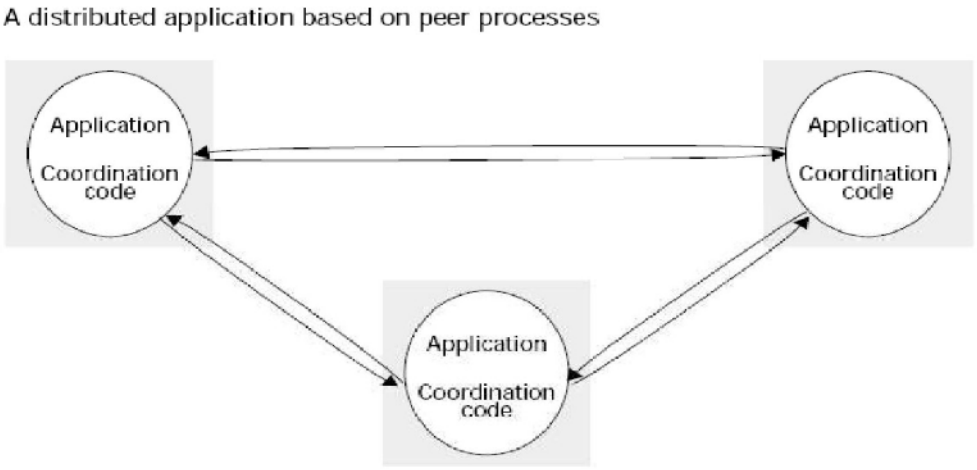
Solution:

Client-Server Architecture I



- Clients send requests to servers (i.e., invocation)
- Servers send responses to clients (i.e., result)\
- Servers may be clients of other servers
- A web server is often a client of a file server
- An Internet service is a client of a DNS server – a server that translates DNS names to IP addresses
- Potential problem: a single server is a scalability bottleneck and a single point of failure

Peer-to-Peer Architecture I



- All processes play similar roles – i.e., they interact as peers
- No central component – potentially better scalability and resiliency to failures
- Use the power of modern desktops to implement a large-scale distributed system
- Examples: Napster, Kazaa, Skype, Bittorrent

(2) State and events in distributed system
Solution:

Each process executes on a single processor, and the processors do not share memory . Each process pi in has a state si that, in general, it transforms as it executes. The process's state includes the values of all the variables within it. Its state may also include the values of any objects in its local operating system environment that it affects, such as files. We assume that processes cannot communicate with one another in any way except by sending messages through the network.So, for example, if the processes operate robot arms connected to their respective nodes in the system, then they are not allowed to communicate by shaking one another's robot hands! As each process pi executes it takes a series of actions, each of which is either amessage send or receive operation, or an operation that transforms pi 's state – one that changes one or more of the values in si. In practice, we may choose to use a high-leveldescription of the actions, according to the application. For example, if the processes in are engaged in an eCommerce application, then the actions may be ones such as 'client dispatched order message' or 'merchant server recorded transaction to log'.

We define an event to be the occurrence of a single action that a process carries out as it executes – a communication action or a state-transforming action. The sequence of events within a single process pi can be placed in a single, total ordering, which we denote by the relation i between the events.

That is, if and only if the event e occurs before e at pi . This ordering is well defined, whether or not the process is multithreaded, since we have assumed that the process executes on a single processor. Now we can define the history of process pi to be the series of events that take place within it, ordered as we have described by the relation Clocks • We have seen how to order the events at a process, but not how to timestamp them – i.e., to assign to them a date and time of day. Computers each contain their own physical clocks. These clocks are electronic devices that count oscillations occurring in a crystal at a definite frequency, and typically divide this count and store the result in a counter register. Clock devices can be programmed to generate interrupts at regular intervals in order that, for example, timeslicing can be implemented; however, we shall not concern ourselves with this aspect of clock operation.

The operating system reads the node's hardware clock value, Hi t , scales it and adds an offset so as to produce a software clock Ci t = Hi t + that approximately measures real, physical time t

for process pi . In other words, when the real time in an absolute frame of reference is t, Ci t is the reading on the software clock. For example,

Ci t could be the 64-bit value of the number of nanoseconds that have elapsed at time t since a convenient reference time. In general, the clock is not completely accurate, so Ci t will differ from t. Nonetheless, if Ci behaves sufficiently well (we shall examine the notion of clock correctness shortly), we can use its value to timestamp any event at pi . Note that successive events will correspond to different timestamps only if the clock resolution – the period between updates of the clock value – is smaller than the time interval between successive events. The rate at which events occur depends on such factors as the length of the processor instruction cycle.

