به نام خدا

حل المسائل کتاب سیستمهای توزیعی تننباوم نگارش ۲۰۰۷

(سوالاتی که به عنوان تکلیف درسی از سوی استاد مطرح شد در این جزوه نیامده است)

این سوالها به زبان اصلی به این مستند اضافه شده اند

دوستان همکار در تهیه این جزوه:

Ahmadi –Aslani – Avakians – behvandi – estedlal – fathi- ghaffari – ghanbari- ghorbanian- jafari- jalaledini- javadi- Karimi- khosravi- kianpour- Maleki- Moazzeni- Niaee- Nikoyi- RahAmooz- ramezani- Safavi- sotoode- tamjidi- zarifi -Siahi

فصل ۱- سیستم توزیعی

✓ ۲- نقش Middleware در سیستم توزیعی چیست؟
برای افزایش transparency که در سیستم های عامل شبکه (الان) وجود ندارد به عبارت دیگر هدف MW بهبود بخشیدن به دید تک سیستمی (single- system) است که یک سیستم توزیع شده باید داشته باشد.

توضیح: MW لایه ای است بین سیستم عامل و شبکه که امکان تک سیستمی بودن را در سیستم برقرار می کند . در تک سیستمی هر کاربر فکر می کند که کل سیستم عامل متعلق به خودش است و همه اطلاعات از سیستم محلی می آید.

✓ ۴- معنی Transparency (Distribution) را توضیح دهید و مثال هایی از انواع مختلف transparency بزنید:
Transparency توزیعی پدیده ای است که بوسیله آن جنبه های توزیعی در یک سیستم از کابرها و برنامه های کاربردی پنهان می ماند.مثال ها:

(توضیحات فارسی از مترجم است)

✓ access transparency(دسترسی محلی و راه دور یکسان به نظر برسد.)
Location transparency(مکان یک سرویس مشخص نباشد)
migration transparency (بدون اینکه کاربر متوجه شود کد و یا پروسس را انتقال دهیم)
replication transparency(از سرور ها بدون اینکه کاربر متوجه شود کپی کنیم واز کپی ها به جای اصلی استفاده کنیم)
relocation transparency (تغییر محل سرویس حین استفاده مختل نشود)
concurrency transparency(پردازش های مختلف مزاحم هم نمی شوند و همه با هم کار می کنند و هریک فکر می کنند که تنها مورد شان حضور دارند.)
failure transparency(اگر یکی از سیستم ها از کار بیفتد، سیستم به کار خود ادامه دهند)

✓ ۵- چرا گاهی اوقات خیلی سخت است که در یک سیستم توزیعی occurrence (رخداد) و recovery (بازیافت)را از خرابی ها پنهان کنیم؟
عموما تشخیص اینکه آیا یک سرور واقعا down شده است یا در جواب دادن سرعت کمی دارد ، غیر ممکن است.درنتیجه، یک سیستم ممکن است مجبور شود گزارش دهد که یک سرویس موجود نیست درصورتی که در واقعیت ممکن است سرور کند باشد.

✓ ۶- چرا همیشه ایده خوبی نیست که هدف ، پیاده سازی بالاترین درجه ممکن transparency باشد؟
زیرا هدف داشتن بالاترین درجهtransparency منجر به از دست رفتن کارایی (performance) قابل ملاحظه ای می شود که مورد قبول کاربران نمی باشد.

1. **Q:** An alternative definition for a distributed system is that of a collection of independent computers providing the view of being a *single system*, that is, it is completely hidden from users that there even multiple computers. Give an example where this view would come in very handy.

   A: What immediately comes to mind is parallel computing. If one could design programs that run without any serious modifications on distributed systems that appear to be the same as nondistributed systems, life would be so much easier. Achieving a single-system view is by now considered virtually impossible when performance is in play.

3. **Q:** Many networked systems are organized in terms of a back office and a front office. How does organizations match with the coherent view we demand for a distributed system?

   A: A mistake easily made is to assume that a distributed system as operating in an organization, should be spread across the entire organization. In practice, we see distributed systems being installed along the way that an organization is split up. In this sense, we could have a distributed system supporting back-office procedures and processes, as well as a separate front-office system. Of course, the two may be coupled, but there is no reason for letting this coupling be fully transparent.

10. **Q:** Explain what is meant by a virtual organization and give a hint on how such organizations could be implemented.

    A: A virtual organization (VO) defines a group of users/applications that have access to a specified group of resources, which may be distributed across many different computers, owned by many different organizations. In effect, a VO defines who has access to what. This also suggests that the resources should keep an account of foreign users along with their access rights. This can often be done using standard access control mechanisms (like the rwx bits in UNIX), although foreign users may need to have a special account. The latter complicates matters considerably.

11. **Q:** When a transaction is aborted, we have said that the world is restored to its previous state, as though the transaction had never happened. We lied. Give an example where resetting the world is impossible.

    A: Any situation in which physical I/O has occurred cannot be reset. For example, if the process has printed some output, the ink cannot be removed from the paper. Also, in a system that controls any kind of industrial process, it is usually impossible to undo work that has been done.

✓ **13. Q:** We argued that distribution transparancy may not be in place for pervasive systems. This statement is not true for all types of transparencies. Give an example.

A: Think of migration transparency. In mnay pervasive systems, components are mobile and will need to re-establish connections when moving from one access point to another. Preferably, such handovers should be completely transparent to the user. Likewise, it can be argued that many other types of transparencies should be supported as well. However, what should not be hidden is a user is possibly accessing resources that are directly coupled to the user's current environment.

✓ **14. Q:** We already gave some examples of distributed pervasive systems: home systems, electronic health-care systems, and sensor networks. Extend this list with more examples.

A: There are quite a few other examples of pervasive systems. Think of large-scale wireless mesh networks in cities or neighborhoods that provide services such as Internet access, but also form the basis for other services like a news system. There are systems for habitat monitoring (as in wildlife resorts), electronic jails by which offenders are continuously monitored, large-scale integrated sports systems, office systems deploying active badges to know about the whereabouts of their employees, and so on.

openness

۷- سیستم توزیع شده open چیست و چه امتیازاتی دارد؟
جواب) سیستم open سرویسهایی بر اساس قوانین تعریف شده مشخص ارائه می دهد.سیستم open قابلیت interoperability با سیستمهای open دیگر را داردو همچنین به application ها اجازه می دهد تا براحتی بین پیاده سازیهای مختلف از سیستمهای یکسان پورت شوند.

۸- منظور از سیستم scalable را به طور دقیق توضیح دهید.
سیستمی scalable است که با توجه به تعداد component ها، اندازه جغرافیایی یا تعداد و اندازه administrative domain ها، در صورت رشد یک یا چند بعدی، بر روی performance کاهش قابل تاثیری نگذارد.(به طوری که performance قابل قبول نباشد.)

۹- Scaling با تکنیک‌های مختلفی انجام پذیر است . این تکنیک‌ها کدامند ؟
Scaling می‌تواند به‌واسطه‌ی Distribution و replication پیاده سازی شود.

به‌علاوه به وسیله Hiding Communication Latency

۱۲- اجرای تراکنشهای تودرتونیازمند نوعی هماهنگی است.توضیح دهید هماهگ کننده چه کاری باید انجام دهد.
هماهنگ کننده باید تضمین کند که اگر یکی از تراکنشهای تودرتواز بین برود سایر ارتباطات زیرمجموعه نیز دچار اختلال شده و از بین می روند .همچنین هماهنگ کننده در صورتی اعلام موفقیت می نماید که تمام زیر ارتباطات به نتیجه رسیده باشند.بنابر این nested transaction منتظر رسیدن به نتیجه می شود تا زمانی که هماهنگ کننده این موفقیت را تایید نماید.

فصل ۲- سیستم توزیعی

+ interface
+ App
+ DB

2- What is a three-tiered client-server architecture?
معماري سرويس گيرنده– سرويس دهنده سه بخشي شامل سه لايه منطقي است كه در اصل در سه ماشين مجزا پياده مي شود . بالاترين لايه شامل واسط كاربر سرويس گيرنده ، لايه مياني شامل application واقعي ، پايين ترين لايه، داده ای را كه استفاده ميشود پياده سازي مي كند .

3-What is the difference between a vertical distribution and a horizontal distribution?
توزيع عمودي برمي گردد به توزيع لايه هاي مختلف در معماري چند بخشي بين چندين ماشين برمي گردد. در اصل هر لايه در ماشيني متفاوت پياده مي شود . توزيع افقي با توزيع يك لايه ميان ماشينهاي مختلف سر و كار دارد، مانند توزيع يك بانك اطلاعاتي (a single database)

7- Considering that a node in CAN knows the coordinates of its immediate neighbors, a reasonable routing policy would be to forward a message to the closest node toward the destination. How good is this policy?
همانطور كه در مثال پرسش قبل ديده شد، نياز ندارد به بهترين مسير منجر شود. اگر نود (۳و۰و۲و۰) از اين روش براي ارسال پيام به نود (٦و۰و۹و۰) پيروي كند آنرا به نود (۲و۰و۷و۰) خواهد رساند.

4. **Q:** Consider a chain of processes P_1, P_2, ..., P_n implementing a multitiered client-server architecture. Process P_i is client of process P_{i+1}, and P_i will return a reply to P_{i-1} only after receiving a reply from P_{i+1}. What are the main problems with this organization when taking a look at the request-reply performance at process P_1?

   A: Performance can be expected to be bad for large n. The problem is that each communication between two successive layers is, in principle, between two different machines. Consequently, the performance between P_1 and P_2 may also be determined by $n-2$ request-reply interactions between the other layers. Another problem is that if one machine in the chain performs badly or is even temporarily unreachable, then this will immediately degrade the performance at the highest level.

5. **Q:** In a structured overlay network, messages are routed according to the topology of the overlay. What is an important disadvantage of this approach?

   A: The problem is that we are dealing only with *logical* paths. It may very well be the case that two nodes A and B which are neighbors in the overlay network are physically placed far apart. As a consequence, the logically short path between A and B may require routing a message along a very long path in the underlying physical network.

6. **Q:** Consider the CAN network from Fig. 2-8 How would you route a message from the node with coordinates (0.2,0.3) to the one with coordinates (0.9,0.6)?

   A: There are several possibilities, but if we want to follow the shortest path according to a Euclidean distance, we should follow the route (0.2,0.3) $\rightarrow$ (0.6,0.7) $\rightarrow$ (0.9,0.6), which has a distance of 0.882. The alternative route (0.2,0.3) $\rightarrow$ (0.7,0.2) $\rightarrow$ (0.9,0.6) has a distance of 0.957.

8. **Q:** Consider an unstructured overlay network in which each node randomly chooses c neighbors. If P and Q are both neighbors of R, what is the probability that they are also neighbors of each other?

   A: Consider a network of N nodes. If each node chooses c neighbors at random, then the probability that P will choose Q, or Q chooses P is roughly $2c / (N-1)$.

9. **Q:** Consider again an unstructured overlay network in which every node randomly chooses c neighbors. To search for a file, a node floods a request to its neighbors and requests those to flood the request once more. How many nodes will be reached?

