Northeastern University – Silicon Valley

CS 6650 Scalable Dist Systems

Homework #5 [100 points]

Name: Nisarg Patel
Email: patel.nisargs@northeastern.edu

***INSTRUCTIONS:* Please provide clear explanations in your own sentences, directly answering the question, demonstrating your understanding of the question and its solution, in depth, with sufficient detail. Submit your solutions [PDF preferred]. Include your full name. Do not email the solutions.**

1. Security Engineering Ch. 11 please answer

11.2 [5 points]

Ans:

A conventional email communication includes a sender, a receiver and a channel over which email is sent. The conventional email is vulnerable to different ways in which the channel is being misused:

1) Eavesdropping: The email is not encrypted before sending. It can be protected by using an encryption algorithm, thus encrypting the email during sending and decrypting during receiving.

2) Masquerading: The sender(or receiver) authentication is missing. It can be protected by digitally signing the email before sending it.

3) Message Tempering: A secure connection is not established for the channel, thus resulting in man-in-the-middle attacks. The proof of secure channel can be established by both sender and receiver sending a digitally signed and encrypted message through the channel before sending the email.

4) Replaying: This type of attack can happen with secure channel if the email message is not timestamped. To protect from this, timestamps can be added to the email message.

5) Denial of Service: It can happen if the channel is open for multiple connections, and one of the senders tries to block the channel. It can be prevented by putting a limit on number of emails that can be sent by a sender in a time period in the channel.

11.3 [10 points]

Ans:

Some of the defences against man-in-the-middle attacks during initial exchanges of public keys are:

1) A third-party key-distribution service can be used to get a public key certificate. This certificate can than be entrusted with security and public keys of both parties can be read from this certificate after ensuring the validity of the certificate.
2) A secondary secure channel like TLS can be established that shares only the public key. The further communication can be encrypted and sent on insecure channels.
3) Another way of sharing public keys can be by using CD-ROM or other hardware devices that stores public keys and certificates which are accessible to only the corresponding parties.
4) The service's domain name can be included in it's public-key certificates so that clients can only acknowledge the service from the corresponding IP address only.

5) A non-digital way of communication can be used for initial exchanges of public keys. Some of the methods include verbal communication of the key, or sharing keys handwritten by both parties.

11.9 (please ignore the last sentence – parallel processor enhancement part) [10 points]

Ans:

We are assuming the values from figure 11.13, Crypto++ 2.1 GHz Pentium 4 for finding the encryption times.

56-bit DES:

Encryption time per key:

Performance = 21.340 $Mbytes/s$

Thus, time to encrypt 8 bytes = 8/(21.340 × 10^6) = 3.75 × $10^{-7} sec$

Inner loop time per key:

Instructions per key = 10 /s

Computer speed = 2000 Million Instructions / s

Thus, time to compute inner loop per key = 10 / (2000 x 10^6) = 10/(2 x 10^9) = 5 x 10^-9 sec

Average Time required to crack:

Key length for DES = 56

Total possible keys = 2^56

Average keys to process before cracking = (2^56)/2 = 2^55

Total time per key = 3.75 x 10^-7 + 5 x 10^-9 = 3.8 x 10^-7 sec

Thus, average time required to crack = 2^55 x (3.8x 10^-7) = 1.37 x 10^10 sec ≈ 434 years

It will take approx. **434 years** to crack the 56 bit DES key using 2000 MIPS

128-bit IDEA:

Encryption time per key:

Performance = 18.963 Mbytes/s

Thus, time to encrypt 8 bytes = 8/(18.963 x 10^6) = 4.22 x 10^-7 sec

Inner loop time per key:

Instructions per key = 10 /s

Computer speed = 2000 Million Instructions / s

Thus, time to compute inner loop per key = 10 / (2000 x 10^6) = 10/(2 x 10^9) = 5 x 10^-9 sec

Average Time required to crack:

Key length for DES = 128

Total possible keys = 2^128

Average keys to process before cracking = (2^128)/2 = 2^127

Total time per key = 4.22 x 10^-7 + 5 x 10^-9 = 4.27 x 10^-7 sec

Thus, average time required to crack = 2^127 x (4.27 x 10^-7) = 7.26 x 10^31 sec ≈ 2.3 x 10^24 years

It will take approx. $\mathbf{2.3 \times 10^{24}}$ **years** to crack the 128-bit IDEA key using 2000 MIPS.