Clock skew and clock drift • Computer clocks, like any others, tend not to be in perfect agreement Coordinated Universal Time • Computer clocks can be synchronized to external sources of highly accurate time. The most accurate physical clocks use atomic oscillators, whose drift rate is about one part in 1013. The output of these atomic clocks is used as the standard second has been defined as 9,192,631,770 periods of transition between the two hyperfine levels of the ground state of Caesium-133 (Cs133).

Seconds and years and other time units that we use are rooted in astronomical time. They were originally defined in terms of the rotation of the Earth on its axis and its rotation about the Sun.

However, the period of the Earth's rotation about its axis is gradually getting longer, primarily because of tidal friction; atmospheric effects and convection currents within the Earth's core also cause short-term increases and decreases in the period. So astronomical time and atomic time have a tendency to get out of step.

Coordinated Universal Time – abbreviated as UTC (from the French equivalent) – is an international standard for timekeeping. It is based on atomic time, but a so-called 'leap second' is inserted – or, more rarely, deleted – occasionally to keep it in step with astronomical time. UTC signals are synchronized and broadcast regularly from land based radio stations and satellites covering many parts of the world. For example, in the USA, the radio station WWV broadcasts time signals on several shortwave frequencies. Satellite sources include the Global Positioning System (GPS).Receivers are available commercially. Compared with 'perfect' UTC, the signals received from land-based stations have an accuracy on the order of 0.1–10 milliseconds,

depending on the station used. Signals received from GPS satellites are accurate to about 1 microsecond. Computers with receivers attached can synchronize their clocks with these timing signals.

Synchronizing physical clocks

In order to know at what time of day events occur at the processes in our distributed system – for example, for accountancy purposes – it is necessary to synchronize the processes' clocks, Ci , with an authoritative, external source of time. This is external synchronization. And if the clocks Ci are synchronized with one another to a known degree of accuracy, then we can measure the interval between two events occurring at different computers by appealing to their local clocks, even though they are not necessarily synchronized to an external source of time. This is internal synchronization. We define these two modes of synchronization more closely as follows, over an interval of real time I:

External synchronization: For a synchronization bound D 0 , and for a source S of UTC time, S t – Ci t < D, for i = 1 2 N and for all real times t in I. Another way of saying this is that the clocks Ci are accurate to within the bound D.

Internal synchronization: For a synchronization bound D 0 , Ci t – Cj t D for i j = 12 N , and for all real times t in I. Another way of saying this is that he clocks Ci agree within the bound D. Clocks that are internally synchronized are not necessarily externally synchronized, since they may drift collectively from an external source of time even though they agree with one another. However, it follows from the definitions that if the system is externally synchronized with a bound D then the same system is internally synchronized with a bound of 2D. Various notions of correctness for clocks have been suggested. It is common to define a hardware clock H to be correct if its drift rate falls within a known bound (a value derived from one supplied by the manufacturer, such as 10–6 seconds/second).

This means that the error in measuring the interval between real times t and t (t t) is bounded:

1 – t – t H t – H t 1 + t – t

This condition forbids jumps in the value of hardware clocks (during normal operation). Sometimes we also require our software clocks to obey the condition but a weaker condition of monotonicity may suffice. Monotonicity is the condition that a clock C only ever advances: t t C t C t For example, the UNIX make facility is a tool that is used to compile only those source files that have been modified since they were last compiled. The modification dates of each corresponding pair of source and object files are compared to determine this condition. If a computer whose clock was running fast set its clock back after compiling a source file but before the file was changed, the source file might appear to have been modified prior to the compilation. Erroneously, make will not recompile the source file.

We can achieve monotonicity despite the fact that a clock is found to be running fast. We need only change the rate at which updates are made to the time as given to applications. This can be achieved in software without changing the rate at which the underlying hardware clock ticks – recall that Ci t = Hi t + , where we are free to choose the values of and . A hybrid correctness condition that is sometimes applied is to require that a clock obeys the monotonicity condition, and that its drift rate is bounded between synchronization points, but to allow the clock value to jump ahead at synchronization points.

A clock that does not keep to whatever correctness conditions apply is defined to be faulty. A clock's crash failure is said to occur when the clock stops ticking altogether, any other clock failure is an arbitrary failure. A historical example of an arbitrary failure is that of a clock with the 'Y2K bug', which broke the monotonicity condition by registering the date after 31 December 1999 as 1 January 1900 instead of 2000; another example is a clock whose batteries are very low and whose drift rate suddenly becomes very large.