   A: An easy upper bound can be computed as $c \times (c-1)$, but in that case we ignore the fact that neighbors of node P can be each other's neighbor as well. The probability q that a neighbor of P will flood a message only to nonneighbors of P is 1 minus the probability of sending it to at least one neighbor of P:

$$q = 1 - \sum_{k=1}^{c-1} \binom{c-1}{k} \left(\frac{c}{N-1}\right)^k \left(1 - \frac{c}{N-1}\right)^{c-1-k}$$

   In that case, this flooding strategy will reach $c \times q\,(c-1)$ nodes. For example, with $c = 20$ and $N = 10,000$, a query will be flooded to 365.817 nodes.

✓ – ۱۰ Not every node in a peer-to-peer network should become superpeer. What are reasonable requirements that a superpeer should meet?

اولا ، نود باید در حد بالایی قابل دسترسي باشد ، مثل خيلي نودهاي ديگر كه متكي به آن هستند . همچنين بايد قابليت كافي براى پردازش درخواستها داشته باشد. .شايد مهمترين موضوع اين باشد كه (از نظر ديگران) براى اينكه بتواند كارش را به خوب انجام دهد قابل اطمينان باشد.

✓ **11. Q:** Consider a BitTorrent system in which each node has an outgoing link with a bandwidth capacity B_{out} and an incoming link with bandwidth capacity B_{in}. Some of these nodes (called seeds) voluntarily offer files to be downloaded by others. What is the maximum download capacity of a BitTorrent client if we assume that it can contact at most one seed at a time?

جواب) ظرفیت نود های seed بین client ها به اشتراک گذاشته می شود.اگر فرض کنیم که s تعدادseeder ها و N تعداد client ها باشد و هر client بطور تصادفی یکی از seeder ها را بر دارد.ظرفیت خروجی مشترک seeder ها S*Bout است و به هر client ظرفیت download S*Bout/N داده می شود.بعلاوه اگر client ها به یکدیگر کمک نمایند هر کدامشان می توانند chunk هایی با نرخ Bout با فرض Bout> Bin دانلود نمایند.
بدلیل سیاستtit-for-tat عموما ظرفیت دانلود یک bittorrent client بر اساس ظرفیت خروجی محدود می شود در نتیجه ظرفیت کل دانلود S*Bout/N+Bout خواهد شد.

17. Q: Sketch a solution to automatically determine the best trace length for predicting replication policies in Globule.

جواب) یک سرور اصلی باید از trace های Ti تاTi+1 استفاده کند تا پیش بینی اش از سیاست را برای آن پر یود چک کند.براحتی می توان فهمید که آیا سیاستی که بر اساس الگوهای دسترسی واقعی انتخاب شده است با انی که بر اساس در خواستها در پریود Ti-1 تا Ti انتخاب شده یکسان است یا خیر. بدینصورت سرور میتواند خطای پیش بینی را محاسبه نماید. با تغییر طول trace سرور اصلی می تواند برای آن پیش بینی ای که مینیمال است طول را بیابد. بدین طریق طول trace بهینه را بطور اتوماتیک تعیین می کنیم.

1. Q: If a client and a server are placed far apart, we may see network latency dominating overall performance. How can we tackle this problem?

A: It really depends on how the client is organized. It may be possible to divide the client-side code into smaller parts that can run separately. In that case, when one part is waiting for the server to respond, we can schedule another part. Alternatively, we may be able to rearrange the client so that it can do other work after having sent a request to the server. This last solution effectively replaces the synchronous client-server communication with asynchronous one-way communication.

12. **Q:** Give a compelling (technical) argument why the tit-for-tat policy as used in BitTorrent is far from optimal for file sharing in the Internet.

A: The reasoning is relatively simple. Most BitTorrent clients are operated behind asymmetric links such as provided by ADSL or cable modems. In general, clients are offered a high incoming bandwidth capacity, but no one really expects that clients have services to offer. BitTorrent does not make this assumption, and turns clients into collaborative servers. Having symmetric connections is then a much better match for the tit-for-tat policy.

✓ **13.** **Q:** We gave two examples of using interceptors in adaptive middleware. What other examples come to mind?

A: There are several. For example, we could use an interceptor to support mobility. In that case, a request-level interceptor would first look up the current location of a referenced object before the call is forwarded. Likewise, an interceptor can be used to transparently encrypt messages when security is at stake. Another example is when logging is needed. Instead of letting this be handled by the application, we could simply insert a method-specific interceptor that would record specific events before passing a call to the referenced object. More of such example will easily come to mind.

✓ **14.** **Q:** To what extent are interceptors dependent on the middleware where they are deployed?

A: In general, interceptors will be highly middleware-dependent. If we consider Fig. 2-0, it is easy to see why: the client stub will most likely be tightly bound to the lower level interfaces offered by the middleware, just as message-level interceptors will be highly dependent on the interaction between middleware and the local operating system. Nevertheless, it is possible to standardize these interfaces, opening the road to developing portable interceptors, albeit often for a specific class of middleware. This last approach has been followed for CORBA.

15. **Q:** Modern cars are stuffed with electronic devices. Give some examples of feed-back control systems in cars.

A: One obvious one is cruise control. On the one hand this subsystem measures current speed, and when it changes from the required setting, the car is slowed down or speeded up. The anti-lock braking systems (ABS) is another example. By pulsating the brakes of a car, while at the same time regulating the pressure that each wheel is exerting, it is possible to continue steering without losing control because the wheels are blocked. A last example is formed by the closed circuit of sensors that monitor engine condition. As soon as a dangerous state is reached, a car may come to an automatic halt to prevent the worst.

16. **Q:** Give an example of a self-managing system in which the analysis component is completely distributed or even hidden.

A: We already came across this type of system: in unstructured peer-to-peer systems where nodes exchange membership information, we saw how a topology could be generated. The analysis component consists of dropping certain links that will not help converge to the intended topology. Similar examples can be found in other such systems as we referred to as well.

فصل ۳- سیستم توزیعی

✓ 2) would it make sense to limit the number of threads in a server process?

جواب) بله-به ۲ دلیل.
thread ها برای set up پشته شان نیاز به حا فظه دارند. بنا بر این تعداد زیاد thread ها حافظه زیادی را از سرور برای اینکه درست کار کند مصرف می کند
دلیل مهم دیگر این است که از نظرسیستم عامل thread های مستقل تمایل دارند که بطور نا مرتب عمل کنند.در سیستم virtual memory ساختن یک مجموعه کاری نسبتا پایدار بدلیل page fault و I/O ممکن است سخت باشد .داشتن تعداد زیاد thread ها ممکن است منجر به کاهش کارایی بدلیل page fault شود و کارایی در مقایسه با single thread کاهش پیدا می کند.
Thread ها برای تنظیمات پشته هایشان نیاز به حافظه دارند در نتیجه داشتن تعداد زیادی Thred حافظه زیادی از Server را برای کارکرد درست مصرف می کنند.دلیل مهمتر دیگر اینکه در یک سیستم عامل thread های مستقل تمایل به وضعیت بی نظمی دارند دریک سیستم virtual memory ممکن است ساختن یک مجموعه کاری پایداربه دلیل Page fault و I/O مشکل باشد داشتن تعداد زیادی Thredممکن است منجر به کاهش کارائی به دلیل page thrashing گردد در دومورد فوق حتی اگر همه چیز در حافظه قرارگیرد باز هم ممکن است در آن حافظه الگوی بی نظمی درایجاد Cache مشاهده شود و کارایی در مقایسه با single thread کاهش پیدا می کند.

✓ 5)having only a single lightweight process per process is also not such a good idea.why not?

جواب) چون دراینصورت ما تنها user-level threads خواهیم داشت یعنی هر blocking system call کل process را بلاک خواهد نمود.

7. Q: X designates a user's terminal as hosting the server, while the application is referred to as the client. Does this make sense?

در مثال سيستم X ، ترمينال يك كاربر بعنوان ميزبان سرور و برنامه هاي كاربردي (Applications) بعنوان مشتري (client) انتخاب مي شوند. آيا اين مفهوم درست است؟
بله- اگرچه در ابتدا يك مقدار گيج كننده بنظر مي رسد، ايده كلي بدين صورت است كه سرور سخت افزار را كنترل مي نمايد و برنامه ها مي توانند با ارسال درخواست ها آن سخت افزار را دستكاري كنند(اداره نمایند) (manipulate). با اين ديدگاه X window server بايد بر روي ماشين كاربر مستقر گردد و برنامه كاربردي مانند (Client) آن عمل می کنند.

8. Q: The X protocol suffers from scalability problems. How can these problems be tackled?

اساسا مشکل Scalibility بر دو قسم است : ۱- مشکل گسترش پذیری از نظر تعداد ؛ مشکلی که در این مقوله احساس می شود این است که پهنای باند بسیار زیادی نیاز است. با استفاده از تکنیکهای فشرده سازی پهنای باند به طور قابل ملاحظه ای کاهش خواهد یافت .
۲- مشکل گسترش پذیری جغرافیایی همانطورکه یک برنامه کاربردی و نمایشگرآن نیاز دارند که با هم کاملا همگام باشند.با استفاده از تکنیکهای کش کردن بطور کاملا موثری وضعیت کنونی نمایشگر درسمت برنامه کاربردی حفظ می شود؛ و برنامه کاربردی می تواند

✓ **1. Q:** In this problem you are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 15 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in a cache in main memory. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded?

A: In the single-threaded case, the cache hits take 15 msec and cache misses take 90 msec. The weighted average is $2/3 \times 15 + 1/3 \times 90$. Thus the mean request takes 40 msec and the server can do 25 per second. For a multithreaded server, all the waiting for the disk is overlapped, so every request takes 15 msec, and the server can handle 66 2/3 requests per second.

3. Q: In the text, we described a multithreaded file server, showing why it is better than a single-threaded server and a finite-state machine server. Are there any circumstances in which a single-threaded server might be better? Give an example.

A: Yes. If the server is entirely CPU bound, there is no need to have multiple threads. It may just add unnecessary complexity. As an example, consider a telephone directory assistance number for an area with 1 million people. If each (name, telephone number) record is, say, 64 characters, the entire database takes 64 megabytes, and can easily be kept in the server's memory to provide fast lookup.

✓ **4. Q:** Statically associating only a single thread with a lightweight process is not such a good idea. Why not?

A: Such an association effectively reduces to having only kernel-level threads, implying that much of the performance gain of having threads in the first place, is lost.

✓ **6. Q:** Describe a simple scheme in which there are as many lightweight processes as there are runnable threads.

A: Start with only a single LWP and let it select a runnable thread. When a runnable thread has been found, the LWP creates another LWP to look for a next thread to execute. If no runnable thread is found, the LWP destroys itself.

9. Q: Proxies can support replication transparency by invoking each replica, as explained in the text. Can (the server side of) an application be subject to a replicated calls?

A: Yes: consider a replicated object A invoking another (nonreplicated) object B. If A consists of k replicas, an invocation of B will be done by each replica. However, B should normally be invoked only once. Special measures are needed to handle such replicated invocations.

✓ **10. Q:** Constructing a concurrent server by spawning a process has some advantages and disadvantages compared to multithreaded servers. Mention a few.

A: An important advantage is that separate processes are protected against each other, which may prove to be necessary as in the case of a superserver handling completely independent services. On the other hand, process spawning is a relatively costly operation that can be saved when using multithreaded servers. Also, if processes do need to communicate, then using threads is much cheaper as in many cases we can avoid having the kernel implement the communication.

