**10. Disk Scheduling**

**1. Suppose that a disk has rotational speed 6000 RPM. What is the average rotational latency?**

At 6000 RPM, there are (6000 revolutions/minute)/(60 seconds/minute) = 100 revolutions/second, or 1/100 second = 10 msec/revolution.

When the head reaches the desired track, it is equally likely to be over any sector, so the average over a uniformly distributed variable is (High - Low)/2.

High is 10 msec (just missed it, have to wait a full rotation)

Low is 0 msec (just exactly on right sector - oh happy day!) so average is 5 msec.

**2. Suppose a disk has mean seek time of 25 msec, rotational time of 8 msec/rotation, and transfer time of 0.1 msec per sector, what is the mean time to obtain one sector of data? If there are 512 bytes/sector, what is the data rate?**

Mean time to obtain one sector is seek + rotational latency + data transfer time = 25 + 8/2 + 0.1 = 29.1 msec.

Data rate = 512 B/29.1 msec. = 17.6 KB/sec (note that K here for data rates is 1000, not 1024 as it would be for memory)

**3. Suppose a disk has mean seek time of 25 msec, rotation time of 8 msec, and transfer time of 0.1 msec per sector, what is the mean time to obtain ten consective sectors of data? If there are 512 bytes/sector, what is the data rate?**

Mean time to obtain ten consecutive sectors is seek + rotational latency + data transfer time = 25 + 8/2 + 10(0.1) = 30 msec.

Data rate = 10(512 B)/30 msec = 5120/30 KBps = 170.7 KBps

**4. Suppose a disk requires 0.8 msec to move the read/write head one track, and that the head is currently on track 35. If there are currently pending requests (listed as request name, track number pairs) of (A, 20), (B, 45), (C, 30), (D, 25), and (E, 45), what is the service order and total seek time (ignore rotational latency and data transfer times) to service these requests using the following policies?**

**a) FCFS**

**b) SSTF**

**c) Elevator, moving up**

**d) optimal**

a) FCFS: ABCDE is order, from 35 to 20, 45, 30, 25, then 45, so total movement of 15+25+15+5+20 = 80 tracks, total time of 80 x 0.8 = 64 ms.

b) SSTF: distances from 35 are 15, 10, 5, 10, and 10, so C is first serviced at 30.

From 30, distances are 10, 15, -, 5, and 15, so D is second, then A, then B and E (use arrival order to break ties). Order is CDABE.

Total head movement is 5+5+5+25+0 = 40 tracks.

Total time is 40 x 0.8 = 32 msec.

c) Elevator, moving up: Order is BECDA. Total head movement = 10+0+15+5+5 = 35 tracks

Total time is 35 x 0.8 = 28 msec.

d) Optimal is same as elevator

**11. I/O**

**1. The popular IBM selectric typewriter had 50 keys on the central keyboard, including tab, shift lock, left and right shift, return, and backspace. How many different characters were on each font ball?**

50 keys – 6 special (non-printing) keys = 44 printing keys.

44 printing keys x 2 modes (shift and not) = 88 characters.

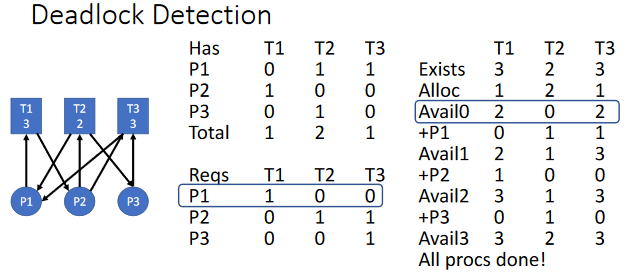
**2. Paper tape readers, used in early computers including the Colossus used to break Nazi ciphers, used presence or absence of holes in the paper tape to represent ones and zeros. Holes were typically spaced at 0.1 inch in both dimensions, and a high-speed reader could process 2000 characters per second, with one row of holes representing a single character. How fast did tape move past the reader in fps?**

2000 cps x 0.1 c/inch = 200 in/sec. (200 in/sec)(12 in/ft) = 16.7 feet/second.

**3. How does a computer convert mouse movements into movement of the cursor?**

The mouse detects motion in the X and Y direction (relative to the mouse itself) in more or less arbitrary but fixed units called mickeys. Movement in mickeys is sent to the computer, which uses a user-settable conversion to interpret mickeys as pixels or distance on the screen.