Note that clocks do not have to be accurate to be correct, according to the definitions. Since the goal may be internal rather than external synchronization, the criteria for correctness are only concerned with the proper functioning of the clock's 'mechanism', not its absolute setting. We now describe algorithms for external synchronization and for internal synchronization.

Logical time and logical clocks

From the point of view of any single process, events are ordered uniquely by times shown on the local clock. However, as Lamport [1978] pointed out, since we cannot synchronize clocks

perfectly across a distributed system, we cannot in general use physical time to find out the order of any arbitrary pair of events occurring within it. In general, we can use a scheme that is similar to physical causality but that applies in distributed systems to order some of the events that occur at different processes. This ordering is based on two simple and intuitively obvious points: • If two events occurred at the same process pi i = 1 2 N , then they occurred in the order in which pi observes them – this is the order i that we defined above.• Whenever a message is sent between processes, the event of sending the message occurred before the event of receiving the message. Lamport called the partial ordering obtained by generalizing these two relationships the happened-before relation. It is also sometimes known as the relation of causal ordering or potential causal ordering.

We can define the happened-before relation, denoted by , as follows: HB1: If process pi : e i e', then e e .

HB2: For any message m, send(m) receive(m) – where send(m) is the event of sending the message, and receive(m) is the event of receiving it. HB3: If e, e and e are events such that e e and e e , then e e .

Totally ordered logical clocks • Some pairs of distinct events, generated by different processes, have numerically identical Lamport timestamps. However, we can create a total order on the set of events that is, one for which all pairs of distinct events are ordered – by taking into account the identifiers of the processes at which events occur. If e is an event occurring at pi with local timestamp Ti , and e is an event occurring at pj with local timestamp Tj , we define the global logical timestamps for these events to be Ti i and Tj j , respectively. And we define Ti i Tj j if and only if either Ti Tj , or Ti = Tj and i j . This ordering has no general physical significance (because process identiiers are arbitrary), but it is sometimes useful. Lamport used it, for example, to order the entry of processes to a critical section.

Vector clocks • Mattern [1989] and Fidge [1991] developed vector clocks to overcome the shortcoming of Lamport's clocks: the fact that from L e L e we cannot conclude that e e . . A vector clock for a system of N processes is an array of N integers. Each process keeps its own vector clock, Vi , which it uses to timestamp local events. Like Lamport timestamps, processes piggyback vector timestamps on the messages they send to one another, and there are simple rules for updating the clocks:

VC1: Initially, Vi j = 0 , for i j = 1 2 N .

VC2: Just before pi timestamps an event, it sets Vi i :=Vi i + 1. VC3: pi includes the value t = Vi in every message it sends.

VC4: When pi receives a timestamp t in a message, it sets Vi j := max Vi j t j , for j = 1 2 N . Taking the componentwise maximum of two vector timestamps in this way is known as a merge operation.For a vector clock Vi , Vi i is the number of events that pi has timestamped, and Vi j j i is the number of events that have occurred at pj that have potentially affected pi . (Process pj may have timestamped more events by this point, but no information has flowed to pi about them in messages as yet.)

Clocks, Events and Process States

A distributed system consists of a collection P of N processes pi, i = 1,2,… NEach process pi has a state si consisting of its variables (which it transforms as it executes)

- Processes communicate only by messages (via a network)
- Actions of processes: Send, Receive, change own state
- Event: the occurrence of a single action that a process carries out as it executes
- Events at a single process pi, can be placed in a total ordering denoted by the relation →i between the events. i.e.e →i e' if and only if event e occurs before event e' at process pi
- A history of process pi: is a series of events ordered by →i
- history(pi) = hi =<ei0, ei1, ei2, …>

Clocks

To timestamp events, use the computer's clock • At real time, t, the OS reads the time on the computer's hardware clock Hi(t)

It calculates the time on its software clock $Ci(t)=\alpha Hi(t) + \beta$

- e.g. a 64 bit value giving nanoseconds since some base time
- Clock resolution: period between updates of the clock value

§ In general, the clock is not completely accurate – but if Ci behaves well enough, it can be used to timestamp events at pi Skew between computer clocks in a distributed system

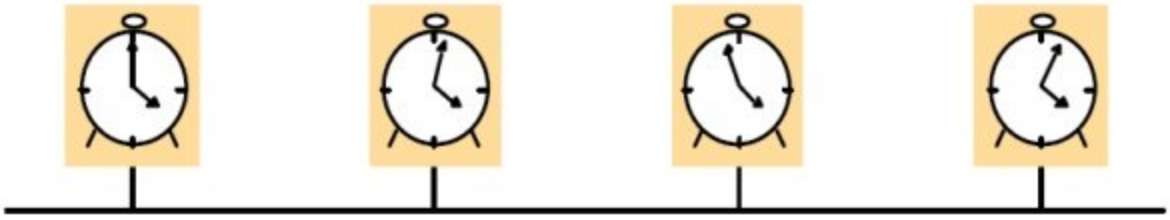


Figure 11.1

Computer clocks are not generally in perfect agreement

Clock skew: the difference between the times on two clocks (at any instant)