✓ فصل ۳سئوال **۱۱** : طرح يك سرور multithreaded را بكشيد كه پروتكل هاي چندگانه را با استفاده سوكت ها به عنوان رابط لايه transport اش به سيستم عامل زيرينش پشتيباني مي كند.
پاسخ: يك طراحي نسبتا ساده اين است كه يك Single thread داشته باشيم كه منتظر پيام transport ورودي (TPDUs) باشد. اگر فرض كنيم كه (header) هر TPDU شامل يك عدد باشد كه مشخص كننده پروتكل لايه بالايي باشد، thread مي تواند payload را بگيرد و آن را به ماژول آن پروتكل بفرستد. هر كدام ازماژول ها يك thread مجزا دارد كه منتظر اين payload است، كه به شكل يك درخواست وارده عمل مي كند. بعد از handleكردن درخواست، يك پيام پاسخ (response) به T فرستاده مي شود، كه به ترتيب آن را در يك پيام لايه transport مي پيچد و به مقصد مورد نظر ارسال مي كند.

۱۲) چگونه می توان مانع یک برنامه کاربردی از حیله گری مدیر پنجره گردید و در نتیجه قادر شد تا بطور کامل صفحه ای را نامنظم کرد.
ج) با استفاده از MicroKernel که سیستم windowing شامل window manager اجرا شده است در چنین وضعیتی عملکرد همه Windowها مستلزم عبور از طریق Kernel است. امر مسلم آن است که این ذات ارسال مدل Client-Server به یک کامپیوتر تنها است.

✓ **۱۳** - سرویس دهنده‌ای که اتصال TCP/IP را با سرویس گیرنده نگه می‌دارد ، بدون حالت است یا حالتمند ؟
به ظاهر server هیچگونه اطلاعات دیگری را روی Client نگه نمی‌دارد ، دراین حالت Stateless می‌باشد .
حقیقت این است که نه Server بلکه لایه Transport در server وضعیت Client را نگه می دارد در سیستم عامل‌های محلی نیز نگهداری به همین شکل بوده و ارتباطی به server ندارد . {توضیح خاتم نادران : Tcp/ip در لایه transport است نیازی به نگهداری در لایه Application و Middleware ندراد}

✓ **۱۴** - فرض کنید سرویس دهنده وب ، جدولی را نگه‌داری می‌کند که در آن ، آدرس‌های IP سرویس گیرنده به صفحات وبی که اخیرا دستیابی شده‌اند ، نگاشت می‌شود . وقتی سرویس گیرنده به سرویس دهنده متصل می‌شود، سرویس دهنده ، سرویس گیرنده را در جدول جستجو می‌کند ، و اگر پیدا کرد ، صفحه‌ی ثبت شده را برمی‌گرداند . این سرویس دهنده بدون حالت است یا حالتمند ؟
ما یقیناً می‌توانیم بگوییم Stateless server است . در طراحی stateless مهم این نیست server اطلاعاتی از Client خود داشته باشد مهم این است که اطلاعات برای عملکرد صحیح لازم است؟ در این مثال ، اگر جدول به هر دلیلی از بین برود ، client و server می‌توانند بدون هیچ مشکلی همچنان با هم ارتباط داشته باشند . در طراحی stateful چنین تراکنشی فقط بعد از اینکه خطاهای احتمالی recover شود انجام می پذیرد.

strong mobility

۱۵- سیاربودن قوی در سیستم‌های لینوکس ، به این صورت پشتیبانی می‌شود که به فرآیند اجازه داده می‌شود فرزند را در ماشین راه دور fork کند . عملکرد آن را شرح دهید .

Fork کردن در Unix یعنی که یک تصویر کامل از والدین [خصوصیات و نحوه‌ی ارتباطات] در childها کپی شده است ، یعنی فرزندان فقط زمانی ادامه پیدا میکنند که از محل انشعاب فراخوانی شوند . روشی مشابه همین میتواند برای Remote clone استفاده شود ، platform مقصد می بایست شبیه والدین فعال فراهم شود . اولین گام چگونگی ذخیره سازی منابع سیستم عامل و ایجاد فرآیندهای مناسب و map حافظه برای Processهای child جدید است . بعد از انجام این موارد ، تصویر [مشخصات] والدین (درحافظه) می‌تواند کپی شود ، و child میتواند فعال شود . (کاملا واضح است که از جزییات صرفنظر کرده‌ایم .)

✓ **16. Q:** In Fig. 3-18 it is suggested that strong mobility cannot be combined with executing migrated code in a target process. Give a counterexample.

A: If strong mobility takes place through thread migration, it should be possible to have a migrated thread be executed in the context of the target process.

✓ **17. Q:** Consider a process P that requires access to file F which is locally available on the machine where P is currently running. When P moves to another machine, it still requires access to F. If the file-to-machine binding is fixed, how could the systemwide reference to F be implemented?

بهترین راه حل این است که یک پروسس جداگانه به نام Q ایجاد کنیم که درخواست های از راه دور به F را تحویل می گیرد و پروسس P از طریق همان اینترفیس قبلی و به عنوان مثال در قالب یک پروکسی به فایل F دسترسی پیدا می کند.در واقع Q به صورت یک فایل سرور عمل میکند.

توضیح نادران : تنها راه ممکن GR است در ماشین جدید Process مثل p داشته باشیم که requestهای ریموت به P را انجام دهد. نمی توان Rb انجام داد چون Process می خواهد به فایل دسترسی داشته باشد p به F دسترسی دارد اگر p به ماشین دیگری برود همچنان به f نیاز دارد.

18. Q: Describe in detail how TCP packets flow in the case of TCP handoff, along with the information on source and destination addresses in the various headers.

راه های مختلفی برای انجام این کار وجود دارد ولی ساده ترین راه این است که front end ، three way handshake را اجرا نماید و از آنجا بسته ها را به سرور انتخاب شده فورورد کند و آن سرور نیز TCP PDU هایی را که آدرس مبدا آنها front end باشد ارسال می کند.روش دیگر فرستادن اولین بسته به Server است لازم به ذکر است که در این روش front end در یک لوپ می افتد.مزیت این روش این است که سرور انتخاب شده خود TCP state های مورد نیاز از جمله شماره های توالی را درست میکند در حالی که در روش قبل سرور این اطلاعات را از front end دریافت می کرد .

فصل ۴- سیستم توزیعی

✓ **1. Q:** In many layered protocols, each layer has its own header. Surely it would be more efficient to have a single header at the front of each message with all the control in it than all these separate headers. Why is this not done?

زیرا هر لایه باید مستقل از لایه های دیگر باشد ودیتایی که از لایه K+1 به لایه پایینی K ارسال می شود هم شامل دیتا ونیز هدر می باشداما لایه K نمیداند که کدام هدر وکدام دیتا می باشد.همچنین با داشتن یک هدر بزرگ واحدکه تمام لایه ها میتوانند از آن بخوانند و به آن بنویسند transparency را ازبین خواهد برد و تغییرات در پروتکل یک لایه برای دیگر لایه ها قابل مشاهده می شود که چندان رضایت بخش نیست.

3. **Q:** A reliable multicast service allows a sender to reliably pass messages to a collection of receivers. Does such a service belong to a middleware layer, or should it be part of a lower-level layer?

اصولا یک سرویس Multicast قابل اعتماد به راحتی میتواند جزو لایه انتقال و یا شبکه باشد .به عنوان مثال سرویس چندیخشی غیر قابل اعتماد IP در لایه شبکه

تمرین ۳ فصل ٤ :اصولا یک سرویس Multicast قابل اعتماد به راحتی میتواند جزو لایه انتقال و یا شبکه باشد .به عنوان مثال سرویس چندپخشی غیر قابل اعتماد IP در لایه شبکه قرار داده شده است .با توجه به اینکه اکنون این سرویس ها در دسترس نیستند به طور کلی با استفاده از سرویس های لایه انتقال پیاده سازی شده اند که آنها را در middle ware جای می دهد .اگربحث قابلیت گسترش پذیری را مورد توجه قرار دهیم ،قابلیت اعتماد تنها زمانی تضمین می شود که تمامی نیازمندی های app مورد توجه قرار گیرد و این چالش بزرگی در پیاده سازی چنین سرویس هایی در لایه های بالاتر می باشد.

۶- زبان C ساختاري به نام Union دارد، كه در آن فيلدي از يك ركورد (كه در زبان C به آن struct گفته مي شود) مي تواند انواع ساختار داده را شامل باشد. در زمان runtime، به طورمشخص نمي‌توان گفت كه چه نوع ساختار داده‌اي در آن وجود دارد. آيا اين قابليت زبان C مفهومي از RPC در بردارد؟ پاسخ خود را توضيح دهيد.

اگر سيستم در لحظه Run time نتواند نوع ساختارفيلد را بدرستي مشخص كند، نمي تواند آنرا براي ارسال مرتب كند. بنابراين union ها نمي توانند RPC را پشتيباني كنند مگر اينكه فيلدtag اي در آن وجود داشته باشد كه دقيقا مشخص نمايد نوع متغييرفيلد چه مي باشد. اين فيلد tag نبايستي تحت كنترل كاربر باشد.

۸- به جاي اينكه يك سرور خود را با يك Daemon در DCE رجيستر كند، مي توان endpoint يكساني را براي هميشه به آن سرورنسبت داد . از آن پس آن endpoint مي تواند به عنوان مرجعي به objectهاي فضاي آدرس سرور (Server's address space) مورد استفاده قرار گردد. اشكال اصلي اين روش چيست؟

اشكال عمده اين روش اين است كه اختصاص دادن Objects ها به سرور به صورت پويا، به مراتب سخت‌تراز حالت قبل است. علاوه براين، endpointهاي زيادي به جاي يك endpoint (در حالت daemon) بايد تثبيت شوند. براي ماشين‌هايي كه تعداد زيادي سرور دارند، اختصاص دادن ايستاي endpointها روش مناسبي نيست.

۹- آيا تفاوت قائل شدن بين RPC ايستا و پويا مفيد مي باشد؟

به دليل مشابه، تمييز دادن بين اين دو از طريق فراخواني از راه دورobjectها مفيد مي باشد چون بسادگي ميزان انعطاف پذيري را افزايش مي دهد. اما اشكال آن اينست كه بخش زيادي از Distribution Transparency ، كه دليل اصلي بوجود آمدن RPC بود، از بين مي‌رود.

۱۰- توضيح دهيد كه چگونه ارتباط Connectionless بين client و Server با استفاده از socket صورت مي گيرد.

هم client و هم Server، هر دو يك socket ايجاد مي كنند، اما تنها server اين socket را به يك endpoint محلي bind مي نمايد. سپس server يك blocking *read* call { يك كال read را به صورت مسدود شده } ايجاد مي‌نمايد كه در آن منتظر داده ورودي از هر clientاي مي‌باشد. همچنين بعد از ايجاد socket، client بسادگي يك blocking call را براي نوشتن اطلاعات درون سرور ايجاد مي نمايد. هيچ لزومي به close نمودن ارتباط وجود ندارد.

2. Q: Why are transport-level communication services often inappropriate for building distributed applications?

A: They hardly offer distribution transparency meaning that application developers are required to pay significant attention to implementing communication, often leading to proprietary solutions. The effect is that distributed applications, for example, built directly on top of sockets are difficult to port and to interoperate with other applications.

