Introduction to Operating Systems

Monday, January 22, 2018 8:32 AM

On edimension, there's a handout and syllabus.
Course would be graded on activity, quizzes and homework. There would be programming assignments and labs. Major weightage given to the two exams. Each would be worth 32%. In class activities, we would have to go to edimension. We are making it clear that there would be a quiz every time there is a lab session. We would leave for the lab session, and first 10-15 min there would be a quiz. There would be scheduled 7 labs and 2 programming assignments. Like to stay and finish the lab exercises, and most can be finished in 2 hours. Please submit all the code.

The two programming assignments would be done in pairs. There would be 2 - 3 weeks to complete programming practices. The second programming assignment, there would be a presentation in the end. There would be checkoffs. There are some rules about the drop, for example you may be on MC and there might be a quiz, so can drop at most 2.

On the other side of this handout, can see the syllabus. The topics are specified in the syllabus. This course is structured in two parts: portion before mid-term would be on operating systems, while the portion afterwards is on networking. This would be your only exposure to the operating systems. What are the modes of operation, what are the systems call and the operations call.

What is the CSE all about, why should we be doing these course. Both OS and networks are aspects of daily life. Computers are being used everywhere.

ACTUAL CONTENT STARTS HERE
What is an operating system? What is an OS? OS is nothing but another program. Think about it, what is a computer. A computer instead of hardware, it has memory, CPU, and anything that runs on that hardware is a program. If you compile your java code, it is a sequence of operating code. What makes an OS so special? There are three things: The operating system must be an interface to the hardware. It must interact with the hardware. It has to access the hardware, it has to give the hardware if an IO device, it has to move between hardware. All that would be done with the help of an operating system.

It has to provide a context/environment for the application program. What application programs are: anything you might want to run in your computer. What's happening, is that applications are running on your profile. Third thing is an interface with the user. The operating system is always running. As soon as you switch your device to your computer, the operating system is always running. When we look at an OS in this manner, you will see that there is a variety of computers and OS. Until now, you haven't used a mainframe or a mini computer. Mainframes and super computers are expensive and powerful, so they are shared between users. They have to have continuously running mainframes. The OS provides an interface with the user and ensures that the user has a fair share of the resources. Resource utilization becomes an important source. There are multiple users, but I'm the only user. How to make this device appear most user-friendly for this particular user. There is a spectrum of different types of operating systems that there may be. There might be a OS that is balanced across a large number of users. But resource utilization: can ensure that the operating system may have to be used.

So once again, we have an OS, and the OS is sitting between the hardware and the application programs and users. This is a layered view; please note the layered view reversed. There are four layers over here. The users interact with applications, applications run on top of the OS, OS run on top of the hardware. The principles of layering occurs in computers a lot.

Hardware consists of memory CPU and other devices. Application programs can be word processors, compilers, web browsers. Users can also be other machines or other users. Users can be robots for example.

Roles of an operating system as a resource allocator: it manages the resources of an operating system. It decides between conflicting requests for efficient and fair resource use. When there is a large number of users, OS determines the fair amount that they use. The OS is also a control program: it controls the execution of programs that may have gone astray. The OS has to be the entity that steps in; does something to make it function properly. It's more like a government. Now we have a hung government in the US. What is the purpose of a government? Does it do any work? But then can you function without a government? It's not possible, especially in today's modern world. You need the government to regulate and control, the operating system does not do any work. You need the OS to ensure the other entities in the system, if they are making use of the resource as it should be.

This brings us to the notion of the dual mode of operation. When you log into a system, be it a linux or a mac, you log in as a user. But then, what happens if let's say, you were to do something like an illegal operation. Since you just finished the java course, let's say you divide by 0. Or you access a memory location not allocated to your stack space. If there were no OS, things would go haywire. The register would be loaded with some arbitrary thing from some place in memory. Therefore, you would need a part of the system called the kernel to take over. The kernel is that part of the operating system that is always running and ready to take over in case it is needed. Maybe the user program needs some data from the disk. The disk access speed, which is in the order of miliseconds, is very slow, as opposed to memory access speed, which is in nano seconds. You can imagine that if it takes 10^-3 from this to memory, should the CPU be waiting for. It should be loading other tasks. It should be letting other programs.

I need this piece of data from the disk. While the disk controller fetches me from the other area of the disk, I'll place it somewhere for waiting. The user mode and the kernel mode. It's hard to define the operating system, if you can sort of describe it, it's not very easy. It's that one program that is running all the time. That's the kernel. What are the kinds of things that a kernel can do, which an ordinary user should not do.

There are two things: input output, and access to memory locations. When I say access to memory, I say any part of the memory, which users should not touch, where all the kernel data structures, interrupt service vectors are stored. Those regions should not be accessed by user programs. Kernel should be able to access IO devices. It gets data from the disk, and vice versa, a very IO device, this IO device, there are peripheral devices in the past. There is a dual mode of operation: User and kernel. These two modes are always interleaved. When there is a switching over from one task to another, the OS will govern that task switch, we call it a context switch.

A end-user, we will see the process user, an end user process can actually generate a call to the operating system, that's a system call. Why need it? A process might be wanting to operate a file from a disk, or enter a character from key. The ability to generate a system call, is something that every user needs to have. Write into a part of memory, the ability to write into an arbitrary memory, belongs to the kernel. That's the privilege instruction; only the kernel can issue those instructions.

How to get the operating systems into raw hardware. How do we load an operating system into that? The bootstrap loader, a sequence about twenty instructions; you press a button and you load a paper tape reader; then that bootstrap loader will go to the operating system. So you can have a bootstrap loader stored in ROM, generally known as firmware, and the bootstrap will load operating system into your hard disk. Then it will handover to the OS first, that operating system will give a login prompt. In case of windows or mac, it automatically hands over to the

RECAP:
Looked at certain views of the operating system
Looked at the dual mode operation: user and kernel mode
Looked at what is called a system call, being from the software as opposed to a function call
Notion of hardware interrupts
Looked at processes and define a process and will see process and concurrency are defined
Looked at memory hierarchy.

GUI. These are the things happening underneath the hood.

X86 architecture: What does it mean? It's an agreement, that when I write my java program, it is assumed to be able to run there. We tend to use the word organization. In a computer, you have one or more CPUs, and this disk controller, a USB controller, a graphics adapter, all going over a shared bus, and on the other side you have a memory. We call this a von Neumann computer. Both the program and the data are residing in main memory. It fetches words from the memory to the CPU and one by one executes that. The next instruction occurs, in the pipeline from disk controller to CPU. They need to put data into memory. It needs to put a segment of data into the memory, so it will need to use the access to memory as well. You see that the all important things is memory and the bus, and when we said the OS is nothing but a program, it is running in the CPU but it is the same as any of the programs. It is arbitrating the access to memory by all the other devices.

Even the disk access has to be mediated by the CPU. The disk controller itself has a powerful CPU. We can directly dump memory, and just invoke the CPU. How does the disk controller that it's task has ended. That the memory contains the data that was requested. There's a CPU here and here, but they are not at the same level. This is not a computer network. The CPU here interrupts the CPU here.

It's a special kind of signal that interrupts the execution of a CPU. Once you get a interrupt, procedures stop and you look at what causes the interrupt to occur. That's the only way in which you get the access to CPU. Say your keyboard, gets the interrupt to CPU, and signals the transfer of data to the memory, and shows where the notion to the data. There are hardware interrupts and software interrupts.

CPU is not the only component able to run code. Other devices execute concurrently because they themselves have their own CPU. When an OS is loaded, a bootstrap loader loads the OS, serves the memory for certain processes. Whether is MAC or Linux, they do it the same way. There is a load memory and an interrupt vector. So all you need is a vector of addresses to interrupt service routines. There is a first one from device 1 that points to space in memory that knows how to run the interrupt vector to device one.

The interrupt transfers control to the interrupt service routine generally through the interrupt vector, which contains the addresses of all the service routines. There are separate interrupt handlers that handles interrupt via the various USB, graphic interface. This is all predefined. These handlers have to be written so that they don't access all parts of memory that are beyond the bounds of particular users. There is a lot of work where it comes to computer security to ensure that the code does not have access to all the other parts of the computers.

Interrupt architecture must save the address of the interrupted instruction. You must make sure that this routine does not happen again. So incoming interrupts are disabled while another interrupt (same or higher priority) is being processed to prevent a lost interrupt or reentrancy problems.

Then we have a trap. A trap is a software-generated interrupt caused either by an error or a request by user code. Latter allows a user program to invoke a OS function (system call) which user processes can call that function but the kernel is the only one that executes. It will run only form the kernel mode. The engine points are carefully controlled into that kernel. Hence entry points are carefully controlled. Why? We will have to go via the interrupt vectors here, but a system call is not a hardware interrupt, but even those entry points, say the OS, maybe 45/0, the entry points have to be carefully controlled.