Computer clocks use crystal-based clocks that are subject to physical variations

Clock drift: they count time at different rates and so diverge (frequencies of oscillation differ)

Clock drift rate: the difference per unit of time from some ideal reference clock

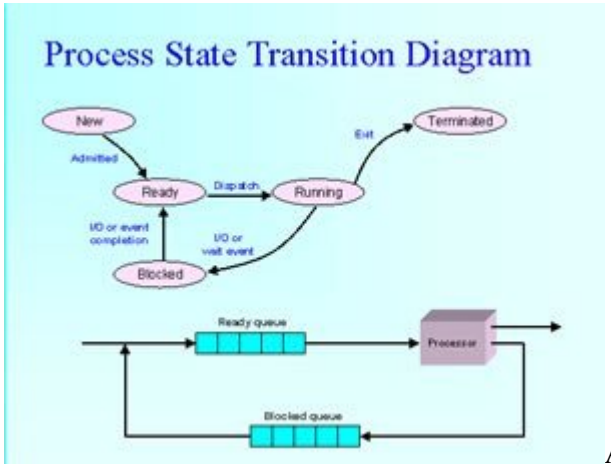
Ordinary quartz clocks drift by about 1 sec in 11-12 days. (10-6 secs/sec).

High precision quartz clocks drift rate is about 10-7 or 10-8 secs/sec

Q3. Explain process and threads.
Solution:

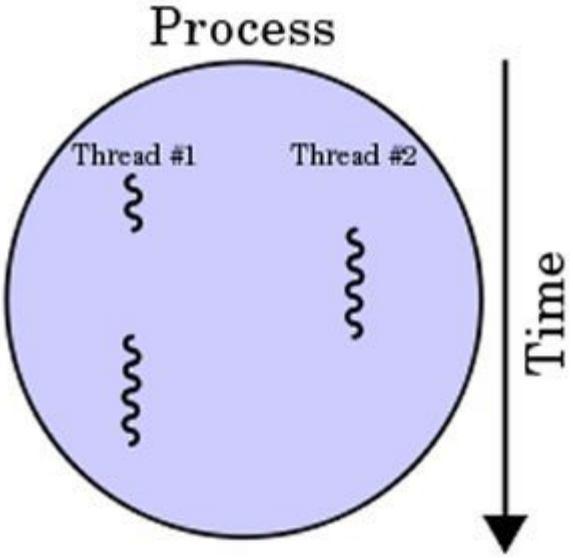
Thread and Process are two closely related terms in multi-threading. The main difference between the two terms is that the threads are a part of a process, i.e. a process may contain one or more threads, but a thread cannot contain a process.

In programming, there are two basic units of execution: processes and threads. They both execute a series of instructions. Both are initiated by a program or the operating system. This article helps to differentiate between the two units.



A process is an instance of a program that is being executed. It contains the program code and its current activity. Depending on the operating system, a process may be made up of multiple threads of execution that execute instructions concurrently. A program is a collection of instructions; a process is the actual execution of those instructions.

A process has a self-contained execution environment. It has a complete set of private basic run-time resources; in particular, each process has its own memory space. Processes are often considered similar to other programs or applications. However, the running of a single application may in fact be a set of cooperating processes. To facilitate communication between the processes, most operating systems use Inter Process Communication (IPC) resources, such as pipes and sockets. The IPC resources can also be used for communication between processes on different systems. Most applications in a virtual machine run as a single process. However, it can create additional processes using a process builder object.



In computers, a thread can execute even the smallest sequence of programmed instructions that can be managed independently by an operating system. The applications of threads and processes differ from one operating system to another. However, the threads are made of and exist within a process; every process has at least one. Multiple threads can also exist in a process and share resources, which helps in efficient communication between threads.

On a single processor, multitasking takes place as the processor switches between different threads; it is known as multithreading. The switching happens so frequently that the threads or tasks are perceived to be running at the same time. Threads can truly be concurrent on a multiprocessor or multi-core system, with every processor or core executing the separate threads simultaneously.

In summary, threads may be considered lightweight processes, as they contain simple sets of instructions and can run within a larger process. Computers can run multiple threads and processes at the same time.