4. Q: Consider a procedure *incr* with two integer parameters. The procedure adds one to each parameter. Now suppose that it is called with the same variable twice, for example, as *incr*(*i*, *i*). If *i* is initially 0, what value will it have afterward if call-by-reference is used? How about if copy/restore is used?

→ در این سوال RPC مد نظر نیست.

A: If call by reference is used, a pointer to *i* is passed to *incr*. It will be incremented two times, so the final result will be two. However, with copy/restore, *i* will be passed by value twice, each value initially 0. Both will be incremented, so both will now be 1. Now both will be copied back, with the second copy overwriting the first one. The final value will be 1, not 2.

6. Q: One way to handle parameter conversion in RPC systems is to have each machine send parameters in its native representation, with the other one doing the translation, if need be. The native system could be indicated by a code in the first byte. However, since locating the first byte in the first word is precisely the problem, can this actually work?

A: First of all, when one computer sends byte 0, it always arrives in byte 0. Thus the destination computer can simply access byte 0 (using a byte instruction) and the code will be in it. It does not matter whether this is the low-order byte or the high-order byte. An alternative scheme is to put the code in all the bytes of the first word. Then no matter which byte is examined, the code will be there.

7. Q: Assume a client calls an asynchronous RPC to a server, and subsequently waits until the server returns a result using another asynchronous RPC. Is this approach the same as letting the client execute a normal RPC? What if we replace the asynchronous RPCs with asynchronous RPCs?

A: No, this is not the same. An asynchronous RPC returns an acknowledgment to the caller, meaning that after the first call by the client, an additional message is sent across the network. Likewise, the server is acknowledged that its response has been delivered to the client. Two asynchronous RPCs may be the same, provided reliable communication is guaranteed. This is generally not the case.

12. Q: Suppose that you could make use of only transient asynchronous communication primitives, including only an asynchronous receive primitive. How would you implement primitives for transient *synchronous* communication?

A: Consider a synchronous send primitive. A simple implementation is to send a message to the server using asynchronous communication, and subsequently let the caller continuously poll for an incoming acknowledgment or response from the server. If we assume that the local operating system stores incoming messages into a local buffer, then an alternative implementation is to block the caller until it receives a signal from the operating system that a message has arrived, after which the caller does an asynchronous receive.

13. **Q:** Suppose that you could make use of only transient synchronous communication primitives. How would you implement primitives for transient *asynchronous* communication?

A: This situation is actually simpler. An asynchronous send is implemented by having a caller append its message to a buffer that is shared with a process that handles the actual message transfer. Each time a client appends a message to the buffer, it wakes up the send process, which subsequently removes the message from the buffer and sends it its destination using a blocking call to the original send primitive. The receiver is implemented in a similar fashion by offering a buffer that can be checked for incoming messages by an application.

14. **Q:** Does it make sense to implement persistent asynchronous communication by means of RPCs?

A: Yes, but only on a hop-to-hop basis in which a process managing a queue passes a message to a next queue manager by means of an RPC. Effectively, the service offered by a queue manager to another is the storage of a message. The calling queue manager is offered a proxy implementation of the interface to the remote queue, possibly receiving a status indicating the success or failure of each operation. In this way, even queue managers see only queues and no further communication.

15. **Q:** In the text we stated that in order to automatically start a process to fetch messages from an input queue, a daemon is often used that monitors the input queue. Give an alternative implementation that does not make use of a daemon.

A: A simple scheme is to let a process on the receiver side check for any incoming messages each time that process puts a message in its own queue.

16. **Q:** Routing tables in IBM WebSphere, and in many other message-queuing systems, are configured manually. Describe a simple way to do this automatically.

A: The simplest implementation is to have a centralized component in which the topology of the queuing network is maintained. That component simply calculates all best routes between pairs of queue managers using a known routing algorithm, and subsequently generates routing tables for each queue manager. These tables can be downloaded by each manager separately. This approach works in queuing networks where there are only relatively few, but possibly widely dispersed, queue managers.

A more sophisticated approach is to decentralize the routing algorithm, by having each queue manager discover the network topology, and calculate its own best routes to other managers. Such solutions are widely applied in computer networks. There is no principle objection for applying them to message-queuing networks.

17. **Q:** With persistent communication, a receiver generally has its own local buffer where messages can be stored when the receiver is not executing. To create such a buffer, we may need to specify its size. Give an argument why this is preferable, as well as one against specification of the size.

A: Having the user specify the size makes its implementation easier. The system creates a buffer of the specified size and is done. Buffer management becomes easy. However, if the buffer fills up, messages may be lost. The alternative is to have the communication system manage buffer size, starting with some default size, but then growing (or shrinking) buffers as need be. This method reduces the chance of having to discard messages for lack of room, but requires much more work of the system.

✓ ۱۱- تفاوت بين mpi_bsend و mpi_isend ابتدايي در MPI را بيان كنيد.

mpi_bsend اوليه از ارتباطات بافري استفاده مي‌كند تا از طريق آن call كننده كل بافر را، كه شامل پيغامهاي آماده فرستادن است، به سيستم در حال اجراي محلي MPI پاس نمايد. وقتي كه عمليات call كردن به پايان رسيد، پيغامها يا انتقال داده مي‌شوند و يا در يك بافر محلي كپي مي‌شوند. در مقابل، در mpi_isend ، call كننده تنها pointer اشاره كننده به پيغام را به سيستم در حال اجراي محلي MPI پاس مي نمايد، كه بعد از آن سيستم بلافاصله قادر به ادامه دادن كار خود مي‌باشد. Call كننده مسئول است تا زماني كه پيغام كپي نشده يا انتقال نيافته است بر روي پيغامي كه به آن اشاره شده است، {اطلاعات ديگري} overwrite نشود.

✓ ۱۸- توضيح دهيد كه چرا ارتباطات سنكرون گذرا مشكلات گستر ش پذيري ذاتي داشته و چگونه اين مشكلات رفع مي‌شوند.

مشکل محدودیت گسترش پذیری جغرافیایی می باشد. چون لازمه ارتباط سنکرون این است که فراخوان کننده تا زمانی که پیغامش دریافت شود، مسدود باشد ؛ و ممکن است در موافعی که دریافت کننده{از لحاظ جغرافیایی} در مکان دوري قرار دارد، مدت زمان زیادی قبل از اینکه، فراخوان کننده قادر به ادامه کار باشد، صرف این موضوع شود.
تنها راه حل این مشکل طراحی برنامه فراخوانی به گونه‌اي است كه فراخوان كننده در حين برقراری ارتباط، کارهای مفید دیگری براي انجام دادن داشته باشد و به طور موثر {درعمل} نوعي ارتباط آسنكرون{ دراين مدت} برقرار نمايد.

✓ ۱۹- مثالي بزنيد كه در آن multicasting براي جريانهاي داده‌اي گسسته نيز مفيد مي‌باشد.

فرستادن یک فایل حجیم برای تعداد زیادی کاربر نمونه ای از مزایای multicasting در این مورد است. به عنوان مثال هنگامي كه به بروزرسانی سایتهای پشتیبان (mirror) براي سرویسهای وب یا توزیعهای نرم افزاری مي پردازيم.

۲۰ – فرض كنيد كه در يك شبكه حسگر دماهاي اندازه گيري شده توسط حسگر timestamp زده نمي‌شوند، اما بلافاصله به اپراتور فرستاده مي‌شوند. آيا تنها تضمين نمودن يك حداكثر تاخير انتها به انتها كفايت مي‌كند؟

نه براستي؛ اگر فرض كنيم كه اپراتور همچنان نياز دارد بداند كه اندازه گيری چه زماني اتفاق افتاده است.
در اين حالت، زماني كه اندازه‌گيری دريافت شد ،برچسب زماني‌اي{a timestamp} مي‌تواند الصاق شود، اما اين بدان معناست كه ما بايد ضمانتهايي برای حداقل تاخير انتها به انتها {end-to-end} داشته باشيم .

✓ ۲۱- چگونه می توانید حداکثر تاخیر انتها به انتها را وقتی که مجموعه ای از کامپیوترها به صورت یک حلقه (ring) (فیزیکی یا منطقی) سازماندهی شده اند، تضمین کنید؟

اجازه میدهیم که یک token حلقه را دور بزنند. هر کامپیوتر اجازه دارد ، فقط زمانی که token را در اختیار دارد و از طریق حلقه داده ارسال کند (در مسیر یکسان با token). به علاوه، هیچ کامپیوتری اجازه ندارد token را بیش از T ثانیه نگه دارد. در عمل، اگر ما فرض کنیم که ارتباطات بین دو کامپیوتر مجاور محدود و متناهی باشد، آنگاه token حداکثر زمان چرخش خواهد داشت، که متناظر است با حداکثر تاخیر انتها به انتها برای هر بسته ارسال شده.

۲۲- چگونه می توانید حداقل تاخیر انتها به انتها را وقتی که مجموعه ای از کامپیوترها به صورت یک حلقه (ring) (فیزیکی یا منطقی) سازماندهی شده اند، تضمین کنید؟

به طرز عجیبی این کار خیلی سخت تر از تضمین حداکثر تاخیر است. مشکل اینجاست که کامپیوتر گیرنده نباید، اصولا {به طور کلی}، قبل از **حداقل** زمان سپری شده، داده دریافت کند. تنها راه حل این است که بسته ها را به هر اندازه {زماني} که لازم است بافر کنیم. بافر کردن هم می تواند در سمت فرستنده یا گیرنده یا هر کجا بین اندو، برای مثال، در ایستگاه های میانی انجام شود. بهترین مکان برای بافر کردن موقت داده، گیرنده است. زیرا در ان نقطه هیچ مانع پیش بینی نشده ای که بتواند رسیدن داده را به تاخیر بیاندازد، وجود ندارد. گیرنده تنها باید داده را از بافرش خارج کند و ان را توسط یک مکانیزم ساده زمانبندی به برنامه (application) ارسال کند. مشکل این روش این است که گنجایش بافرسازی به اندازه کافی باید فراهم شود.

۲۳- با وجود اینکه multicasting از لحاظ فنی مقرون به صرفه {عملي} است، پشتیبانی خیلی کمی برای پیاده سازی آن در اینترنت وجود دارد. راه حل این مشکل باید در مدل های کاری عملی جستجو شود: هیچ کس واقعا نمی داند چگونه از multicasting درآمد به دست آورد. چه طرحی می توانید ابداع کنید؟

مشکل عمدتا توسط ISP ها به وجود می آید، به این خاطر که آنها دلیلی برای ذخیره پهنای باند نمی بینند (مشتریهان آنها در هر حال هزینه را پرداخت می کنند). اما موضوع می تواند در سناریو های زیر تغییر کند. یک سرویس broadcasting اینترنت برای QOS (کیفیت سرویس) مشخصي، تهیه شده توسط ISP های مختلف، هزینه پرداخت می کند. هرکدام از این ISP ها یک افت در درآمدهایشان مشاهده خواهند کرد وقتی که نتوانند احتیاجات و ملزومات آن QOS ها را فراهم آورند. در این زمان، آنها می توانند انگیزه شروع پیاده سازی multicasting را برای اینکه بتوانند سرویس های بهتر و (تضمین شده) ارائه دهند، داشته باشند.

۲۴- معمولا، درخت های multicast در سطح Application، با توجه به کشیدگی [فاصله] که برحسب تاخیر یا تعداد hop ها اندازه گیری می شود، بهینه شده‌اند. مثالی بیاورید که در آن این معیار به درخت های خیلی ضعیفی بیانجامد.