Need to make sure that two users do not write into the same space of memory. This entry point eneds to be carefully guarded, need to make sure that the kernel is carefully executing system call, and make sure that the system call does not write into portions of memory that are into the process. So when the process is executing from here, calls a system call, the interrupt handler of the system routine must not access some other address space here. It is only operating within the address space of the system that calls it.

So further interrupts are disabled. The state of that process has to be saved. That process generates a system call to the hard disk, to get a request from the disk. And that input request is handled by the system call over here. So once that happens, this process must be put to sleep for a while while this data arrives. You need to save a few things; need to save the program counter and the registers, and they would not have been saved by then. Anything that pertains to the state of the process of that time needs to be saved by that hard disk. It's something called a process control block. PCB. Once all that is saved, the kernel will go off and wake up another process. (study later)

See, this is what's happening. There is an IO request, happening from other process, then there is the user process executing. Since the IO request has finished, the controller interrupts the CPU and CPU executes a interrupt process, write the save data and then go back to the user process. So the user process would have been put to sleep and then woken up again. There are a lot more things that are happening behind it. It could be some other user process. At any given time, you can just look at all the other processes that are running at any given time, and they are all operating on this basis. Comes to the speeds: nanoseconds, miliseconds. You have plenty of time; there are many processes in between.

Notions of storage structure: When we talk about storage, and we talk about memory. We are talking about main memory or RAM or DRAM. But is that the only kind of store? We just mentioned hard disks, another kind of store. Then there are faster: it's a hierarchy. We still have tapes: the cost effective means of storage. If we have hundreds and hundreds of tape drives.

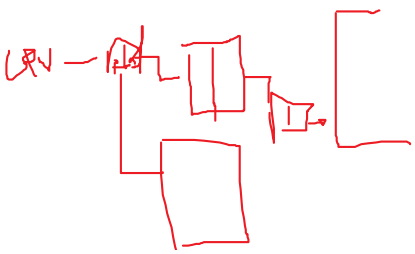
Magnetic tapes, slower than optical disks, closer than magnetic disks. Magnetic tapes are a linear machine, while optical and magnetic disks are circular so can access any portions of the file. Magnetic disks are slower than electronic disks (semiconductors), slower than DRAM. Electronic disks give the notion that it is a disk. It is a slow speed RAM backed up by memory, backed up for power failure. Then you can to the main memory. CPU accesses main memory upwards, doesn't access any of the disks. In between the main memory and storage, they are at many levels of storage. You can have banks of memory architectures.

Between main memory and registers you have cache, which is very high speed, reduces the access time. There is a hierarchy and at every point in the hierarchy there is a performance and a price. Performance goes one way, price goes up.

There is this element of volatility. Some of those devices are volatile, some are not. IF there is a power failure, it will disappear. Electronic disks are also not volatile. Main memory is volatile, cache is, register is volatile.

Idea of caching is to bring stuff into a higher level of storage. It will cache from main memory to cache, from cache to register. It is to copy information into faster storage systeml main memory can be viewed as a last cache for secondary storage. It is something one can do throughout the hierarchy. Disk surface is logically divided into tracks, which are subdivided into sectors. They are radial parts, and each of these parts is a sector. The circular ring is a track. The disk controller determines the logical interaction between the device and the controller.

When we have memory, being brought from a file to main memory, it happens to something called a paging architecture. The data is stored in a file, the size of the page is determined by the architecture, sometimes 24 bytes, but the key thing to remember is that on the disk; you have a number of pages stored. You only bring in those pages that are needed. What the CPU does is to fetch the pages needed and puts into main memory, using the paging architecture to do so. Similar things that are done is between the main memory and the cache. The cache is of course not as much; only a few pages of main memory are brought in to the cache.



but the key thing to remember is that on the disk; you have a number of pages stored. You only bring in those pages that are needed. What the CPU does is to fetch the pages needed and puts into main memory, using the paging architecture to do so. Similar things that are done is between the main memory and the cache. The cache is of course not as much; only a few pages of main memory are brought in to the cache.

So what happens when a page that is supposed to be in the cache is not found there. The page table resides in main memory. It is just as slow as main memory. It looks for that page number in the cache and the page table. If there is a match in cache, it will go for frame number, compose the physical address and bring from physical memory. Slower, if it doesn't find a match, will go to the association table. You can see that there is a cache here, which is much smaller, which is used to access seconds of memory. But this is to illustrate the principle, and have a simple formula, which I will leave to you to derive, say the cache ratio is r, you will find the data in cache.

Faster storage (cache) checked first to determine if information is there
- If there is, information used directly from the cache (fast)
- If not, data copied to cache and used there

Cache smaller than the storage being cached
- Cache management important design problem
- Cache size and replacement policy

Cycle time = rt + (1-r)T

Migration of integer A from disk to register:
Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy:
Magnetic disk -> main memory -> cache -> hardware register Value of A is fetched from magnetic disk to main memory, generating some time, loading it, then the main memory would cache it througha similar structure. Then it will load the value of I to a hardware register.
At that point, there are different values, with the same variable r. In a multiprocessing environment, you could have a problem. In a multi-tasking problem, different processes have a different value of what A is. Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache.

All modern operating systems have the ability to support symmetric multiprocessing architecture. My computer will have more than one CPU. These Cpu will have access to memory in the same architecture as the diagram. In this case, we have just one CPU, but in the case of systems nowadays, we can have the possibility of 8 CPUs. So each CPU will have its own cache and then we will have memory, and the rest of the memory, whatever you have, and then have the cache coherency problem, Say the cache has integer A = 5, processed, but not flushed to memory and thus the other CPU have not updated value of I.

Dual Core Design: Many companies are producing multi-core chips. Why would it be better to have dual cores on the same chip. The benefit is that when you have multiple caches and registers, the speed is much faster. Want to reconcile the cache value: it's faster. Reconcile the memory value, it's faster. Because both caches are on the same chip, by means of other routines, then it will be much faster, because they have to be on the same things. There's another type of architecture; a blade. A blade is a complete set of CPU and networking card and memory, and it is operating in data centers.

Where there is secondary storage, there is a storage network. And the storage area network is called a blade.

Operating System Structure

Wednesday, January 24, 2018 11:33 AM

Look at its position on top of the hardware. Any user shouldn't be able to access the hardware without system manager.
Multiprogramming provides user interface. Needed for efficiency. Looking at dual mode operation. Dual mode means that there are two modes of operation: user mode (few privileges) and kernel mode (all privileges). All the user mode is to make requests. We use interrupts. We call them software interrupts/traps. It is called system call as well. One can access to the kernel to do something on behalf of the user. The kernel would go to a supervisory mode or a privilege mode. Supervisor mode is the same as privileged mode. Whenever user wants to fetch data from harddisk or send to screen, it does so by means of system call.

Whenever user program uses IO, it has to go through system call. We'll be looking at dual mode and management of processes. What is a process as opposed to a program. There will be activity in the middle, and then talk about user interface. We need to understand how time shared operating systems came to be.

Computers were new and expensive; most didn't have access to them. Use through jobs: submitted for execution. We have multiprogramming because how powerful your job is, it cannot keep CPU busy all the time. It might need to wait for input from a device; a keyboard. It was pretty much a disk. You need to put that job to wait. There are two ways of waiting; bridge the gap between computational structures: waiting by polling. It's a kind of busy way; CPU waiting until device is done. You're wasting a lot of CPU resource. We talked about interrupts in the first lecture; there are hardware and software interrupts.

What is a hardware interrupt? It's a signal. Signal is something that is an event that happens. That event says that this hardware is signalling the operating system that you are done. Someone presses the keyboard, then the operating system has to send a signal to the interrupt handler. Various parts of the service routines, just to jog your memory. That interrupt is a mechanism by which the OS responds either to hardware or software.

When you have a job. Once a job has an IO request, it sleeps for a while and another job brought into main memory. A job is executed in nanoseconds space. Typical time for a fetch or access to memory is in order of nanoseconds. But time taken to access harddisk is done in miliseconds. Therefore, this job needs to wait until IO is done, and during that time, another job can be taken to the CPU. So this is actually RAM (primary memory, not secondary memonry)
Another job is being processed for execution by the computer. We are scheduling the CPU.

One job is selected and run, another is put to sleep for a while. Computers became more faster, and can access through terminals. We can have a shared access to personal computers. So we can have multiple users using the same share (mainframe). Through their own terminals, you have the illusion that you have the terminals to yourself. Seven characters per second is slow, even when you wake up for the job, process that incoming character, and then put it into sleep. You can accommodate a large number of users, and give the illusion that they have a single computer. That is time sharing. Many jobs residing in the memory, executing at the same time, but no notion of user interaction through time sharing. When end users come to the picture, you give the illusion that you have the computer to yourself, and this is the speed time difference. You have to maintain the response time criterion, because if you go slower, the user is not very happy.