Comparison between Process and Thread:

	Process	**Thread**
Definition	An executing instance of a program is called a process.	A thread is a subset of the process.
Process	It has its own copy of the data segment of the parent process.	It has direct access to the data segment of its process.
Communication	Processes must use inter-process communication to communicate with sibling processes.	Threads can directly communicate with other threads of its process.
Overheads	Processes have considerable overhead.	Threads have almost no overhead.
Creation	New processes require duplication of the parent process.	New threads are easily created.
Control	Processes can only exercise control over child processes.	Threads can exercise considerable control over threads of the same process.
Changes	Any change in the parent process does not affect child processes.	Any change in the main thread may affect the behavior of the other threads of the process.
Memory	Run in separate memory spaces.	Run in shared memory spaces.
File descriptors	Most file descriptors are not shared.	It shares file descriptors.
File system	There is no sharing of file system context.	It shares file system context.
Signal	It does not share signal handling.	It shares signal handling.
Controlled by	Process is controlled by the operating system.	Threads are controlled by programmer in a program.

Dependence	Processes are independent.	Threads are dependent.

Q4. Explain Client/Server model in detail.
Solution:

The client-server model is a distributed communication framework of network processes among service requestors, clients and service providers. The client-server connection is established through a network or the Internet.

The client-server model is a core network computing concept also building functionality for email exchange and Web/database access. Web technologies and protocols built around the client-server model are:

- Hypertext Transfer Protocol (HTTP)
- Domain Name System (DNS)
- Simple Mail Transfer Protocol (SMTP)
- Telnet

Clients include Web browsers, chat applications, and email software, among others. Servers include Web, database, application, chat and email, etc.

A server manages most processes and stores all data. A client requests specified data or processes. The server relays process output to the client. Clients sometimes handle processing, but require server data resources for completion.

The client-server model differs from a peer-to-peer (P2P) model where communicating systems are the client or server, each with equal status and responsibilities. The P2P model is decentralized networking. The client-server model is centralized networking.

One client-server model drawback is having too many client requests underrun a server and lead to improper functioning or total shutdown. Hackers often use such tactics to terminate specific organizational services through distributed denial-of-service (DDoS) attacks.

Q5 Write short note on services of distributed system.
Solution:

The distinguishing characteristics of a distributed system may be summarized as follows:

1. Concurrency

   The components of a distributed computation may run at the same time.

2. Independent failure modes

   The components of a distributed computation and the network connecting them may fail independently of each other.

3. No global time

   We assume that each component of the system has a local clock but the clocks might not record the same time. The hardware on which the clocks are based is not guaranteed to run at precisely the same rate on all components of the system, a feature called clock drift.

4. Communications delay

   It takes time for the effects of an event at one point in a distributed system to propagate throughout .

JNIT JAGANNATH GUPTA INSTITUTE OF ENGINEERING & TECHNOLOGY JAIPUR

I-Mid Term Examination Session 2018

B.Tech IV Year VIII Semester

Branch: CS **Subject: Mobile computing**

Time: **Subject Code: 8CS1**

Date: **Max. Marks: 20**

1. **What is mobile computing? Explain mechanism for Adaption.**

Mobile Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be connected to a fixed physical link. The main concept involves:

- Mobile communication
- Mobile hardware
- Mobile software

Mobile computing is characterized by a mobile device with computing capabilities operating in a wireless environment. This scenario inserts constraints not present in traditional fixed systems. Mo-bile devices have power restrictions and their computing capabilities are more restricted when com-pared to fixed devices. Also, wireless communication brings higher transmission error rates and lower and highly variable bandwidth. In order to offer acceptable services in this environment, mo-bile devices must provide some mechanisms to manage the environment modifications and to react to those changes. This problem can be mainly solved through adaptation. Adaptation in mobile computing means the ability an application or algorithm has to output different valid results, depending on the characteristics of the environment where the mobile device is located. This is not common in fixed systems, where the conditions are practically stable. In mobile systems, however, the environment is highly variable, which leads to this solution.

Adaptation Architecture

The different types of devices, network connection, and execution context of mobile systems when compared to fixed ones insert many constraints in the mobile environment not present in the fixed one. Due to those constraints, when defining an adaptation architecture in mobile computing, various aspects should be observed: data, security, quality of service, available resources, costs, performance, and broadcast and multicast issues. It is also important to consider whether or not the adaptation architecture is addressed to a specific application. Architectures addressed to specific applications often achieve better results, since the target application is known, and the designer may focus on its main characteristics and propose the best adaptation techniques for that application. However, those adaptation architectures can only be used for that target application. Other applications running upon this architecture will not benefit from it.