اساس فرض کشیدگی این است که تاخیرات ارتباط بر بازده تاثیر اساسی و قاطع دارد. هرچند، به عنوان مثال در زمینه پخش ویدئو، این در دسترس بودن است که به شمار می آید. در آن صورت، ما تمایل خواهیم داشت که درختانی با حداکثر هزینه (اندازه گیری شده بر اساس پهنای باند) بسازیم.

۲۵- در موقع جستجوی فایل ها در یک سیستم Peer-to-peer ساخت نیافته، محدود کردن جستجو به نودهایی که فایل هایی مشابه خودتان دارند، می‌تواند کمک کنند.توضیح دهید که چگونه شایعه پراکنی (Gossiping) می‌تواند در پیدا کردن آن نودها کمک نماید.

ایده کار بسیار ساده است: اگر در طی مدت شایعه پراکنی، نودها اطلاعات عضویت را رد و بدل کنند، هر نود سرانجام تمام نودهای دیگر موجود در سیستم را شناسایی خواهد کرد. هر زمان که نود جدیدی را کشف می‌کند، می تواند با توجه به نزدیکی معنایی با آن نود، ارزیابی را انجام دهد. برای مثال با شمارش تعداد فایل های اشتراکی. سپس نودهایی که از لحاظ معنایی نزدیک هم هستند، برای جستجو از طریق پرس و جو ارائه می شوند.

فصل ۵- سیستم توزیعی

۱- مثالی بزنید که در آن، آدرس موجودیتی به نام E نیاز دارد به آدرس دیگری تبدیل گردد تا در حقیقت به E برسد.

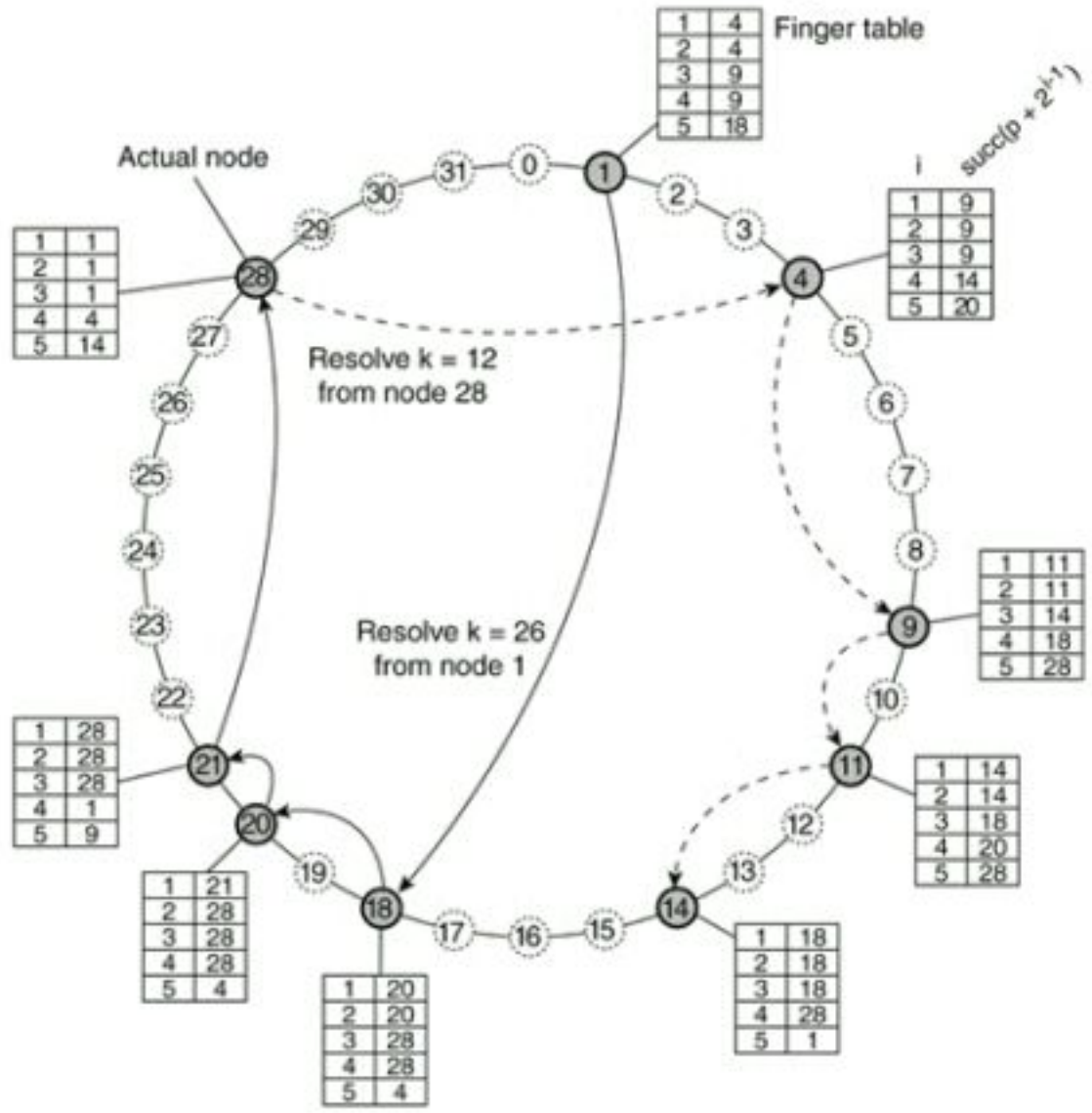
آدرس های IP در اینترنت برای آدرس دهی hostها استفاده می شوند. اگر چه برای دسترسی به یک host، آدرس IP اش باید ، به عنوان مثال، به یك آدرس اترنت {که اگر شبکه اترنت باشد یعني به MAC address } resolve شود.

۳- مثال هایی را از شناسه های واقعی ارائه کنید.
مثال ها عبارتند از : شماره های ISBN که در کتابها مورد استفاده قرار می گیرد- شماره های شناسایی برای محصولات نرم افزاری و سخت افزاری – شماره های کارمندی در یک سازمان – آدرس های اترنت (اگر چه بعضی از آدرس ها بجای اینکه فقط در بورد اترنت استفاده شوند، برای شناسایی یک ماشین مورد استفاده قرار می گیرند).

۴- آیا یک شناسه مجاز است حاوی اطلاعاتی راجع به موجودیتی که به آن اشاره می کند، باشد
بله – اما اطلاعات مجوز ندارند که تغییر کنند. زیرا این عمل، تغییر شناسه را می رساند. شناسه قدیمی می بایستی معتبر باقی بماند چون تغییر آن دلالت بر این دارد که یک موجودیت دو تا شناسه دارد. و این نقض ویژگی دوم شناسه ها می باشد.
[همانطور که می دانیم یک شناسه واقعی، نامی است که این ۳ خصوصیت را دارا باشد: ۱- یک شناسه حداکثر به یک موجودیت اشاره می کند. ۲ – به هر موجودیت حداکثر از طریق یک شناسه اشاره می گردد.(شناسه و موجودیت رابطه یک به یک دارند) ۳- یک شناسه همیشه به یک موجودیت اشاره می کند و عوض نمیشود (یعنی هرگز مجددا استفاده نمی گردد).]

6. **Q:** Consider the Chord system as shown in Fig. 5-4 and assume that node 7 has just joined the network. What would its finger table be and would there be any changes to other finger tables?

A: Let us first consider the finger table for node 7. Using the same method as we introduced when discussing Chord, it can be seen that this table will be [9, 9, 11, 18, 28]. For example, entry #2 is computed as $succ(7+21) = succ(9) = 9$. More tables will need to change, however, in



7. **Q:** Consider a Chord DHT-based system for which k bits of an m-bit identifier space have been reserved for assigning to superpeers. If identifiers are randomly assigned, how many superpeers can one expect to have in an N-node system?

A: The total number of superpeers in an overlay of N nodes will be equal to $\min\{2^{k-m}N, 2^k\}$.

10. **Q:** A special form of locating an entity is called anycasting, by which a service is identified by means of an IP address . Sending a request to an anycast address, returns a response from a server implementing the service identified by that anycast address. Outline the implementation of an anycast service based on the hierarchical location service described in Sec. 5.2.4.

A: Each service has a unique identifier associated with it. Any server implementing that service, inserts its network-level address into the location service at the directory node of the leaf domain in which the server resides. Lookup requests use the identifier of the service, and will automatically return the nearest server implementing that service.

۵- پیاده‌سازی یک globally unique Identifier را بطورکلی توضیح دهید.

چنین Identifier ای را میتوان بطور محلی و بصورت ذیل تولید کرد. آدرس شبکه ماشینی که Identifier در آن تولید می‌شود را گرفته و زمان محلی را به‌همراه یک عدد تصادفی ساختگی به آن می‌افزاییم. اگرچه در تئوری ، احتمال اینکه یک ماشین دیگر در نقاط دیگر دنیا نیز همین آدرس را تولید نماید وجود دارد اما می‌توان این شانس را نادیده انگاشت.

✓ ۸- اگر یک نود به سیستم Chord اضافه شود، آیا نیاز است که کل finger table بروزرسانی شود؟ ممکن است برای تعدادی لازم باشد.
خوشبختانه نه. سوال قبلی را در نظر بگیرید و تصور کنید که تنها finger table نود ۷ نصب شده است و بقیه همانطور که بوده اند باقی مانده اند. بدترین اتفاقی ممکن است بیافتد می تواند این باشد که یک درخواست که می‌بایست مورد جستجو قرارگیرد، فرض کنید کلید ۵، بجای نود ۷ به نود ۹ مسیردهی شود. درهرصورت نود ۹ هم می‌داند که نود ۷ به سیستم ملحق شده است و بنابراین اقدام اصلاحی را انجام می‌دهد.

✓ ۹- مهمترین اشکال جستجوی بازگشتی (Recursive) زمانی که آدرس یک کلید در یک سیستم DHT-based می‌بایست یافته شود، چیست؟
اشکالی که وجود دارد اینست که زمانی که یک Client درخواستی را ارسال می‌کند، اگر پاسخ آن بازگشت داده نشود، Client نمی‌داند که چه اتفاقی افتاده است. مثلاً درجایی در طول مسیری که برای یافتن آدرس کلید موردنظر می‌بایست طی شود، ممکن است پیغام گم شده و یا نودی فعال نباشد. بنا به این دلایل گاهی اوقات روش چرخشی (Iterative) ممکن است ترجیح داده شود. در این روش Client دقیقاً می‌داند که کدام قسمت جستجو دچار مشکل شده است و می‌تواند نود جایگزینی را برای حل مشکل انتخاب نماید.
مشکل دیگر روش بازگشتی این است که روی root بار زیادی وجود دارد ولی در حالت iterative چنین نیست .

✓ ۱۱- اگر تصور کنیم که روش two-tiered home-based ، حالت خاصی از روش سرویس‌های مکانی سلسله‌مراتبی است، دراینصورت نود ریشه درکجا قرار می‌گیرد؟
در این حالت نود ریشه از مجموع مکان‌های خانه هایی که به هم ملحق شده اند تشکیل می‌شود که البته بصورت جداجدا طوری ناحیه بندی شده اند که هر موجودیت متحرکی ، root server مختص به خود را داشته باشند.