The operating system has to maintain a response time of roughly about a second. This is one of the

criteria of timesharing response. We don't use the term job, but the term process. At any given time, if I have a Linux/Mac/Windows, it will be executing a large number of processes. What are these processes doing? Various tasks on my behalf. Looking at email, and background daemons. Say Hello World .java also becomes a process executed. What's the difference between a process and program? A program is a static thing, you can have a printout and submit as a assignment. Process is dynamic; like a river; but you can't define it as a tangible thing. It's continuous and dynamic. How can we get a handle on a process to distinguish from one and another.

You have created the heart of a CPU in Computational Structures. You also have to have certain other things to make it into a computer. You need a PC, program counter, to keep track of the memory location that you have on. Stack counter, where you have functions keeping track of other functions, and the status of functions have to be put into the stack. You have some registers, like PC and SC, and special purpose registers, and all of the contents of the register are what constitutes the snapshots of processes. You have a state, take a snapshot of that, and when changing from one job to another process, have to save that information. You are saving the context.

You come to the notion of context switching. What is context switching. When one process is going to wait for input, another process is brought in, so the context of this process would need to be saved somewhere. Context information has to be brought in, and then the job would then execute. If several jobs are ready to run in the same time; when I talk about context switching, job not ready yet. But when all jobs are ready, waiting, how you bring in is through CPU scheduling.

The last concept is that of swapping and virtual memory. Why would you need to swap out a process? This job resides in this part of memory, but you have not much memory, and you would like to have huge amounts of address space. It's a logical concept, address space, the way that you can write your program assuming you have access to logical address. Logical address vs physical address. Logical address is huge, can't accommodate in main memory; so secondary store for main memory.

MMU, hardware chipset, every reference to memory go to hardware, and tell whether this region memory resides in primary or secondary. Need to do a context switch, and give the CPU some other task while fetching.

Key notions: CPU scheduling: Need to schedule because CPU needs to be given to another task ready for execution. Second notion is job scheduling. Job need to be brought in and out of the secondary memory. You can have hundreds of processes, not in main memory in the same time. Virtual memory is something needed to allow for process to have very large address spaces for computation spaces. Done through the OS, but made to a part of a hardware, you have main memory units to do this task of virtual memory access.

DUAL MODE OPERATION

We can also have hardware support for dual mode operation. We have two ways of operation: kernel mode and user mode. Why should this be supported by hardware? It's not a matter of speed because in the other case, the latency of hard disk is a speed issue. A user wanting access to kernel routine to make a request and switching the process temporarily, which are more than what it has at the moment. You can have a software system call. Why would it be important to do this in hardware? If you do kernel mode in hardware, it is better interrupted. The more general thing is that it's a protection/security issue.

Look at many security threats come in through the operating system. Entered through the network and attacked the OS. Very important for hardware mechanism to prevent illegal access to certain regions of memory. That would be reason to have hardware support for dual mode. Every time I make a request, hardware would check if in kernel mode or end user mode, the mode bit is 0 for kernel, 1 for user. Mode bit gets set to 1, and the last thing that happens when control goes back to user is that the mode bit

goes back to 0. The kernel is 0 and user is 1.

If anyone can switch from kernel to user, then you will have chaos. Anyone can switch the mode bit to change my process to your process ID. Switching from kernel to user is a privileged operation. How the bootstrap loader starts the operating system: it does initialization, sets the tables and devices, and hands over to the user. You have a nice GUI, and user has access to part of the resources. Started out in kernel mode and gave the user the access. Should be able to switch the mode bit to kernel. The very first thing the kernel would do for switching should be given to anyone.

There is this RETT. Another name for system call is trap. Many of the CPUs support this RETT instruction, it means return from trap.

PROCESS MANAGEMENT
Main ideas already said: when there are large number of processes executing, need to manage those processes. Say I write an erroneous program running in an infinite loop. Program goes on and on. Not requesting user resources or hard disk, and should this program run continuously? No, other processes awaiting. How would we solve the problem of a runaway process. How to solve this problem? Basic idea is to have a timing mechanism. Every process would have 10-20 50 miliseconds of execution time. Then after that, put to sleep. If that process hasn't made any request for input and output, then that process is taken out and another process put in. How to put in a timer?

A clock is an electronic device; it has sophisticated machine, You have a clock that can be very stable; it doesn't drift. The oscillator is on the quality. All they have is that oscillator that gives you a pulse every few nanoseconds. All you get from the clock is a pulse; how to decide to put a process to sleep for a while? Can I use the time of day to put the process to sleep? Count the number of pulses. Have to have a counter. A timer is a counter.

Should it be a privilege mode or a user mode? Should any user be given the privilege to start their own timers. They can generate the pulse whenever they want, and the setting of the timer has to be privileged, but the reading of the timer should not be privileged. Timer is a software artifact, access to system clock, needs to be able to send pulses (units of time) to itself. It just depends on how long you want the process to be running for. Pulse would be a signal, appear to that particular timer, then timer goes to x, and then it puts process to sleep.

MULTI THREAD AND SINGLE THREAD
Processes all operate within own address space. Say this process goes to user X, there's no way process 1 has access to this address space. That's allocated by hardware. There is another level of parallel computing; that is threads. You have multiple CPUS. At the thread level, running different processes or threads? Threads right? But how? All you can do is start a process, but if java gives you the capability to start threads.

This is called the address space of a process. As my process is executing, it starts a number of threads. Threads should not be larger than the CPUS you have. In scientific, you may have hundreds of threads, but parallelisms is the number of CPUS. You have assignment later on where you program these things. All threads have access to shared address space, while processes don't. This squiggly line is the execution of one thread. This is happening in parallel. Different CPUS have their own instructions and registers. They each have their own program counters.

(Dual - Core Design Slide)
OS prevents one process from accessing another address space. If one thread inadvertently accesses another address space, kernel would not intervene at all. You would have to deal with the shared memory in whatever space that you want. Both are looking at the same piece of memory, you will do a lot of damage to yourself. This motivates the discussion on synchronization. Need to have

synchronization between the OS, and it is all protected by hardware.

? If you have multiple CPUs, can they share the same address space? Not at the threading level, but can allocate shared memory as the OS. Nothing to do with threading. Can have shared memory, shared segment allocation, go through OS sys call.

SUMMARY:
Creating and deleting both user and system processes
Suspending and resuming processes : important need to have fairness, even with no IO, all users/processes have fair share of time.
Providing mechanisms for process synchronization: If two processes need to share something, the operating system gives certain mechanisms, called synchronization mechanisms.

Providing mechanisms for process communication. When you do a networking mechanism: So I have a process sending out messages, and receiving messages, and processes are communicating by messages, but no shared memory. On the other hand, both these processes are on the same host, you can communicate by means of shared memory. Say I have a large message, don't copy! Just switch the pointer that your message has arrived. Two modes of operation: Message passing and shared memory. Message passing is the default because it is clean; no shared memory regions and so on.

Providing mechanisms for deadlock: It's in week 4, two processes aren't moving further. P1 waiting for P2, waiting for P3. There is no advancement or progress.

? If I have one CPU, can I have concurrency? Can I have two processes running at the same time? Your understanding of concurrency is at the thread level, but notion comes from process level. The notion of concurrency is virtual, can have two processes executing concurrently. I have only one CPU, do I have concurrency? Yes, but I don't have parallelism. Parallel computing is what I don't have because this idea that all my processes are running at the same time. They all think they have it to themselves. Because of this high speed that the user is running. More than one course, I can have real parallelism, I can execute my process faster. This difference between real and virtual parallelism.

OTHER IMPORTANT OS SUBSYSTEMS
There are certain ways to manage secondary storage devices are magnetic disks, but we don't about tapes and CDROMS. We only talk about how the artifact of the disk which is called a file system. It's an invention to make this access become very easy at the programming level. You will not find it with any IO device. Mass storage management. IO management not discussed.

At the operating system level, we support mechanisms that help to enforce security. What's the difference? Protection is to have some mechanisms, security has to go beyond that. Even to prevent inadvertent harm. I'm a bad programmer, run lousy code, but I don't do that maliciously. Protection has to protect against erroneous attacks and malicious attacks. Whereas security has to do with intentional attack. The defense of a system against external and internal attack. You need protection systems and security together. Just by having traps, dual mode operations, not going to deal with the security problem. Why? Say someone has stolen my password? Many things that cannot be done by the OS. Even beyond the realm of netwokr security, there are other means to enforce that. The second thing is that the mechanisms for access. There is this issue of access control: three levels of access computer: read write and execute, three classes, user group and others. In linux, you can issue a command like ugo + rwx. In other words, I want, for all users, read write and execution purpose.

Privilege escalation allows user to change to effective ID with more rights. Computer has an ID, when user logs in, it has the users. The OS remembers users by an ID , a UID. Get UID gets the user Id. Sometimes need to execute with a higher privilege. Will execute your process with a higher level of privileges. So has the notion of set user ID to be whatever runs at a higher level of privilege. So it has

something called an effective user ID, to find out how to go from user to super user and it is done in the management.

Need this capability for dual mode operation, need to have privilege escalation, but also will create problems for you. Privilege escalation has to be done carefully.