Generic architectures, on the other hand, do not address a target application. Instead, some general adaptation techniques are implemented. When an application is running upon a generic architecture, it will benefit only from the adaptation techniques that can be applied to it. Since these techniques were not

designed specifically for this application, they often will not achieve the best possible adaptation level. However, they will be able to achieve good adaptation levels with many applications. There is a trade off between generic and specific adaptation architectures. The most generic the architecture is, the most applications it can run. On the other hand, the most specific it is, the better results can be achieved for that specific application.

2. **Explain Principle and technique of Location management in details**

The ability to change locations while connected to the network creates a dynamic environment. This means that data, which is static for stationary computing, becomes dynamic for mobile computing. LM schemes are essentially based on users mobility and incoming call rate characteristics. The main task of LM is to keep track of a users location all the time while operating and on the move so that incoming messages (calls) can be routed to the intended recipient.

LM consists mainly of:
1. Location Tracking and Updating (Registration):A process in which an end-point initiates a change in the Location Database according to its new location. This procedure allows the main system to keep track of a users location so that for example an incoming call could be forwarded to the intended mobile user when a call exists or maybe bring a users profile near to its current location so that it could provide a user with his/her subscribed services.

2. Location Finding (Paging): The process of which the network initiates a query for an end-pointís location. This process is implemented by the system sending beacons to all cells so that one of the cells could locate the user. This might also result in an update to the location register. As we can see, the main difference between location tracking and paging is in who initiates the change. While location tracking is initiated by a mobile host, paging is initiated by the base system. Most LM techniques use a combination of location tracking and location finding to select the best trade-off between the update overhead and the paging delay.

LM methods are classified into two groups:

1. Group one includes methods based on network architecture and algorithms, mainly on processing capabilities of the system.

2. Group two includes methods based on learning processes (i.e. which require the collection of statistics on subscribers mobility behavior).

Classification of LBSs: LBSs can be classified as Reactive LBSs and Proactive LBSs :

1. Reactive LBSs: Reactive LBSs are always explicitly activated by the user. The interaction between LBS and users is roughly as follows: the user first invokes the service and establishes a service session, either via a mobile device or a desktop PC. The user then requests for certain functions or information whereupon the service gathers location data (either of the user or of another target person), processes it, and returns the location-dependent result to the user. This

request/response cycle may be repeated several times before the session is finally terminated. Thus, a reactive LBS is characterized by a synchronous interaction pattern between user and service.

2. Proactive LBSs: Proactive LBSs are automatically initialized as soon as a predefined location event occurs, for example, if a user enters, approaches, or leaves a certain point of interest or if he/she approaches, meets, or leaves another target. Thus, proactive services are not explicitly requested by the user, but the interaction between them happens asynchronously. In contrast to reactive LBSs where the user is only located once, proactive LBSs require to permanently track a user in order to detect location events.

3. Explain the concept of reliable Agent Transfer.

ANS: Mobile agent is a software program that migrates from one node to another while performing given tasks on behalf of a user. Mobile agent technology is used to develop many distributed applications. A mobile agent can communicate independently with other agents, with users, and with the hosts in the network and preserves all of its state when it moves from one network node to another. However, the mobility of agents makes it more difficult to trace mobile agents and transfer messages reliably. Therefore, a reliable communication protocol that provides efficient location management and reliable message delivery is the key issue to the design of mobile agent system.

Architecture of Mobile Agent:
A mobile agent consists of the program code and the program execution state (the current values of variables, next instruction to be executed, etc.). Initially a mobile agent resides on a computer called the home machine. The agent is then dispatched to execute on a remote computer called a mobile agent host (a mobile agent host is also called mobile agent platform or mobile agent server). When a mobile agent is dispatched the entire code of the mobile agent and the execution state of the mobile agent is transferred to the next host.

The host provides a suitable execution environment for the mobile agent to execute. The mobile agent uses resources (CPU, memory, etc.) of the host to perform its task. After completing its task on the host, the mobile agent migrates to another computer. Since the state information is also transferred from the host, mobile agents can resume the execution of the code from where they left off in the previous host instead of having to restart execution from the beginning. This continues until the mobile agent returns to its home machine after completing execution on the last machine in its itinerary.

The agent would be programmed to visit a number of bookstores and find the best deals on books we need. Mobile agents play an important role in the development of active and dynamically managed networks and distributed systems. It provides several advantages , most notably of which are: reduction in network traffic, asynchronous autonomous interaction, overcome network latency, robustness and fault tolerance and its support for heterogeneous environments.