✓ ۱۲ – با فرض به اینکه موجودیت سیار خاصی، هرگز از دامنه D خارج نمی شود؛ و اگر خارج شود انتظار می رود که فورا برگردد، چگونه می توان از این اطلاعات برای بالا بردن سرعت عملیات جستجو درسرویس hierarchical location استفاده کرد؟
جواب :
با کدگذاری کردن دامنه D در identifier موجودیتی که برای عملیات جستجو استفاده میشود ، به سادگی قابل انجام است . سپس عملیات جستجو میتواند سریعا به نود دایرکتوری dir(D) از جایی که جستجو ادامه پیدا می کند، هدایت شود.

✓ ۱۳- در سرویس hierarchical location با عمق K وقتی موجودیت سیار مکان خود را تغییر می دهد، حداکثر چند رکورد مکان (location record) باید update شود؟
جواب :
تغییر مکان می تواند بصورت ترکیب یک عملیات درج و یک عملیات حذف توصیف شود.

برای یک عملیات درج بدترین حالت K+1 رکورد باید تغییرکند و همچنین برای یک عملیات حذف نیاز به تغییر K+1 رکورد است . وقتی که رکورد در ریشه ، میان این دو عملیات مشترک است در نتیجه به 2K+1 رکورد منتهی می شود .

۱۴- موجودیتی را در نظر بگیرید که از مکان A به مکان B می رود و این در حالی است که از چندین مکان میانی عبور می کندو زمان نسبتاً کمی در هر مکان توقف دارد. وقتی به B می رسد مدتی می ماند. تغییر یک آدرس در سرویس hierarchical location ممکن است در زمان نسبتاً طولانی کامل شود و در نتیجه هنگام ملاقات مکان میانی از این تغییر اجتناب شود.موجودیت چگونه می تواند در مکان میانی ، مکان یابی شود؟

forwarding pointers

جواب :

سرویس hierarchical location را با اشاره گرهای هادی ترکیب میکنیم.هنگامی که موجودیت شروع به حرکت می کند، اشاره گر هادی A به دنبالش به مکان (میانی) بعدی حرکت میکند. هر زمان که موجودیت دوباره شروع به حرکت کند یک اشاره گر هادی به دنبالش حرکت میکند.به محض ورود B ، موجودیت ادرس جدیدش را داخل سرویس hierarchical location درج می کند.زنجیره اشاره گرها به طور پیوسته پاک می شود و ادرس در A حذف می گردد.

۱۵ – گره ریشه در سرویس hierarchical location می تواند به گلوگاه بالقوه تبدیل شود.چگونه می توان این مشکل را به طور موثری حل کرد؟

جواب :

نکته مهم این است که ما فقط از رشته بیتهای تصادفی به عنوان شناسه استفاده می کنیم در نتیجه می توانیم به آسانی فضای identifier را پارتیشن بندی کنیم و نود ریشه جداگانه برای هر پارتیشن قرار دهیم

بعلاوه باید نود ریشه پارتیشن بندی شده ، در میان شبکه انتشار داده شود تا دسترسی به آنها گسترش پیدا کند.

۱۶ – مثالی درباره اینکه چگونه مکانیزم closure برای یک URL کار میکند ارایه کنید.

فرض میکنیم که یک پراسس میداند که با یک URL در ارتباط است . ابتدا برنامه ی identifier را از URL استخراج میکند ، همچون: ftp . سپس در جدولی جستجو میکند تا یک interface مرتبط با پروتکل ftp پیدا کند (که به صورت محلی اجرا میشود). در قسمت بعد ی مکانیزم ، host name از URL استخراج میشود، همچون : www.cs.vu.nl و به dns سرور محلی داده میشود . دانستن محل و چگونگی تماس با سرور dns مهمترین بخش از مکانیزم closure است، که اغلب در داخل URL name resolver که پراسسی در حال اجراست به صورت خاص طراحی شده است . در نهایت آخرین بخش از URL که فایل مورد جستجوست ، به host پیدا شده تحویل داده میشود . این مورد اخر از مکانیزم closure محلی اش برای name resolution فایل استفاده میکند.

۱۷- تفاوت بین Hard Link و Soft Link در سیستمهای یونیکس چیست ؟ آیا مواردی وجود دارند که با Hard Link انجام شود ولی با Soft Link نتوان آن را انجام داد و یا برعکس ؟

جواب :
در Hard Link آدرس گره به صورت آدرس مطلق میباشد و برای یک گره ممکن است چندین آدرس مطلق موجود باشد (همانند گره n5 در شکل ۹-۵ کتاب)
در Soft Link بعد از مراجعه به یک گره، در داخل آن آدرس مطلق یک گره دیگر وجود دارد و میتوان به آن مراجعه کرد.
مراجعه با دیسک نیاز به آدرس مطلق دارد اما برای مراجعه به پارتیشنهای آن دیسک باید از Soft Link استفاده کنیم.

✓ ۱۸- server های سطح بالا در DNS یعنی Name server هایی که نودها را پیاده سازی می کنند در فضای نام DNS که نزدیک به ریشه اند ، معمولا از Name Resolution بصورت بازگشتی پشتیبانی نمی کنند. اگر پشتیبانی می نمودند آیا می توانستیم کارایی بالاتری را انتظار داشته باشیم؟
جواب:
خیر- زیرا Name server های سطح بالا، تشکیل دهنده لایه جهانی فضای نام DNS می باشند و انتظارمیرود که تغییرات در آن بخش از فضای نام اغلب اتفاق نیفتد. در نتیجه ، caching خیلی موثر خواهد بود و هزینه ارتباط کم خواهد شد. بخاطر داشته باشید که name resolution بصورت بازگشتی برای Name server های دو سطحی مهم می باشد زیرا در آن صورت name resolution در domain سطح پایین تری که resolution در حال انجام است بصورت محلی در نظر گرفته می شود.
[در کل Name Resolution بصورت بازگشتی روی Root name server فشار زیادی را ایجاد خواهد کرد ، پس در لایه های بالاتر از iterative و در لایه های پایینتر از recursive استفاده میشود.]

✓ **19. Q:** Explain how DNS can be used to implement a home-based approach to locating mobile hosts.

A: The DNS name of a mobile host would be used as (rather poor) identifier for that host. Each time the name is resolved, it should return the current IP address of the host. This implies that the DNS server responsible for providing that IP address will act as the host's name server. Each time the host moves, it contacts this home server and provides it with its current address. Note that a mechanism should be available to avoid caching of the address. In other words, other name servers should be told *not* to cache the address found.

۲۰- Mounting Point در اغلب سیستمهای یونیکس چگونه جستجو میشود ؟
جواب:
بوسیله یک mount table که شامل یک entity point به mount point می باشد.
بدان معنی که، زمانی که یک mounting point جستجو می شود، ما باید کل mount table را مرور کنیم تا ببینیم کدام entity با mount point داده شده مطابقت دارد.

۲۱- یک فایل سیستم توزیع شده را در نظر بگیرید که از فضاهای per- user name (نام به ازای هر کاربر) استفاده میکند. یعنی هر کاربر دارای فضای نام خصوصی اش است. آیا نامهایی از این فضاهای نام میتواند برای اشتراک منبع بین دو کاربر مختلف استفاده شود ؟

جواب:
بله، provided names درفضاهای per- user name می توانند با name هایی که در فضای global name به اشتراک گذاشته شده اند، resolve شوند. بعنوان مثال ۲ identical name در name space های مختلف، کاملا مستقل از هم می باشند و ممکن است به entityهای متفاوتی اشاره کنند. برای اشتراک entityها لازم است به آنها توسط name های از یک shared name space اشاره شود.
برای مثال Jade به DNS names و IP addresses ، rely می کند [اعتماد می کند!] که می تواند برای refer کردن به shared entity هایی مانند سایتهای FTP، استفاده شود.

✓ **22.** **Q:** Consider DNS. To refer to a node *N* in a subdomain implemented as a different zone than the current domain, a name server for that zone needs to be specified. Is it always necessary to include a resource record for that server's address, or is it sometimes sufficient to provide only its domain name?

A: When the name server is represented by a node *NS* in a domain other than the one in which *N* is contained, it is enough to give only its domain name. In that case, the name can be looked up by a separate DNS query. This is not possible when *NS* lies in the same subdomain as *N*, for in that case, you would need to contact the name server to find out its address.

۲۳- شمارش فایلهای مشترک، روش ساده ای برای تعریف همجواری معنایی است. فرض کنید که میخواهید یک شبکه همپوشانی معنایی (Semantic Overlay) را بر اساس اسناد متن کتاب بسازید، چه تابع همجواری منطقی (semantic proximity function) دیگری را میتوان در نظر گرفت ؟
جواب:
یک روش (یک توطئه - روش فریبکارانه One intriguing one is) این است که در زمان ممکن، نگاهی به actual content بیاندازیم. در خصوص document ها می توانیم به function های مشابه که از اطلاعات بازیابی بدست آمده نگاهی کنیم. مانند Vector Space Model (VSM).

فصل ۶- سیستم توزیعی

2. **Q:** Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what is the maximum clock skew that will occur?

A: The second clock ticks 990,000 times per second, giving an error of 10 msec per second. In a minute this error has grown to 600 msec. Another way of looking at it is that the second clock is one percent slow, so after a minute it is off by 0.01×60 sec, or 600 msec.

5. **Q:** Add a new message to Fig. 6-0 that is concurrent with message A, that is, it neither happens before A nor happens after A.

A: The solution cannot involve 0 or it would be ordered. Thus it must be a message from 1 to 2 or from 2 to 1. If it departs or arrives from 1 after 16, it will be ordered with respect to A, so it must depart or arrive before 16. The possibilities are a message leaving process 2 at 0 and arriving at process 1 at 8, or a message leaving process 1 at 0 and arriving at process 2 at 10. Both of these are concurrent with A.

9. **Q:** In the centralized approach to mutual exclusion (Fig. 6-0), upon receiving a message from a process releasing its exclusive access to the resources it was using, the coordinator normally grants permission to the first process on the queue. Give another possible algorithm for the coordinator.

A: Requests could be associated with priority levels, depending on their importance. The coordinator could then grant the highest priority request first.

الگوریتم توزیعی

11. **Q:** Ricart and Agrawala's algorithm has the problem that if a process has crashed and does not reply to a request from another process to access a resources, the lack of response will be interpreted as denial of permission. We suggested that all requests be answered immediately to make it easy to detect crashed processes. Are there any circumstances where even this method is insufficient? Discuss.

A: Suppose that a process denies permission and then crashes. The requesting process thinks that it is alive, but permission will never come. One way out is to have the requester not actually block, but rather go to sleep for a fixed period of time, after which it polls all processes that have denied permission to see if they are still running.

۱- حداقل ۳ منبع تاخیر را نام ببرید که می توانند بین wwv پخش کننده ی زمان و پردازنده ها در سیستمهای توزیع شده ای که ساعت های داخلی آن ها را تنظیم می کنند وجود داشته باشد.

جواب :

۱- تاخیر در انتقال سیگنال در اتمسفر ۲- تاخیر در انتقال بسته ها در شبکه lan

۳- تاخیر در خود پروسسور ها پس از دریافت بسته ها به دلیل وقفه های داخل پروسسور یا صف های داخلی

۳- یکی از سرویس های مدرنی که در سیستم های توزیع شده معرفی شدند گیرنده های GPS است. مثال هایی از کاربرد های توزیع شده را ارائه دهید که می تواند از اطلاعات GPS استفاده کند.