COHORT 2

We can look at operating systems services as opposed to operating system functions. Functions are the low level details. Let's look at the services that an OS must perform on behalf of various activities. Perspective of end user must present a nice user interface. Initially, it was in a batch interface. Batches are still around, are called .bat files. They are relics but can still write and execute them. So use command line or GUI.

There are various functions of an operating system, all end user oriented. Resource allocation, IO operations, are all system oriented to maintain fairness and security on behalf of all the users. Why would accounting be important? Accounting is keeping track of what resources the user is used. That amount of memory, disk space, CPU time. So that we can tell which programs are using the most energy. Can tell which are the ones consuming most resources. Don't know why system is so tardy. Find there's a bug somewhere there. There's a major financial matter why. If there's a cloud, then there's no mainframes to access in this fashion. Do need to keep track of what the cloud's resources are used.

Need to get used to cloud. All you are using right now Is transferable to cloud. AWS will bill you for everything.

Communication: Processes exchange information. Network communication through message passing and shared memory may be conducted. You can pass a pointer to shared memory to another. Error detection: Errors happen in a variety of ways: some illegal access to memory is an error. Someone pull out a cable from some IO device and reported as an error. The debugging facilities greatly enhance users and facilities. Many Os's give you core memory. Core dump means the contents/states of the memory at the time of failure. So you get this core down. You can look at core down through a debugger and see what went wrong.

We talked about protection and security, it's just repetition. Accounting, resource allocation. That brings us to the final topic of access interface to OS services. What is this shell? You will be programming at the shell level. You can issue these commands: on certain operating systems, these commands are pre-built in. Under windows, MSDOS, you can only have a rename as ren. Or remove as del. These commands are pre-built and you can't add to them. But in Unix, you get the kernel small, but allow you to add to it.

What is rm? Remove files. Because everybody has access code to source code, if you have super user privileges, you can add to the functions. Your shell can invoke those commands. Sometimes, implemented kernel, by system programs, but there's no one saying that you can't write it yourself.

Cohort 1

Monday, January 29, 2018 8:28 AM

In our last discussion on OS services, there are two classes of services. Services for efficient operation of the system via resource sharing, and services that help the user use the computer. To give the user better user interface, better experience, faster response time and so on. Let's go through these services and identify which belong in which category.

2) User interfaces have three kinds: First was the batch chronologically, because there were no terminals. Then the Command line interface, in linux. It's not just GUI, it's just people prefer it on Linux systems. GUI gives you a metaphor of a file system on desktop. That's called a desktop metaphor. Came from Xerox Palo alto.
3) Let's look at the other operating system services. Program execution is 2. Accounting is one, as the purpose of accounting is to ensure that people pay for the use of computer resources. There's a notion of how much of a resource people have used for the sharing of resources. How about file systems? 2. IO operations. 2 Error selection is also 2. Resource allocation is 2. Protection and security is both. Communication is 1.

By clicking on a particular icon, can achieve desired outcome in the GUI. In a CLI, you have to type it in. A user-friendly desktop metaphor, you double click on that icon to activate a program. Many systems now include both CLI and GUI interfaces. Example being your own computer. Then you look at Shell command interpreter. There are various kinds of shells, shell is a program that is waiting for user commands. So you have to do an assignment in your lab where you have to write a shell program that waits for user input and waits for it in the background.

? Any program takes in some kind of input and then an output. So is there anything special about the shell? Resource allocation: privileges? The answer is, the shell needs to come back and do more things. How does it achieve that? Your shell has to come back and do something for you. Say you write something in Android studio, does it come back for more input? You have to put something to execute in the background. That executes concurrently with what is in the foreground in the shell. The concurrency is what distinguishes shell from programs.

If shell is so special, why is it a user program rather than a kernel program? The original computers have the shell as part of the kernel. The MSDOS, it is a dinosaur of the operating system. It was part of the routine. Why would it be necessary a part of the user interface. It's more flexible that way: you can have your favorite shell, change your environment variables. You need to have a kernel that is modifiable by users to have a shell that is customizable.

System calls are important as it is the only means to communicate their request to the kernel. It's not a good idea for users to have access to special software, because then they can harm each other, and just call the system resources. All the user can do is to make requests of the kernel.

When we write software programming languages, one way would be to allow those system calls be directly to the programming language. What we have is a thin layer of software, a system interface between the programming language and the OS. This is known as an API. So for example, A C language read or write; that is supported by C compiler, but would translate to an equivalent system call, which is also called a read or write. In windows 32, the names are quite different. There are three types of API, three kinds of standardized API, win32 for Windows, POSIX API was standardized for all varieties of UNIX, LINUX and MacOS, and the Java Virtual Machine has its own API. When you have an application that needs to access the file system, the application is in C, and will have some API that will be translated by the operating system with an equivalent system call. What would that system call be written in would be native to that machine. If they are very low level system calls, those are written in assembly language because it is much faster. You can have much efficient manipulations if written in C and assembly language. At a higher level, the API would be in several languages, depending on the programmer. If you use C, you would be using POSIX pthread library in Lab2.

What would be the system call that process builder maps into? We saw that there is an API, and beneath it is a system call that machine language does. Fork or exec are OS system calls and the program language that java process builder supports. What is the advantage of building API rather than having direct system call

1. So that you don't access things that you are not supposed to access. Protection is one. Allow a general API that allows for secure capabilities.
2. I have a variety of architectures, with different applications of system calls. You don't want the programmer to know, to apply portability to every architecture. Some times the system call can be exactly the same as the API. Most of the time system calls and API are different.

In our course, we would be using generic system calls. You have to find out the relevant ones for C and Java.

Now we briefly talked about the manner in which the program executes the interplay of the program and the system call. What our program does is to open up the source file and destination file and copies out from source file to destination file. So it just copies and paste. What would be the system calls that invoke during this simple thing. I can have one command line interface, at the back, this is what happens.

1. You don't know which file it is. There's a way to pipe. Say you don't use pipe, we ask the user to input the keyboard. So acquire input file name. As the program waits for the input file name to be typed in. For each character, that's an IO operation. So that generates a system call. You are inputing a file name that generates a number of IO requests.
2. You also need to write the prompt to the screen. Another IO request for the device
3. Then accept the input.
4. Acquire the file name
5. Write prompt to screen
6. Accept input
7. Open the input file
8. If you open input file, you invoke a system call. C and Python is open. You have a open system call. That gives rise to a call to the operating system, which sends a call to the file system. So that's a call right here. If the input file does not exist, you have to terminate the program. If that file exists, you have to start the process of creating a file. If the file exists you have to abort. Either you abort the file or ask the user if you want to overwrite that file or not. You can have another round of system calls whether or not to overwrite the file. Then there is a loop of reading and writing to the output file. Let's say you're doing it character by character. Every output from the file, you have a call. Then until the read fails. There's a lot of system activity going in, that you are not aware of. You close the output file and write the completion message to screen. Terminate normally.

Typically if you do a man on read, you get a return value, you do a read which is an API call, and then some descriptors. And then a maximum size. Not sure about any API or system call, then you will have a lot of information about it.

Now, the way the system call is supported is quite similar to the way hardware interrupts are supported. At the hardware level, whenever a device wants to interrupt the CPU it raises a hardware interrupt. At the software level, there is a system call from the software value, a system call interface maintains a system table. It's also known as a trap. So the caller needs to know nothing about how the system call is implemented.

Caller doesn't know anything about it. Caller needs to know what you see in the man page. In the definition of the API, as long as the call adheres to that definition, the system call should be supported by the underlying show. So pictorially, it is something like this. The system call and the API have the

same name. At the kernel level, you use the same name open, which is okay, and the system call interface looks upward, supports open, and find out what are the parameters needed and so on. The system call issues a syscall called open, and goes to the entry I in this table, which is the entry for open(). All the implementation for open() system call is that location, and will return upon successful implementation of this kind of code. This is a simple illustration of how the API is supported by the system call interface.

This is my C program, which includes printf('greetings') is supported by standard C library. When I include the standard IO, that piece of code with the system call is linked to the program. Do it yourself with a make, which is a way of putting the source code necessary to make a library, resolves all the dependencies. You have to include them in your make file. This was done before your IDES. How all the various linked in libraries were brought in and the dependencies between them. All the libraries are linked by a linker, loader, and a compiler. So anyway, the point is that the standard C library is linked in with your code, contains your real system call which has your printf. But the native disk write, your secondary write operation is supported by your system call. They all will have different syntax for the write operation. So as we said before, we can have successful printing, you're not aware of all that.

How do you pass parameters into a syscall. At the higher level, you pass in parameters like strings. But not at the low level, as it has to be standardized and frugal in some sense. Needs to localize the lower level. There are three ways of passing parameters to the operating system. First is through registers. If you don't have many parameters, you can pass parameters to the registers. But then there could be more parameters than registers. You can push the parameters onto the stack by the program and popped off the stack by OS.