4. **What is Mobile web caching? Explain mobile cache maintenance schemes:**

ANS: A **web cache** (or HTTP cache) is an information technology for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce server lag. A web cache system stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met.[1] A web cache system can refer either to an appliance, or to a computer program. Caching is the term for storing reusable responses in order to make subsequent requests faster. There are many different types of caching available, each of which has its own characteristics. Application caches and memory caches are both popular for their ability to speed up certain responses.

Web caching, the focus of this guide, is a different type of cache. Web caching is a core design feature of the HTTP protocol meant to minimize network traffic while improving the perceived responsiveness of the system as a whole. Caches are found at every level of a content's journey from the original server to the browser.

Web caching works by caching the HTTP responses for requests according to certain rules. Subsequent requests for cached content can then be fulfilled from a cache closer to the user instead of sending the request all the way back to the web server.

Benefits

Effective caching aids both content consumers and content providers. Some of the benefits that caching brings to content delivery are:

- **Decreased network costs**: Content can be cached at various points in the network path between the content consumer and content origin. When the content is cached closer to the consumer, requests will not cause much additional network activity beyond the cache.
- **Improved responsiveness**: Caching enables content to be retrieved faster because an entire network round trip is not necessary. Caches maintained close to the user, like the browser cache, can make this retrieval nearly instantaneous.
- **Increased performance on the same hardware**: For the server where the content originated, more performance can be squeezed from the same hardware by allowing aggressive caching. The content owner can leverage the powerful servers along the delivery path to take the brunt of certain content loads.
- **Availability of content during network interruptions**: With certain policies, caching can be used to serve content to end users even when it may be unavailable for short periods of time from the origin servers.

Terminology

When dealing with caching, there are a few terms that you are likely to come across that might be unfamiliar. Some of the more common ones are below:

- **Origin server**: The origin server is the original location of the content. If you are acting as the web server administrator, this is the machine that you control. It is responsible for serving any

content that could not be retrieved from a cache along the request route and for setting the caching policy for all content.

- **Cache hit ratio**: A cache's effectiveness is measured in terms of its cache hit ratio or hit rate. This is a ratio of the requests able to be retrieved from a cache to the total requests made. A high cache hit ratio means that a high percentage of the content was able to be retrieved from the cache. This is usually the desired outcome for most administrators.
- **Freshness**: Freshness is a term used to describe whether an item within a cache is still considered a candidate to serve to a client. Content in a cache will only be used to respond if it is within the freshness time frame specified by the caching policy.
- **Stale content**: Items in the cache expire according to the cache freshness settings in the caching policy. Expired content is "stale". In general, expired content cannot be used to respond to client requests. The origin server must be re-contacted to retrieve the new content or at least verify that the cached content is still accurate.
- **Validation**: Stale items in the cache can be validated in order to refresh their expiration time. Validation involves checking in with the origin server to see if the cached content still represents the most recent version of item.
- **Invalidation**: Invalidation is the process of removing content from the cache before its specified expiration date. This is necessary if the item has been changed on the origin server and having an outdated item in cache would cause significant issues for the client.

Certain content lends itself more readily to caching than others. Some very cache-friendly content for most sites are:

- Logos and brand images
- Non-rotating images in general (navigation icons, for example)
- Style sheets
- General Javascript files
- Downloadable Content
- Media Files

Caching Headers

Caching policy is dependent upon two different factors. The caching entity itself gets to decide whether or not to cache acceptable content. It can decide to cache less than it is allowed to cache, but never more.

The majority of caching behavior is determined by the caching policy, which is set by the content owner. These policies are mainly articulated through the use of specific HTTP headers.

Through various iterations of the HTTP protocol, a few different cache-focused headers have arisen with varying levels of sophistication. The ones you probably still need to pay attention to are below:

- **Expires**: The Expires header is very straight-forward, although fairly limited in scope. Basically, it sets a time in the future when the content will expire. At this point, any requests for the same

content will have to go back to the origin server. This header is probably best used only as a fall back.

- **Cache-Control**: This is the more modern replacement for the Expires header. It is well supported and implements a much more flexible design. In almost all cases, this is preferable to Expires, but it may not hurt to set both values. We will discuss the specifics of the options you can set with Cache-Control a bit later.
- **Etag**: The Etag header is used with cache validation. The origin can provide a unique Etag for an item when it initially serves the content. When a cache needs to validate the content it has on-hand upon expiration, it can send back the Etag it has for the content. The origin will either tell the cache that the content is the same, or send the updated content (with the new Etag).
- **Last-Modified**: This header specifies the last time that the item was modified. This may be used as part of the validation strategy to ensure fresh content.
- **Content-Length**: While not specifically involved in caching, the Content-Length header is important to set when defining caching policies. Certain software will refuse to cache content if it does not know in advanced the size of the content it will need to reserve space for.
- **Vary**: A cache typically uses the requested host and the path to the resource as the key with which to store the cache item. The Vary header can be used to tell caches to pay attention to an additional header when deciding whether a request is for the same item. This is most commonly used to tell caches to key by the Accept-Encoding header as well, so that the cache will know to differentiate between compressed and uncompressed content.