جواب:

سیستم های health care & sport که در هنگامی که فرد در بیرون از خانه مشغول فعالیت های ورزشی است ضربان قلب و علائم دیگر او را اندازه گیری می کند. میتوان این دستگاه را به کامپیوتری وصل کرد تا اطلاعات را برای آنالیز های بعدی ثبت کند. همچنین سیستم های راهنمای خودرو که می توانند به یک کامپیوتر وصل باشند که مرتب وضعیت ترافیک و نقشه های راهنما را از اینترنت بگیرد.

۴- *چرا ایده استفاده از مقادیر ساعت اندازه گیري شده قبلي موقع تنظيم ساعت با يك ند ديگر مفيد است؟مثالي از چگونگي به حساب آوردن ساعت هاي گذشته بزنيد.*

جواب-يك پاسخ واضح امكان اشتباه در خواندن زمان جاري است.فرض كنيد لازم است ساعتها در نهايت و رفته رفته تنظيم شوند ، يك امكان در نظر گرفتن آخرين N مقدار و محاسبه ميانه يا ميانگين آنها است . اگر عدد محاسبه شده خارج از بازه زماني جاري بود، به حساب نمي آيد . (اما در ليست اضافه مي شود) همچنين مقدار جديد مي تواند توسط ميانگين وزني يا يك الگوريتم ديگر محاسبه شود.

۶-برای دستیابی به چند پخشی مرتب کامل با برچسبهاي زماني لامپورت ،آیا لازم است که هر پیام اعلام وصول شود؟

خیر .زیرا شرط رسیدن هر پیام به گیرنده آن است که تمام پیام ها از فرآیندهاي های دیگر با برچسب زماني بزرگتر دریافت شده باشند. این روش تضمین میکند تمام پیام ها با برچسب زماني کوچکتر قبلا رسیده اند.

✓ ۷-یک لایه ارتباطي را در نظر بگیریدکه در آن پیام ها فقط به ترتیب ارسال ،تحویل داده میشوند .مثالی ارائه نمایید که در آن این مرتب سازی نیز بطور غیر ضروري محدود کننده است

مثلا فرستادن عکس با حجم بالا. در این صورت عکس باید به قسمت های مختلف تقسیم شود . مسلما هر بخش باید طول و عرض و همچنین موقعیت قرار گیری خود را نسبت به عکس اصلی به همراه داشته باشد . در این حالت دیگر به صف FIFO احتیاجی نیست زیرا گیرنده بسادگي میتواند هر قسمت را در موقعیت مناسب خود قرار دهد.

✓ ۸-بسیاری از الگوریتم های توزیع شده نیاز به استفاده از یک فرآیند هماهنگ کننده دارند . تا چه اندازه میتوان چنین الگوریتم هایی را واقعا توزیع شده در نظر گرفت؟

جواب: در يک الگوريتم مرکزي ، اغلب يک پروسس مشخص وجود داشته که بعنوان هماهنگ کننده عمل مينمايد . توزيع شدگي ناشي از اين واقعيت است که پروسسها ، در ماشينهاي مختلف اجرا ميشوند. در الگوريتم های توزيع شده با هماهنگ کننده غير ثابت ، هماهنگ کننده از ميان پروسسهاي تشکيل دهنده الگوريتم انتخاب ميشوند.اين حقيقت که يک هماهنگ کننده وجود دارد ، الگوريتم را کمتر توزيع شده نمي نمايد.

۱۵-شکل ١٤-٦ را در نظر بگيريد. فرض کنيد که هماهنگ کننده از کار بيفتد . آيا اين اتفاق هميشه باعث از کار افتادن کل سيستم ميشود؟ اگر نه، تحت چه موقعيتي اين امر رخ می دهد؟ آيا هيچ راهی برای اجتناب از اين مشکل و ايجاد سيستمي که قادر به تحمل از کارافتادن هماهنگ کننده باشد وجود دارد؟

جواب-فرض کنيد الگوريتم به صورتي است که بلافاصله هر درخواست را با اجازه دسترسي و يا عدم اجازه دسترسي پاسخ دهد.

اگر هيچ فرآيندي در حال استفاده از منابع نبوده و همچنين در صف انتظار براي منابع نيز نباشد ، در اينصورت خرابي هماهنگ کننده باعث از کار افتادن کل سيستم نميشود. هر درخواست پروسس بعدي براي دسترسي به منابع بي پاسخ مانده و در نتيجه فرايند انتخاب هماهنگ کننده جديد به جريان مي افتد. اين سيستم می تواند با ذخيره هر درخواست رسيده و قبل از ارسال پاسخ به ان حتي مقاوم تر نيز ساخته شود. در اين شيوه موقع خرابي ، هماهنگ کننده جديد می تواند ليست دسترسي به منابع و همچنين صف در خواستها را با خواندن فايل بازسازي نمايد.

۱۲ - اگر فرض کنيم که بتوان الگوريتمهاي جدول١٧-٦ را در در يك شبکه محلي با قابليت همه پخشي پياده سازي نماييم ،عناصر جدول گونه تغيير ميکنند.

تنها موارد مربوط به حالت توزيع شده در جدول تغيير ميکنند. چون ارسال يك پيام بصورت ند-به-ند هزينه اي برابر ارسال يك پيام فراگير(broadcast) دارد ، تنها لازم به ارسال يك پيام براي تمام ندهاي درخواست کننده منبع ميباشد. بهمين دليل ارسال تنها يك پيام فراگير موقع ازاد ساري منبع کفايت ميکند. بنابراين تاخير مي شود ١ + (n - 1): يکی از تاخيرها مربوط به درخواستbroadcast است و n-1 تاخير نيز مربوط به دريافت پيام از پروسسهاي ديگر که قبلا به انها اجازه دسترسي به منابع داه شده است .

۱۳- يک سيستم توزيعي ممکن منابع مستقل متعددي داشته باشند. تصور است که فرايند ۰ ميخواهد به منبع A و فرايند ١ ميخواهد به منبع B دسترسي داشته باشد. آيا الگوريتم هاي Ricartو Agrawala از بن بست جلوگيري ميکند؟ پاسخ خود را توضيح دهيد.

جواب- بستگي به قوانين در محيط سيستم دارد. اگر دسترسي به منابع بصورت ترتيبي موکد بوده بطوريکه که يك پروسس داراي منبع نتواند براي دسترسي به منبع جديد ، در حالي پروسسهاي ديگر منتظر منابع او باشند تلاش کند ، سپس امکان انتظار براي دريافت منبع جديد وجود نداشته و سيستم فاقد بن بست (DeadLock) خواهد بود.

از طرف ديگر اگر پروسس ۰ منبع A را در اختيار بگيرد و براي دسترسی به منبع B تلاش کند و در همان زمان پروسس ديگري در جهت عکس رفتار نمايد امکان رخداد بن بست (DeadLock) وجود دارد .الگوريتم Ricart and Agrawala نميتوانند به خودی خود کمک کند زيرا هر منبع به طور مستقل اداره مي شود .

۱۴ - سوال: تصور کنيد دو فرايند همزمان متوجه مرگ يک coordinator شوندو هر دو تصميم به در دست گرفتن يک انتخابات (Election) با استفاده از الگوريتم bully بنمايند.چه اتفاقی خواهد افتاد؟

پاسخ : هر کدام از فرایند های higher-numbered دو پیام Election (انتخابات)در یافت خواهند کرد، ولی از دومی صرف نظر می کنند. انتخابات (Election) به طور معمول انجام می شود.

۱۵-سوال : در شکل ۲۱-۶ ما دو پیام انتخابات (Election message) همزمان چرخشی داریم. تا زمانی که داشتن دوتای آنها آسیبی نمی زند ، برازنده تر است که اگر یکی از آنها بتواند از بین برود. الگوریتمی برای انجام این کار بدون تاثیر گذاشتن بر عملکرد پایه الگوریتم انتخابات طراحی کنید.

پاسخ : هنگامی که یک فرایند یک Election message دریافت می کند ،چك میکند که چه کسی آن را آغاز کرده است. اگر خودش شروع کننده باشد(توضیح: شماره اش در سر لیست است)، پیام را به یک Coordinator message تغییر می دهد، همانطور که در متن گفته شد.اگر هیچ Election message را شروع نکرده بود، شماره پروسسش را اضافه می کند و آن را در طول حلقه forward می کند. اما اگر فرایند Election message خود را پیشتر فرستاده بود و فقط یک رقیب را کشف کرد ، شماره صادر کننده پیام را با خودش مقایسه میکند. اگر فرایند شماره کمتری داشت ، به جای پاس دادن پیام انرا دور می ریزد.اگر رقیب بالاتر باشد ، پیام به روش معمول forward می شود.در این صورت ، اگر چندین Election message شروع شده باشند ، ان که اول از همه ثبت شود بیشتر باقی می ماند. سایرین در طول کانال از بین می روند.

فصل 7- سیستم توزیعی

سوال۱ : دسترسی به object های مشترک جاوا میتواند به وسیله شناسایی متد هایشان پیاپی شود تا همزمان شوند.آیا این برای تضمین پی در پی پذیری هنگامی که چنین شیئی replicate می شود کافیست؟

پاسخ : خیر. مساله این است که دسترسی به هر نسخه serialized است . اما عملیات متفاوت روی نسخه های متفاوت که ممکن است در یک زمان انجام شود، متغیر های لحظه ای چند نسخه سازی را در یک وضعیت inconsistent قرار می دهد.

سوال ۲: دليل اصلي در نظر گرفتن مدلهاي ضعيف سازگاري را توضيح دهيد

پاسخ: مدل های ضعیف سازگاري(consistency) براي نسخه هاي تكراري (Replicate) وجهت رسيدن به كارايی لازم ميباشند.با وجود اين نسخه هاي تكرار (Replicate) كارا را فقط ميتوان با صرفه نظر كردن از همسان سازي سراسري و از طريق چشم پوشي از محدوديتهاي آن بدست آورد.

سوال۳ :شرح دهید که چگونه چند نسخه سازی در DNS رخ می دهد ، و چرا عملا اینقدر خوب کار می کند؟

پاسخ : ايده اصلی اين است كه سرورهاي نام نتايج جستجوي قبلي را در حافظه نهان(Cache) ذخيره مي كنند. این نتایج می توانند در حافظه نهان (Cache) برای زمان طولانی نگهداری شوند، زیرا DNS فرض می کند که map کردن نام-به-آدرس معمولا تغییر نمی کند.

بنا به گفته یکی از دوستان استاد تا آخر صفحه ۲۹۶ کتاب تدریس کردند و سوالاتي که با رنگ آبي مشخص شده تدریس نشده است و مورد امتحان نمي باشد.

٤- اگر یک برنامه یک ذخیره دیتای consistent را انتظار داشت و با چیز دیگری نتوانست ادامه دهد ،انبار داده باید sequential consistency را تامین نماید.همچنین ضروری است که نرم افزار با قوانین القا شده توسط این مدل سازگار گردد.به طور کلی این به این معنا می باشد که برنامه هایی که از این قوانین پیروی می کنند با ذخیره داده persistent مواجه خواهند بود.
به طور کلي به این معناست که برنامه هایي که این قوانین را پذیرفته اند چیزي شبیه sequential consistency را درك کنند.