You have a system stack and you push the parameters on system stack. Then you have parameters stored in block, or table in memory, and address of block are passed as a parameter in a register. It may have a thousand bytes, so those thousand bytes are in the address space of the object. The kernel can read from the address space. Block and stack methods do not limit the number or length of parameters being passed.

This is a pictorial view. This is the user program, this is system call 13, so this is actually the code in the OS that supports system call 13. You have a table of system calls, just as you saw before. So this is load memory, somewhere here, there is an interrupt service or system vector, which is a system call table, and you have number 13 refers to a handler for that system call. There is a parameter passed through the register, and that is X. That gets passed here, but note that this X is a pointer to some other place, which is a process access space where the data is stored.

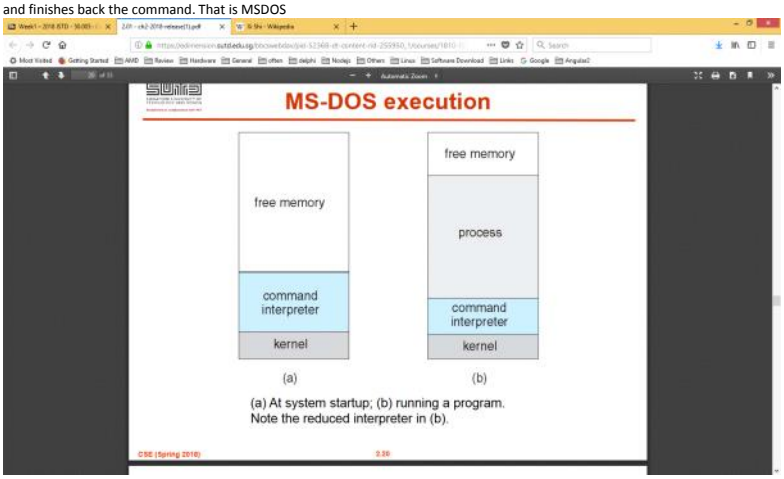
In general, most system calls can be classified to Process control (starting and ending processes), file management, device management, information maintenance, communications, protection. If I start LS, to start a process, and end that process, this will all be system calls. If it is successful in generating contents of directory, I need to terminate the process, and sometimes if it is taking too long, Then what should I do? I should temporarily halt the process and let some other process have the CPU. Pause the process and a simple thing like process control can have a lot of complexity. To start and end the process, I need to be able to pause it and to zoom it.

Second class of management is the secondary store. It is the only non volatile part of our system. It gets lost as long as you switch off the system. You need to manage the file system. There is a special way in which it is structured and managed through file names and directories/folders. File management has to do with system calls because file management creates new files, delete files like someone says. We have to permit, read write execute bits, which has to do with file management and done by the kernel. So the end user has the permission to change the RWX but the OS will do it on his behalf. Every operation on files has to go through the operating system and makes use of system calls.

Device management. Management of your keyboard, mouse and printers. All of these have to be invoked through system calls. There are people who ear making device drivers because there are so many architectures, you have to write the device drivers for each of them. Device drivers are part of the operating system.

Information maintenance: time of the day, IO.
Communication: shared memory, message passing. Encompasses a lot. If sending an email across, you are doing message passing. Your operating system is taking a huge piece of info, packaging it to some message and sending it to a TCP/IP socket, go there, and reverse process takes place. Just refer to it as message passing, which is one of the two ways of interprocess communication. Shared memory is used when two processes on the same host are communicating. In message passing, you have the kernel, which will have a buffer on top of a kernel, so you will write the information into the buffer. Then P1 writes into the buffer, while P2 reads the buffer into the address space. P1 informs the kernel to pass message to P2, and it is important that P1 should not mess with the address space of P2. Kernel writes message into buffer, wakes up P2, and then sends the message that P2 should read it into their own buffer. There are some system calls which are shared memory attached. P1 and P2 agree to share a piece of memory.

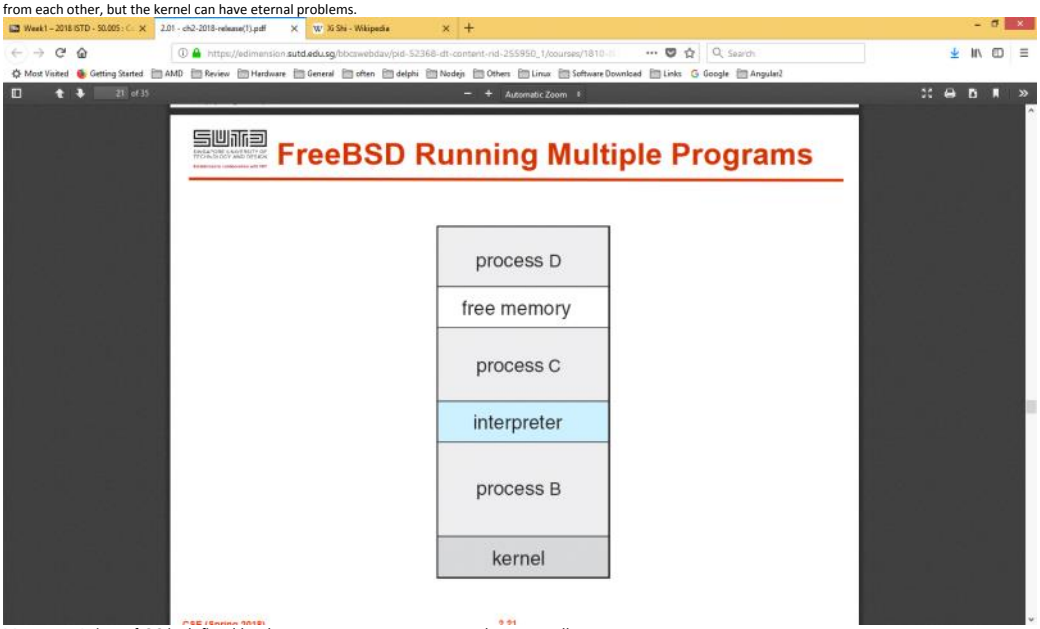
This is an illustration of unix system calls. But of course they are identical. The command interpreter, was the only thing in memory. When there's a process to be executed, it overloads itself. The command interpreter loads into the memory, overwriting itself. The process continues to execute with some leftover free memory. Once process finishes, the first thing is that the command interpreter needs to load back itself. The command interpreter finishes and brings back the rest of the command interpreter and finishes back the command. That is MSDOS



BSD had a kernel. It was a multitasker, and the interpreter and the processes have their own address space in memory. The problem with this is that this kernel has to do everything. It's a monolithic kernel, has to do the low level process management, the handling of the disk and devices. All of those things with the services and the operating systems. A monolithic kernel. It's a security protection issue,

something can go wrong, and contaminate any part of the operating system. Say there is a disk driver and process scheduler. Once they interfere, or even the memory manager, the hard disk would be contaminated. So not a good idea to have a memory system like the BSD.

The one thing you have over the MSDOS. One is of course multitasking, and the other one is protection. Process B and D cannot overwrite each other's address space. That way, the processes are protected from each other, but the kernel can have eternal problems.



Most users' view of OS is defined by the system programs, not actual system calls. Browser is the application program. Even a web browser is a system program, because you don't sit down and write web browsers. Most users' view of OS is defined by the system programs, not actual system calls. You use the system program offered by the operating system. So you don't get to see the system call underneath that. You don't see the system call beneath the browser.

System programs provide a convenient environment for program developments and execution. Some of them are system programs sometimes called utilities. Configuration: you want to set some kind of system permissions. Editors, programming language support, program loading and execution. In a sense, an ordinary user that writes his software, is an application programmer.

The whole area of operating systems is not very well defined. An operating system is nothing but a program. It is a sequence of instructions. Need to be careful on how you think of the operating system. Some approaches have to be successful. You can have an operating system, for say, a real time system. RTOS, things with a sharp deadline. Just the way of thinking about operating systems. An operating system takes commands on behalf of the user and schedules them for operation. Manufacturing can have assembly line or job shot.

In an operating system, every input that comes in is special. You have a request. And once you received that request, you start a process. Or a sandwich. And the chef has to start preparing the sandwich. Something needs to be set into process. Some machine needs to be set in the background. That's similar to a process execution. You can have concurrent execution as well. An operating system takes in orders and issues orders for processing. These processes all terminate in the end. So sometimes, you have this requirement of real time.

Many embedded systems have real time. Let's say real time tracking, in a mission critical application, the operating system needs to be meeting a real time deadline. If you miss that real time deadline, it can be a catastrophic failure of operating system. The job must be brought in for execution before the timed deadline runs out. We talk about this area of user goals vs system goals.

User goals should be that OS is convenient to use, easy to learn, and system goals should be fairness, throughput, have enough time on the computer. Every process should get its own share of the CPU. Shouldn't have someone who is just waiting for a network packet to come in. That's all a matter of designing an OS correctly.