Cache control

HTTP defines three basic mechanisms for controlling caches: freshness, validation, and invalidation.

Freshness

allows a response to be used without re-checking it on the origin server, and can be controlled by both the server and the client. For example, the Expires response header gives a date when the document becomes stale, and the Cache-Control: max-age directive tells the cache how many seconds the response is fresh for.

Validation

can be used to check whether a cached response is still good after it becomes stale. For example, if the response has a Last-Modified header, a cache can make a conditional request using the If-Modified-Since header to see if it has changed. The ETag (entity tag) mechanism also allows for both strong and weak validation.

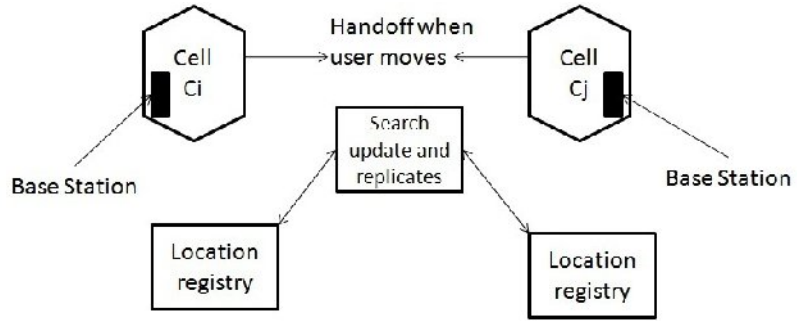
Invalidation

is usually a side effect of another request that passes through the cache. For example, if a URL associated with a cached response subsequently gets a POST, PUT or DELETE request, the cached response will be invalidated.

5.. Write shote note on:

1. Mobility Management

Mobility management is nothing but the technique in which uninterrupted signal connectivity is maintained, when a mobile device changes location from cell Ci to Cj or from network Ni to network Nj . Following are the two important points to ensure constant connectivity:
1. Infrastructure management that connects two or more cells or networks.
2. Location management and registration management by handoff when mobile devices move from one cell to another cell.
The technique of mobility management is as shown in following diagram:



Mobility Management is one of the major functionality of a GSM or a UMTS network. Mobile devices inform the cellular network, whenever it moves from one location area to another. Mobiles devices detects the location area codes. When a mobile finds that the location area code is different from its last update, it performs another update by sending to the network, a location update request, together with its previous location, and its Temporary Mobile Subscriber Identity (TMSI) as well. Thus a subscriber enjoys an uninterrupted access to the network. Roaming is the fundamental mobility management procedures of all cellular networks. Roaming is referred as the ability for a customer to automatically make and receive voice calls, send and receive data, or access other services, including home data services, when travelling outside the geographical coverage area of the home network, by means of using a visited network. This can be possible by using a communication terminal. Roaming is always technically supported by mobility management, authentication, authorization as well as billing procedures.

2. Broadcast disk Scheduling:

In a push-based information system, the server must construct a broadcast "program" to meet the needs of the client population. In the simplest scenario, given an indication of the data items that are desired by each client listening to the broadcast, the server would simply take the union of the requests and broadcast the resulting set of data items cyclicly. When an application running on a client needs a data item, it first attempts to retrieve that item from the local memory or disk. If the desired item is not found, then the client monitors the broadcast and waits for the item to arrive. With the flat broadcast, the expected wait

for an item on the broadcast is the same for all items (namely, half a broadcast period) regardless of their relative importance to the clients. This "flat" approach has been adopted in earlier work on broadcast-based database systems such as Data cycle. Alternatively, the server can broadcast different items with differing frequency. Such a broadcast program can emphasize the most popular items and deemphasize the less popular ones. Theoretically, the generation of such non flat broadcast programs can be addressed as a bandwidth allocation problem; given all of the client access probabilities, the server determines the optimal percentage of the broadcast bandwidth that should be allocated to each item. The broadcast program can then be generated randomly according to those bandwidth allocations, such that the average inter-arrival time between two instances of the same item matches the needs of the client population. However, such a random broadcast will not be optimal in terms of minimizing expected delay due to the variance in the inter-arrival times.