٦ – در شکل ۷-۷ ، آیا مقدار ۰۰۱۱۱۰ برای حافظه با سازگاری دنباله ای (sequentially consistent) ، خروجی معتبری است؟ شرح دهید.

پاسخ :بلی ، اگر پروسس ها به ترتیب (a) ، (c) ، (b) اجرا شوند این نتیجه بدست می آید.

۷- معمولا بحث می شود که مدل های Weak Consistency بار زیادی را برای برنامه نویسان فراهم می کند. این جمله تا چه حد درست است؟

پاسخ : این واقعا بستگی دارد.بسیاری از برنامه نویسان داده های مشترکشان (shared data) را از طریق مکانیزم های قفل گزاری (locks) و تراکنش ها محافظت می کنند. ایده اصلی این است که آنها نیازمند حالت شدیدتری از همزمانی نسبت به سطحی در دستورالعملیات read و write وجود دارد می باشند. هرچند که ، برنامه نویسان انتظار می رود عملیاتی در متغیر های متقارن سازی انجام دهند که منجر به سازگاری دنباله ای (sequentially consistency) شود.
به هر حال که برنامه نویسان انتظار دارند sequential consistency با انجام عملیات روي متغیر هاي همگام سازي همچنان باقي بماند.

سوال ۸: آیا پخش فراگیر مرتب (totally ordered multicasting) بوسیله یك مرتب کنند ه و براي سازگاري نسخه هاي تکراري فعال (active replication) منطق انتها – به- انتها را در سیستم هاي توزیعي نفي نمیکند
جواب ۸:بله. مفهوم انتها-به –انتها مي رساند که مسائل باید در سطح رخداد حل شوند.در پروتکلهاي پایه اي اصلي ، سازگاري بوسیله ارسال همه دستورات به نسخه اولیه بدست مي آمد.استفاده از یك ترتیب دهنده (sequencer) همان عمل را در سطح پایین تري از تجرد انجام میدهد.در این وضعیت ممکن است بهتر باشد تا از پروتکل پایه اي اصلي براي انتشار تغییرات بوسیله ارسال عملیات استفاده شود.

۹ – برای پیاده سازی بازار بورس الکترونیکی از کدام consistency استفاده خواهید کرد؟ شرح دهید.

پاسخ : سازگاری علیتی (casual consistency) ممکن است کافی باشد.دلیل آن این است که واکنش به تغییرات در متغیر های بورس باید consistent باشد.تغییرات در بورسهایی که مستقل هستند می تواند به ترتیب های مختلف دیده شوند.

تمرین ۱۰:تمام ان، باید توجه داشت که صاحب ان باید همیشه آن mailbox را ببیند(منظور این است که همیشه همه ایمیل ها را در mail box خود ببیند) بدون توجه به اینکه او آن را می خواند و یا بروز رسانی می کند.در حقیقت ساده ترین پیاده سازی برای چنین mailbox ای primary-based local –write protocol می باشد به گونه ای که primary در کامپیوتر کاربر سیار قرار گرفته باشد.

۱۱- پیاده سازی ساده ای از read-your-write Consistency را برای نمایش صفحات وبی که به هنگام شده اند ، شرح دهید.

پاسخ : ساده ترین پیاده سازی این است که به مرورگرمان اجازه دهیم آخرین ورژن های صفحات وب را نمایش دهد. این ملزم به ارسال درخواست به web server است.این شمای ساده در بسیاری از سیستم ها قبلا پیاده سازی شده است.

12. **Q:** To make matters simple, we assumed that there were no write-write conflicts in Bayou. Of course, this is an unrealistic assumption. Explain how conflicts may happen.

پاسخ: شرايط زيادي هستند كه ممكن است تداخل write-writeاتفاق بيفتد. مثلا كلاينت مي تواند يك برنامه به اشتراك گذاشته شده را update كند. Update مي تواند زمانبندي يك جلسه در يك بازه زماني باشد كه كلاينت ان بازه را قبلا براي جلسه يا برنامه ديگري زمانبندي كرده است اما اطلاعات زمانبندي كلاينت هنوز به ساير كپي ها منتشر نشده است.

13. **Q:** When using a lease, is it necessary that the clocks of a client and the server, respectively, are tightly synchronized?

پاسخ: خير- اگر كلاينت يك ديد بدبينانه (pessimistic view) نسبت به ميزان همگام بودن ساعتش با ساعت سرور داشته باشد تلاش مي كند قبل از اينكه lease فعلي اش منقضي شود يك lease جديد را بدست آورد.

14. **Q:** We have stated that totally ordered multicasting using Lamport's logical clocks does not scale. Explain why.

پاسخ: در روش totally ordered multicasting ارائه شده توسط lamport لازم است كه تمامي سرور ها فعال و درحال اجرا باشند(up and running) كه در صورت كند شدن و يا خرابي يكي از سرور ها، performance به شدت متاثر خواهد شد و در اين صورت بايد همه سرور ها از اين موضوع مطلع شوند كه با افزايش تعداد سرورها مشكل حادتر خواهد شد.

15. **Q:** Show that, in the case of continuous consistency, having a server S_k advance its view $TW_k(i,k)$ whenever it receives a fresh update that would increase $TW(k,k) - TW_k(i,k)$ beyond $\delta_i / N - 1)$, ensures that $v(t) - v_i \leq \delta_i$.

پاسخ: متاسفانه این بخش را نفهمیدم به محض مطالعه پاسخ خواهم داد.

تمرین ۱۶: س: برای Consistency پیوسته و مداوم، ما فرض کردیم هر بار نوشتن، مقدار عنصر داده ای x را فقط افزایش می دهد. راه حلی را به طور خلاصه ارائه دهید که در آن ، کاهش مقدار x نیز امکان پذیر باشد.
ج: این وضعیت نسبتا ساده خواهد بود اگر، ما مقادیر بروز رسانی های مثبت را از منفی ها جدا کنیم و حساب جداگانه ای برای هر کدام از آنها داشته باشیم. به طور خاص، مورد زیر را دنبال می کنیم:

$$TWN[i, j] = \Sigma \{weight(W)\ weight(W) < 0 \& origin(W) = Sj \ \&\ W \in Li\}$$
$$TWP[i, j] = \Sigma \{weight(W)\ weight(W) > 0 \& origin(W) = Sj \ \&\ W \in Li\}$$

نکته آنکه $TWP[i, j] \equiv TW[i, j]$. دوباره، هر گره، رد نما های (View) $TWPk[i, k]$و $TWNk[i, k]$ را متعاقبا نگه می دارد و زمانی که متوجه شود که یک نوشتن تازه، یکی از مقادیر $|TWN[k, k] - TWNk[i, k]|$ و یا $|TWP[k, k] - TWPk[i, k]|$ را بیش از $d\ i\ /\ (N - 1)$ افزایش دهد، نمای خود را گسترش می دهد.

تمرین ۱۷-
س: فرض کنید که یک پروتکل Nonblocking primary-backup برای تضمین consistency متوالی در یک سیستم توزیع شدهٔ ذخیرهٔ داده، بکار رود؛ آیا سیستم ذخیره سازی مذکور، همیشه Consistency از نوع read-your-writes را فراهم می سازد؟
ج: نه خیر. به محض اینکه پراسس بروز رسانی یک پیام تصدیق (Acknowledgement) مبنی بر پردازش بروز رسانی خودش دریافت کند، ممکن است از سیستم ذخیرهٔ داده قطع ارتباط کرده و به یک نسخهٔ کپی دیگر متصل شود. هیچ تضمینی وجود ندارد که آین بروز رسانی به نسخهٔ کپی مذبور نیز رسیده باشد. در مقابل، با یک پروتکل از نوع Blocking، پروسس بروز رسانی، فقط پس از انتشار کامل بروز رسانی اش به تمامی نسخه های کپی دیگر، می تواند ارتباط خود را قطع کند.

۱۸-س: برای اینکه تکثیر فعال (Active replication) در حالت کلی ، درست کار کند، لازم است که تمامی اعمال بر روی نسخه های کپی دیگر نیز با یک ترتیب انجام شوند. آیا این ترتیب انجام، همیشه لازم است؟
ج: خیر. اعمال خواندن را در حالتی در نظر بگیرید که بر روی داد ه های تغییر نیافته و یا اعمال نوشتن جابجایی پذیر انجام می شوند. اساسا چنین شرایطی، اجازهٔ اختلاف ترتیب عملیات در نسخ کپی مختلف را می دهد. اگر چه تشخیص اینکه ، به طور مثال، دو عمل نوشتن جابجایی پذیر(Commutative) هشتند یا خیر، دشوار و یا حتی در برخی موارد ، غیر ممکن است.

19 – برای پیاده سازی چند پخشی کاملا مرتب بوسیله دنباله ساز (sequencer) ، یک روش این است که ابتدا عملیات به sequencer ارسال شود تا شماره یکتایی به آن نسبت داده شود و سپس این عملیات چندپخشی شود.دو روش دیگر پیشنهاد کنید و سه راه حل را با هم مقایسه کنید.
پاسخ : رهیافت دیگر چندپخشی کردن عملیات است ، اما تحویل آن تا زمانی که sequencer متعاقبا یک شماره ترتیبی (sequence number) را برای آن چندپخش کند به تعویق می افتد

و چندپخشی عملیات بعد از اینکه عملیات (operation) توسط sequencer دریافت شد انجام می شود. رهیافت سوم اول گرفتن sequence number از sequencer است و سپس عملیات را چندپخشی می کند.

رهیافت اول (ارسال عملیات به sequencer) ، درگیر ارسال یک پیام نقطه به نقطه حاوی عملیات می شود، و بعد پیام را چند پخشی می کند.رهیافت دوم دو پیام چند پخشی را نیاز دارد ، یکی حاوی عملیات و دیگری حاوی یک sequence number . رهیافت سوم ، در نهایت ، یک پیام نقطه به نقطه حاوی یک sequence number را هزینه دارد که به دنبال آن یک پیام چندپخشی حاوی عملیات است.

20. Q: A .le is replicated on 10 servers. List all the combinations of read quorum and write quorum that are permitted by the voting algorithm.
A: The following possibilities of (read quorum, write quorum) are legal.
(1,10), (2, 9), (3, 8), (4, 7), (5, 6), (6, 5), (7, 4), (8, 3), (9, 2), and (10, 1).

21. Q: State-based leases are used to offload a server by letting it allow to keep track of as few clients as needed. Will this approach necessarily lead to better performance?

پاسخ: خير- همانطور كه براي برخي كلاينت ها همچنان بهتر است كه در زمان وقوع update به آنها اطلاع رساني شود براي سرور ها هم به همين شكل است و نگهداري هر وضعيتي منجر نمي شود كه اين كلاينت ها هميشه به سروري كه در حال حاضر busy است راي دهند.

۲۰: فایلی بر روی ۱۰ سرور کپی سازی (Replicate) شده است. ترکیب تمامی Read quorum و Write quorum هایی که توسط الگوریتم رای گیری (Voting Algorithm) اجازه داده می شوند را فهرست کنید.

ج: احتمالات زیر برای (read quorum,Write quorum) مجاز می باشند:

(1,10), (2,9),(3,8),(4,7),(5,6),(6,5),(7,4),(8,3),(9,2),(10,1)