2. Multimedia Apps Ch. 20 please answer

20.2 [5 points]

Ans:

Even though Internet does not currently provide quality of service or resource reservation, the multimedia applications are deployed based on the following categories:

1) Web-based multimedia: These types of applications are used to provide a high quality audio and video data streams that are published through Web. For these applications, synchronization of data streams is not of importance. Even without resource scheduling, they provide a feasible multimedia playback. They use the extensive buffering at the personal computers to smooth the variation of bandwidth and latency.
2) Video-on-demand services: These provide video information by retrieving data from a remote storage and provide user with the video. A high bandwidth and dedicated servers are required to provide high quality. These also use the extensive buffering solution.

   Extensive buffering can be carried out by traffic shaping. Using a buffer helps to regulate the burst of the data in multimedia streams. It uses the leaky bucket algorithm to eliminate such bursts. A limitation of this would be the delay of the initializing the buffer would be higher, which could lead to multiple seconds of initial delay.

3) Highly interactive applications: These types of applications require cooperation, synchronization, and coordination among multiple users. These include internet telephony, and video conferencing apps. These require that the delay should be greater than 100-300 ms to achieve synchronization. Since quality of service is not available in the internet, these types of applications adjust the quality of media presentation. One way would be to drop some packets to ensure coordination. Other way is to use scaling of different types. Scaling in video streams include temporal scaling(reducing the resolution), spatial scaling(reducing the number of pixels per image), frequency scaling, amplitude scaling and color space scaling. The drawback of such methods would be that the quality of media is traded to ensure interactivity.

20.3 [10 points]

Ans:

Synchronous Distributed State: In this form of synchronization, the state of the application should be same for all the users in the session. If one user performs an action, it should be affected to all the users. In a video conferencing, if a user is presenting any screen, then a view dedicated to that screen should come up at every user. Similarly, if a user stop presenting, then the view should be removed from every user. Thus, the state of screen sharing should be same on all users. This can be achieved by sharing state information along with the data streams to update the current state at every users.

Media Synchronization: All the users in the session should have the audio and video performances of similar times(within 50 ms). In the case of video conferencing, all the users should hear the video and audio at the same time to achieve synchronization. A user if presents, then the audio or video

streams should reach the participants at similar times. This can be achieved by providing a timestamp to the data streams and playing the same timestamp audio and video together.

External Synchronization: This involves synchronization of other forms of data available to the application session like CAD data and shared documents. Updates to such external data should be visible to all users in the session at the similar times. For video conferencing, an example would be a whiteboard shared among all users, with some having permissions to update and some having permissions to just view. All the updates can happen concurrently and can be view as if it is approximately synchronized. This requires both, application state sharing and timestamping the state to achieve coordination.

20.4 [10 points]

Ans:

A QoS Interface can be defined in java as:

```
public interface QoSManager {
…
}
```

A QoS Manager has two main subtasks:

1) Quality of service negotiation: An application would request resource requirements to the QoS Manager. QoS Manager then looks at the available resources and negotiates with the application if not enough resources are available. It then returns a ResourceContract, stating the reserved resources and time limit, if the resources are available in the first place or after negotiation with application. This can be fulfilled with the application calling the following method:

   ResourceContract Negotiate(QoSRequirements requirements);

   Here, QoSRequirements is an object that contains the information about the requirements including Bandwidth, Latency and Loss Rate and resources of the components required. Negotiate would have to call ApplicationNegotiate(QoSRequirements possibleRequirements) if the original requirements cannot be fulfilled.

2) Admission Control: After receiving the ResourceContract, the application can run normally. The application would then have to nofity the QoSManager if there are any changes in the ResourceRequirements. Application can then call:

   ResourceContract NotifyRequirementChanged(QoSRequirements newRequirements);

   This method which check the requirements again. If it is decreased, then additional resources are released and a new ResourceContract is sent back to the application. If increased, then it should perform QoS negotiation again by calling Negotiate(QoSRequirements requirements).

---

3. Ch 18 Replication [25 points]

a. 18.1

Ans:

Mean time between failures = 5 days = 120 hours

Failure time per failure= 4 hours.

Probability that a server is failed (P)

= Failure time per failure / Mean time between failures + Failure time per failure

= 4/(120+4) = 0.03226

Number of servers (n) = 3

Availability = 1 – Probability of all servers failing at a time

$= 1 - P^n = 1 - (0.03226)^3$

$= 0.999966$

Thus, the availability of the replicated service is **99.9966%**.

---

b. Study only pages 2 to 7 of the IBM Redbook

IBM High Availability Solution for IBM FileNet P8 Systems

Define Availability.