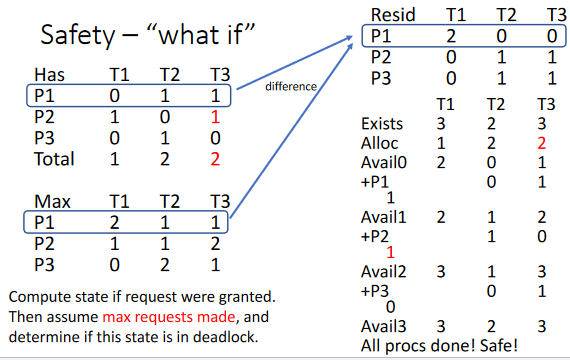
**12. Deadlock**

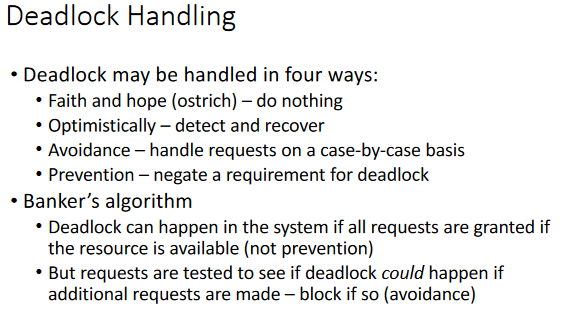


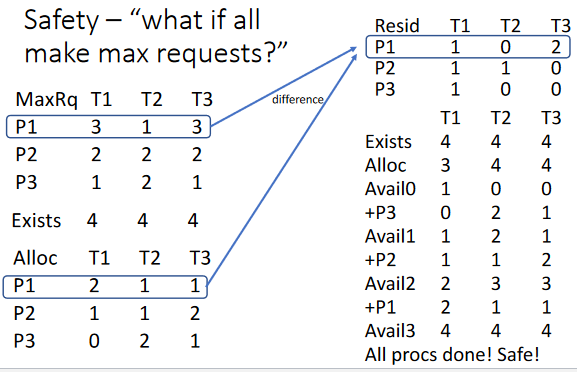
**Safe states-** In the Banker’s Algorithm, the maximum possible request must be known ahead of time.Requests are denied if granting the request could lead to deadlock.

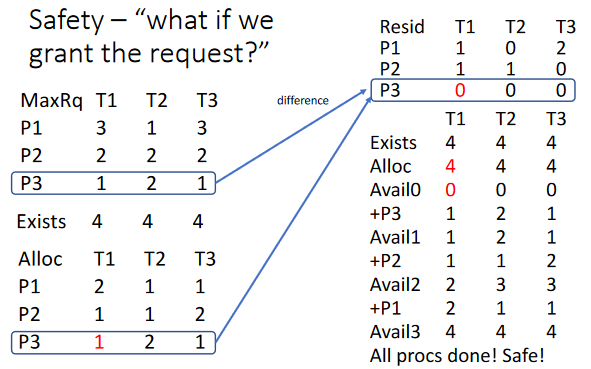
This could happen if all processes make their maximum demands (less the resources they already hold) and that state is deadlocked. So unsafe does not mean deadlocked, it only means deadlock would occur if all processes make their maximum remaining demand; safe means that the system will still not deadlock even if all processes make their maximum remaining demand.

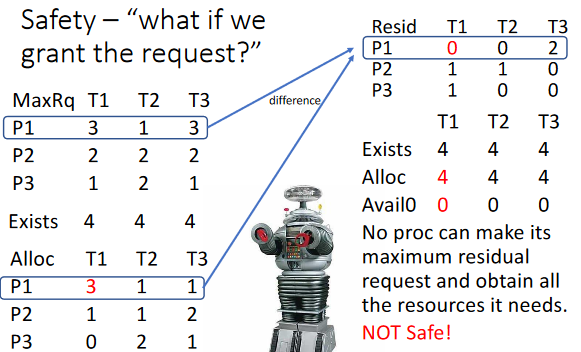
This matters because deadlock avoidance using the Banker’s Algorithm ensures that deadlock can not occur, at the expense of reducing concurrency.











**13. Security**

**1. What is required to store the ACM? What is required to store ACLs? What is required to store capabilities? Compare, assuming that most access types are either no access or public read access. Assume D domains and O objects with P permissions, with an average of N domains with non-default access to a particular object.**

To store the full ACM there are D rows and O columns with P bits per cell. This amounts to DOP bits total. For ACLs and capabilities, the domain or the object must be named, which requires log(D+1) or log(O+1) bits, respectively. The +1 is for a special domain meaning default (i.e., all domains) or special object (i.e.,all objects). We will assume a bitmap is used for permissions as in the ACM.

ACLs will require an ACL per object, with each ACL on average taking (N+1)(log(D)+P) bits for default access plus N non-default accesses, for a total space of O(N+1)(log(D)+P) bits.

Capability lists will require a CL per domain, with each CL taking on average (NO/D+1)(log(O)+P) bits, for a total of D(NO/D+1)(log(O)+P) bits.

Both ACLs and CLs use much less space than the ACM, assuming that N (P+log(D+O))

is less than D, which is usually the case.

**2. Compare the effort needed to revoke access rights to an object using ACM, ACLs, and**

**capabilities. Which are easy and which are difficult? ACM and ACL are easy - just turn off the permission bit in the right entry.**

CLs are hard - all the CLs must be searched for the object and the permission removed for that object.

**3. Compare the effort needed to determine which domains have a particular access permission to an object using ACM, ACLs, and capabilities. Which are easy and which are difficult?**

ACM requires a search down the column of the object to find all domains with the

particular permission. This is moderate effort (D entries need to be searched).

ACLs require search of the list for domains with that permission - this is easy (only N items on average need be searched).

CLs require that every domain's CL needs to be searched - this is hard.

**4. Compare the effort needed to determine to which objects a domain has a particular access permission using ACM, ACLs, and capabilities. Which are easy and which are difficult?**

ACM requires a search down the row of a domain to find all objects with the particular permission set. This is moderate effort (O entries need to be searched).

ACLs require search of the ACL for every object - this is hard.

CLs are easy - just the CL for the domain in question needs to be searched.

For the following, assume levels of Low, Medium, and High, and categories of A, B, and C.

**5.** **Suppose a process has label = <Medium, {A, B}>. Files with what labels can it read?**

**List all such labels.**

<L, X>, <M, X>, where X = {}, {A}, {B}, {A,B}.

**6. Suppose a process has label = <Medium, {A, B}>. Files with what labels can it write?**

**List all such labels.**

<H, X>, <M, X>, where X = {A,B}, {A,B,C}.

**7. Suppose a process has label = <Medium, {A, B}>. Files with what labels can it both read**

**and write? List all such labels.**

<M, {A,B}> only