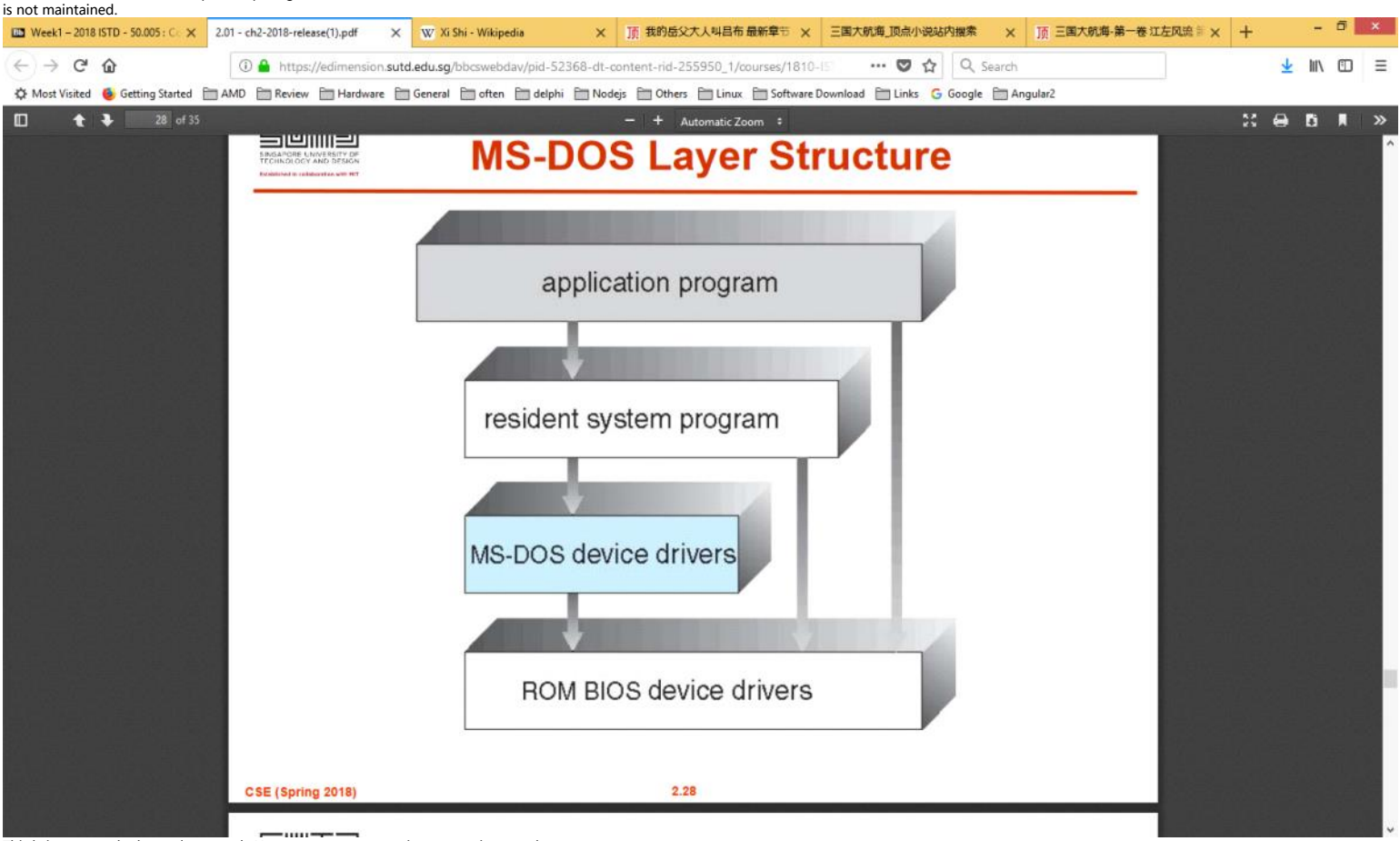
For example, in a government scenario: the government has certain mechanisms: pay your income tax. If you have a real time operating system, every process has to be done within twenty seconds. I need to set that timer interrupt, well in advance of the deadline, so that the task has been serviced. This task has been serviced before the deadline. The policy is one second, twenty miliseconds for this service task. We have read write execution bits for a file to enforce access to it, policy is who gets the ability to read to a particular file.

Grant: Am I allowed to grant access to others. Some operating systems do enforce such policies. That will be an add on the operating system. I have the ability to grant read access but not write or execute access. The policy would be that grant would not be given to most users.

Second example: I have a mechanism with a periodic timer interrupt, gives me clock ticks to allow rescheduling of CPU to different ready processes. After a certain amount of time, this process leaves the system and some other process is brought in. Allow a process to run for how many ticks before we switch it out and switch in another process.

Back to MSDOS. Didn't have any of these mechanisms or policies, doesn't even support dual mode operation. User and kernel. DOS didn't have that because it is a single user system. Didn't make any difference between user and kernel. And maybe that's okay

because it isn't meant for multiple computing. It did have some kind of a structure but it is not maintained.



This brings us to the layered approach. You structure an operating system in a number of layers. Ensure that the software at one layer is working properly before bringing in software at a higher level. If I were to write a program. Something happens and my program doesn't work, I have to debug a program. Maybe the compiler has a problem.

Or do I have to look into the hardware. Most of the time that's not the case. When my program doesn't work because there's a bug in my program which I am responsible for. It's an assurance; because people have gone to great lengths to ensure that the hardware is working properly. Only then would the compiler be offered to end users to write software. Your android, IDE have been tested and then you can write your programs easily.

Most of the heavy weight operating system has been moved. Only the user executing that command would be useful. Since the error is restricted to that user process, the Mach has moved (microkernel) the extra operating systems, it does not mean the user can execute them. Some people might think that the user can access all of those things. The user can go through the operating system call. All the functions of that call can be processed within the user of that address space. Benefits? It's easier to extend a microkernel, can port OS to new Architectures. Because only the various essentials are in the market kernels. How to allocate new processes. Even the swapping is out in the user space.

Detriments: There's a lot of communication between user and kernel space, including copying as kernel data structures cannot reside without kernel.

Cohort 2

Wednesday, January 31, 2018 11:32 AM

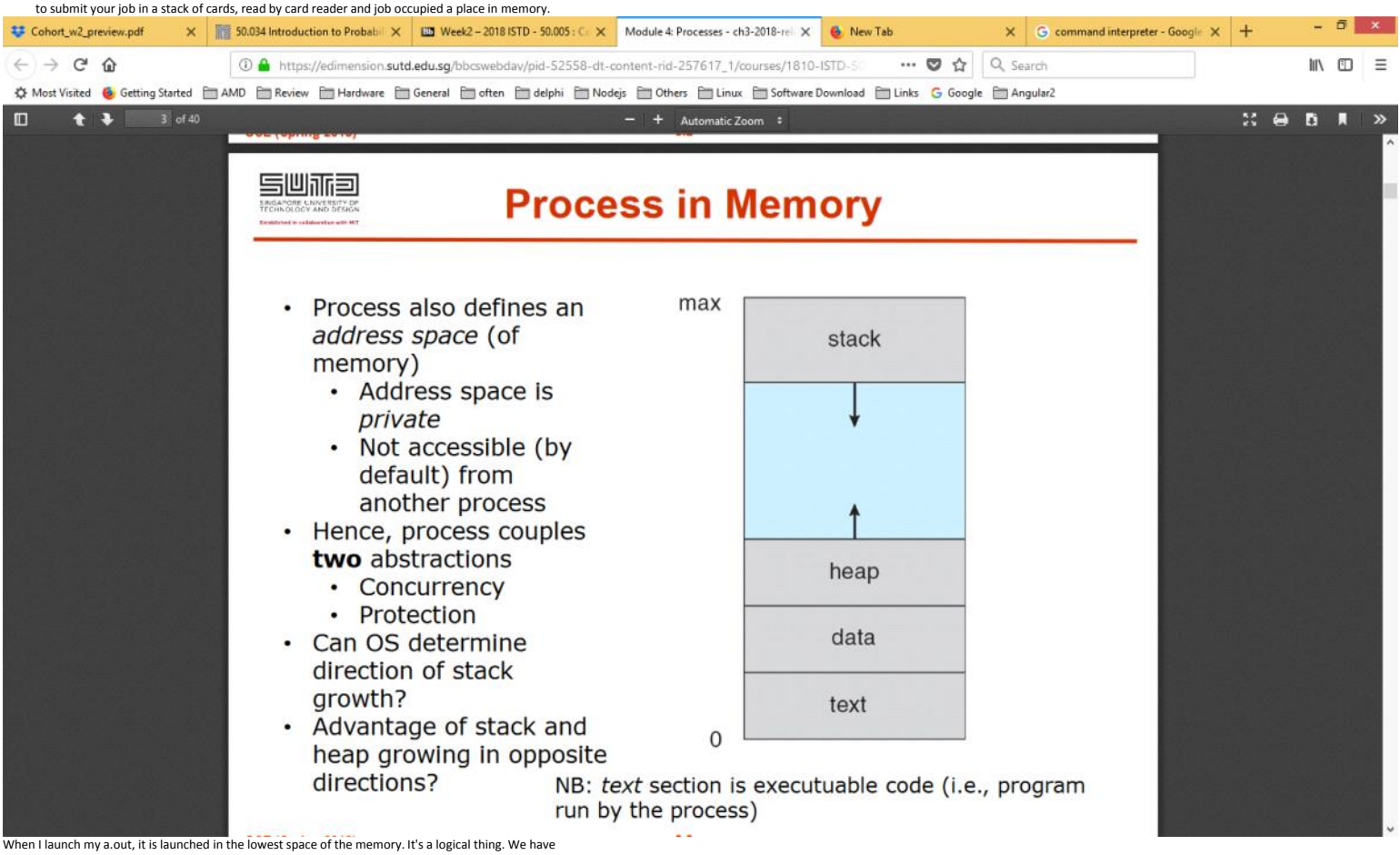
What is the purpose of a system call? To allow users to make requests to the operating systems. What is a command interpreter? It's a program that inputs user's commands. It is not part of the kernel, because you want to customize your interpreter. Used to be part of the kernel. What system calls have to be executed by a command interpreter or shell in order to start a new process? Fork. What is the purpose of system programs? Hard to interpret. Those that are required by everyone. But in application programs, there is a loose way of thinking about it. There is no application vs system programs.