For a typical web app cluster (Web server, App server, Db server) with Active-Active redundancy using a total of 6 servers (2 replicas of the same web cluster), explain how Availability calculations are made and how HA can be achieved. Show your calculations.

Ans:

Availability of a server is the percentage of the time that a particular server or a process can function as required by the usage. Conversely, it also measures the amount of time required by the recovery after a process shuts down.

The total availability of a system takes into account the availability of all the components and measures the percentage of time, in which all the components function properly. It can be calculated as:

Total Availability of a system

= Availability of Component1 x Availability of Component2 x … x Availability of Component(n)

Consider the case of a typical web app cluster with 2 replicas each of Web server, App server and Database server with Active-Active redundancy.

Let us assume that availability of a web server is 99% (1% failing probability), availability of an App server is 95% (5% failing probability), and availability of a Database server is 90% (10% failing probability).

Based on these availability, we can calculate the availability of each component as follows:

Availability of web server component = 1 – (probability of all web servers failing at a time)

$= 1 - P^n = 1 - (0.01)^2$

$= 0.9999$ **= 99.99%**

Availability of app server component = 1 – (probability of all app servers failing at a time)

$= 1 - P^n = 1 - (0.05)^2$

$= 0.9975$ **= 99.75%**

Availability of database server component = 1 – (probability of all database servers failing at a time)

$= 1 - P^n = 1 - (0.10)^2$

$= 0.99$ **= 99%**

With we have,

Availability of the system = Availability of web server component
x Availability of app server component
x Availability of database server component
= (0.9999) x (0.9975) x (0.99)
= 0.9874 = **98.74%**

Thus, the availability of this web app cluster is 98.74%.

To achieve high availability, the availability of each component must be increased, since even one component performing poorly can lead to decrease in availability. Availability in a component can be increased by replication of the servers in that component.

---

4. Ch 19 Mobile computing [25 points]

a. 19.1

Ans:

A volatile system is a mobile or ubiquitous computing model with properties of a distributed system. For these kinds of systems, the changes in the model are common rather than an exceptional case to handle. The set of users, the hardware and software in these systems are very dynamic and prone to spontaneous changes. The main types of changes that can occur in such a system include:

1) Ubiquitous systems are volatile to device failures and communication link failures due to the presence of a number of mobile computers in the system.
2) The characteristics of communication can also change such as bandwidth and latency.
3) In these systems, different software processes or objects have multiple logical communication relationships between them. These set of associations are continuously created and destroy, thus leading to change in the system.

These changes should be considered as a rule in a ubiquitous system rather than as an exception.

---

b. Consider the Uber Car Hiring Platform. In this context, what is adaptation? How is it used in the overall design and what problems/requirements is it addressing?

Ans:

The volatile systems contain a variety of devices in terms of their processing power, the screen size, network bandwidth and different capacities like energy and memory. Some of the devices can be poor in some kind of resources while other can be richer in other resources. Eg, a simple volatile application can be used by different sized mobile phones, and tablets, having different operating systems. Adaptive systems are the one that are based on a model of such heterogenous volatile

systems and they adapt their behaviour while running based on the current resource availability on each device.
Consider the case of the Uber Car Ride Hiring Platform. It is an application to request a car with the ease of a few clicks in a mobile device, the drivers can accept such requests. The application needs to be highly adaptable based on the user's location, fare rates, the device used for the application and the preferences of the users.

1) Since Uber app can be concurrently used by users at different locations, the requirement for the location adaptability is a necessity. The application should be able to locate the drivers nearby the user's pickup location. In the overall design, the user's pickup location is shared with the uber cab database servers, which then responds with the nearby drivers list within a specific radius. These nearby drivers are then shown as a car symbol on the map. Furthermore, the exact location of driver and remaining time of the trip before and during the ride is also shared through the app. It addresses the requirement to know the exact location and time left of the ride. This can be considered to similar navigation problem in Google Maps and other navigation providers. This would require to continuously update the current location of the driver to the server which in turn would calculate these parameters and return back to the driver and the passenger.

2) The app can be used by devices of different screen sizes, and the preferences set by users beforehand for the device as well as for the application itself. The user application preferences could be shared to the uber booking server which would filter the booking items and respond back for a more user-friendly UI. This addresses the problem of user preferences in some level.

3) Furthermore, the application's fare calculation for a ride should also be adaptive for different rides. For this, the Uber's fare calculating algorithm can take into account various factors including the location, traffic, demand, distance and time of the trip. This would give the drivers a fair amount for every individual rides.