Why do some system store the operating system in firmware? Pic controller. Smartphones.

We are only concerned with two things: managing processes and memory. We look at a process as being a program in execution. Defined as a locus of execution of a program. Something you can write. It's a print out of your program. Process is dynamic: state of execution of a program. Process could be one of several states. Slash bin slash ls, it is an executable file, resides in secondary storage in a file system. It is not run until I launch it with an ls command. Can I issue more than one ls command at the same time? If two users are logged into the same machine, they can each launch their process at the same time. They have each of their copies running at the same time.

A process has certain state information. What kind of information do we need to store to make sure that these processes are executing concurrently?

> We need to store the program counter, the stack pointer and all of the stack when you store the elements of a stack. The stack wasn't deep just now, no function calls or recursion. It was just one level deep. In our normal assignments, we write functions that invoke other functions, invoked recursively. Especially when you go to deep learning and other data mining operations. The Stack is going to be quite deep. Stack is important in a programming language. Also need the data section. Data section holds global variables, seen by all parts of the program. We can change program variable, but generally do not. Dynamically allocated memory, in heap. When I want to allocate temporary data structures, with an algorithm that uses a lot of memory. You would like to allocate temporarily some nodes from a heap, and store it in a heap.
>
> When you program in C, you set up data structures and set up your own garbage collection. After you finish that piece of memory, you free that memory with a free instruction. In more advanced programming languages, the garbage collection is handled by the JVM. Java makes strong typing so that the language itself would know when the particular block of memory is no longer in used and can be released. Garbage collection is done automatically done for you.
>
> The terms jobs and processes are used synonymously these days. Jobs are used initially, you didn't have time sharing initially on the OS. The way CPU was shared was by submitting a job. You have to submit your job in a stack of cards, read by card reader and job occupied a place in memory.



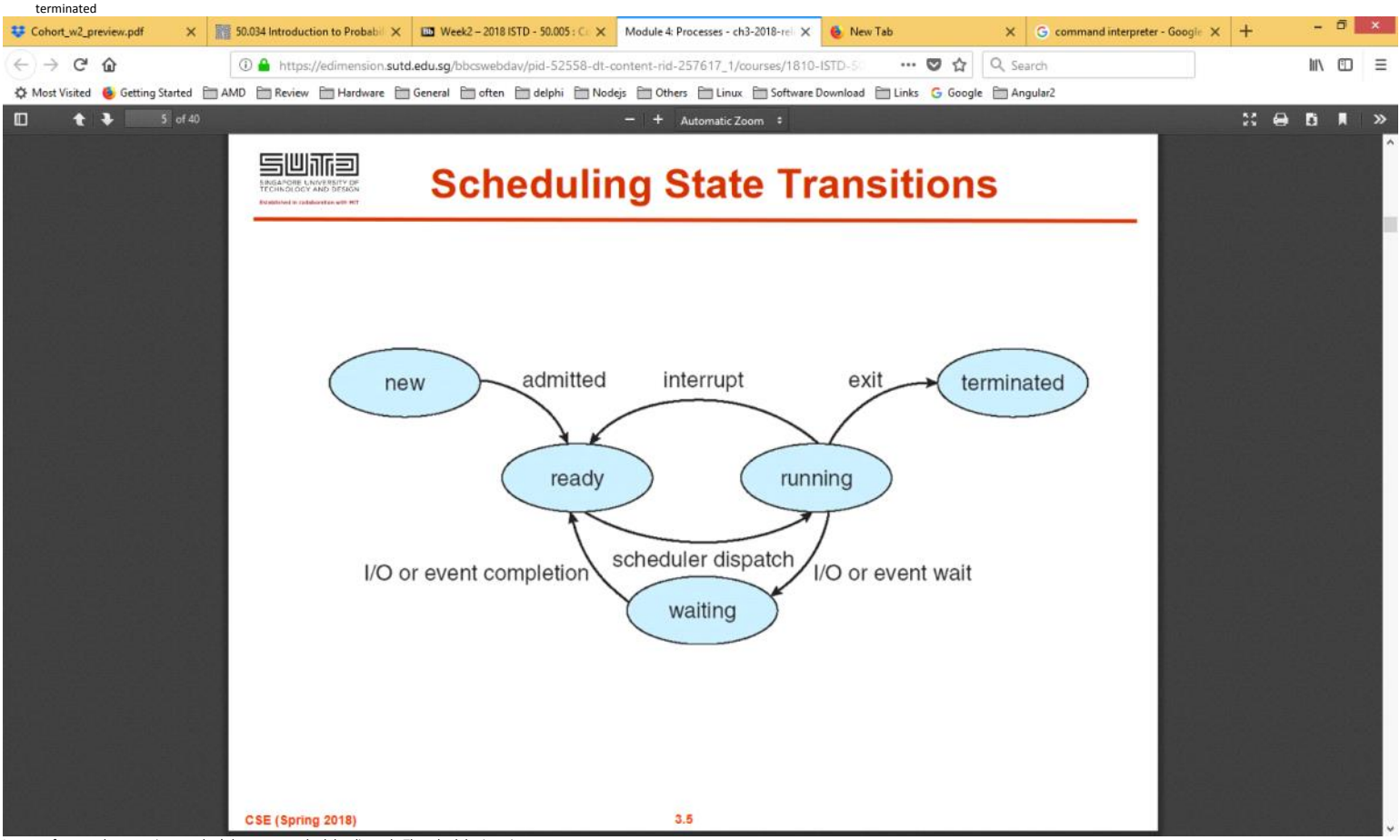
When I launch my a.out, it is launched in the lowest space of the memory. It's a logical thing. We have the global data, and we have the stack, growing as the stack grows, the stack would move downwards. The pointer would move up. Can you think of a reason why it would be this way? It gives you flexibility. I can have a program with a lot of heap memory but stack is not very deep. I can have a program with not a lot of memory allocation, but a lot of searching like a chess playing program.

Processes are insulated from each other. Gives you concurrency. As long as I have a process state stored separately, I can switch between processes. I can also protect one process from another process. Memory protection has gone down to the level of hardware through the use of special registers. It has a base register and a limit register. This would tell you where this zero is and where the maximum is. Any memory process does not exceed the bound of those two registers.

Process scheduling State.

As a process executes, it changes scheduling state. There is this distinction of being created. You have to do a number of things. There is room in the memory to take yet another process. If that's an issue, can write a recursive program that writes a lot of processes, run a lot of fork, table explodes.

New: process being created
Running: assigned the CPU; instructions being executed
Waiting: process waiting for some event to occur
Ready
terminated



It moves from ready to running, to what's known as a scheduler dispatch. The scheduler is an important part of the OS; first part of the execution process. Moves from ready to running. What can happen if it's running? It can try to get data from a keyboard. Or get data from a file system. Whenever there's something like that happening. An input output request takes place. The IO is an incredibly slow device or activity as compared to the speed of your execution. So this process which is running needs to be pulled out from running stage to the waiting stage. Once it is completed. This transition takes place from ready to waiting, because there are many processes also ready for running.

Maybe the process is not waiting on any IO. It can't keep on running, it will hog all the resources. I need to have a timer. If I don't allocate, the Cpu has to do it on my behalf. A timer starts at the same time. It could be ten, twenty miliseconds. When that time slice runs out, there would be an interrupt generated by the timer, forcing me to go back to the ready state.

It's not existing at all, and when it comes to being, it is being admitted, and moving to the ready queue, to the running state. There is only one process running at any given time. We have a single CPU, with only one process at that state.

The kernel needs to maintain some housekeeping information, like student records, phone numbers, ID. House keeping information in an OS. It's a data structure that consists of all the information necessary. It's in a ready state, There are systems in a higher priority than other systems. Scheduler is an important process, should be given a higher priority. Which process has what priority is what is known as scheduling information. Memory-management information: Do I give every process 1 MB of address space? These things set by admin user when booted. Accounting information: This process has been executed for so long, done what on the disk. Especially mainframe and backend systems. They bill you based on all of these things. It can move from a running state to a waiting state, and finish all the IO requests. Or it can move from a waiting state, and in order to move to a ready state, it has all of its IO requests completed.

Process Control Block (PCB)

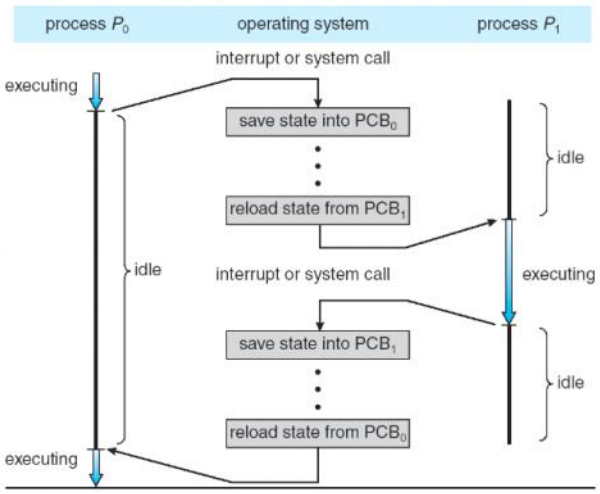
This is a data structure that stores all the information. It stores all the process memory in RAM.

This brings us to the notion of a context switch:

Cohort_w2_preview.pdf | 50.034 Introduction to Probabil | Upload Assignment: Lab_1 Sub | Module 4: Processes - ch3-2018-rel | New Tab | command interpreter - Google
https://edimension.sutd.edu.sg/bbcswebdav/pid-52558-dt-content-rid-257617_1/courses/1810-ISTD-5
Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2

CPU Switch From Process to Process

- Processes run independently (logically separate) and concurrently
- *Context switch*: change CPU from running one process to running another
- For a uniprocessor system, concurrency is an illusion
  - *As if* each process has its own CPU
  - Can view it as *virtual* CPU

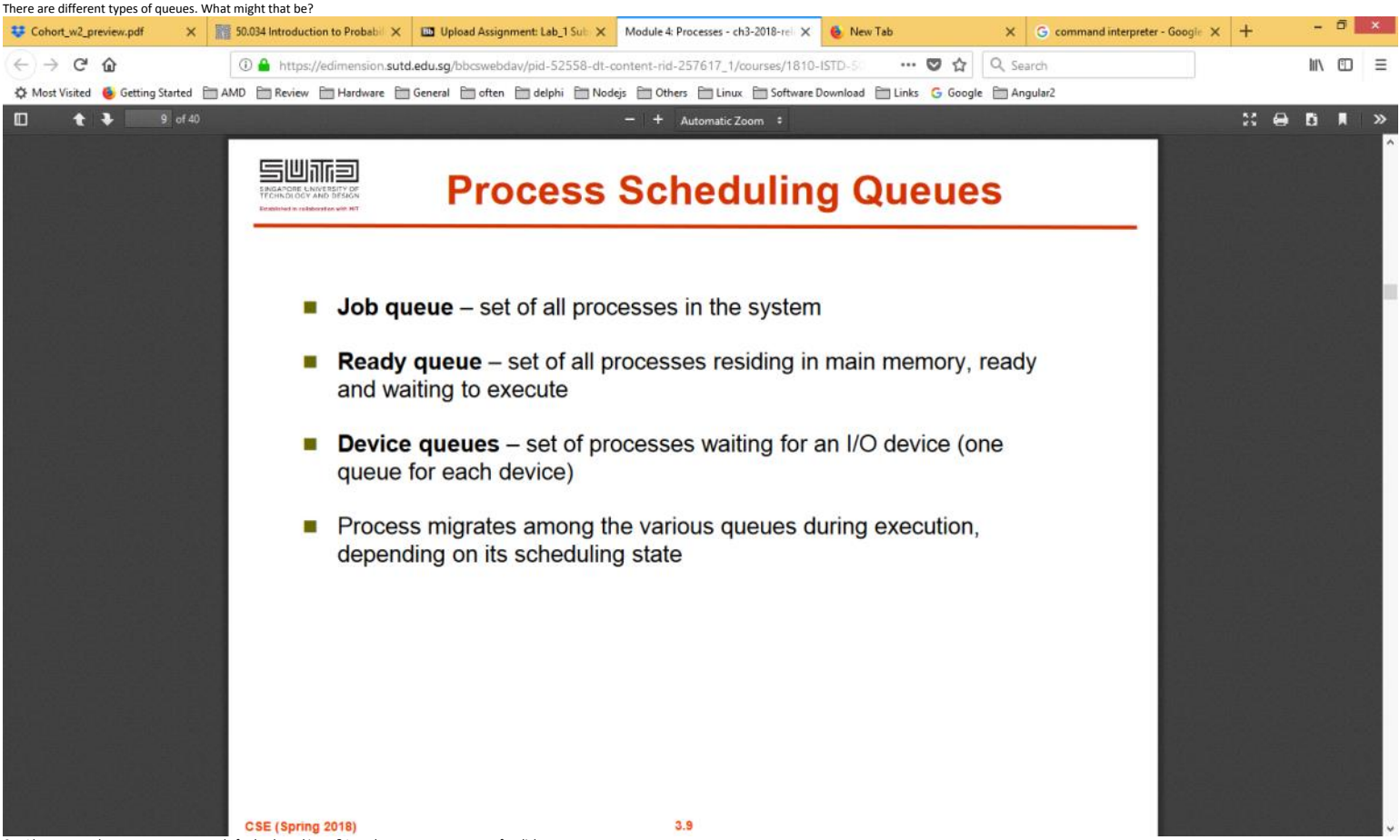


What is a context switch? The status of all this information, in the process control block. This word save, save to RAM or secondary storage? If at that point in time, I only have two processes in the system, if I have enough memory, there's no reason to put it on secondary store. We are done with saving the

state, it may take a long time. Once saving the state, we need to reload state from PCB 1. This assumes that process 1 is in the ready state.

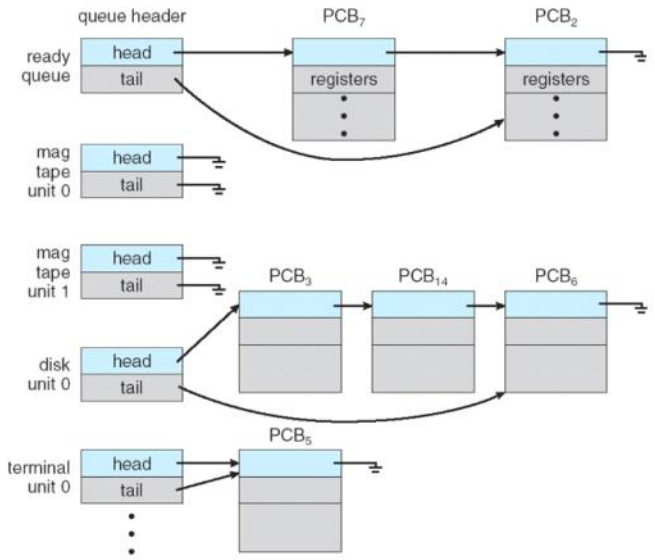
Reload from process 1, and then P1 is executed. Then there is an interrupt of a system call for P1. Save the execution of the control block for P1 and then reload the information from PCB 0 into the various registers, and then you are ready to run. This is a very fine grained thing.

There are different types of queues. What might that be?

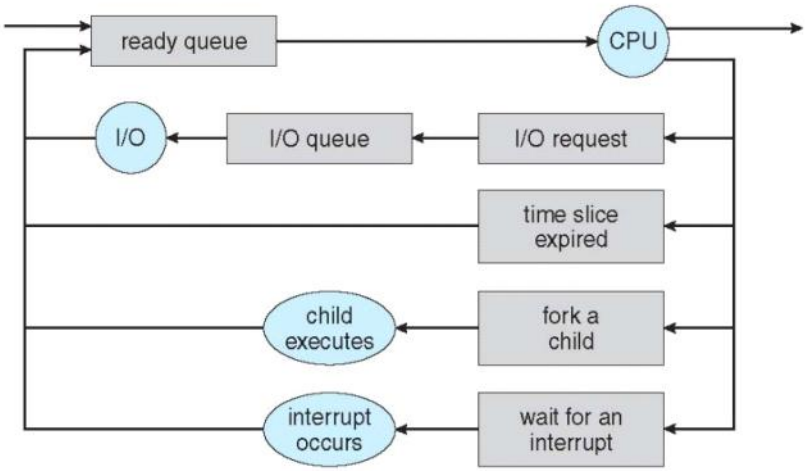


Can I have more than one processes ready for keyboard input? I can have many concurrent for disk or keyboard input. Each of the IO devices could have their own queue. Who should manage is a separate question.

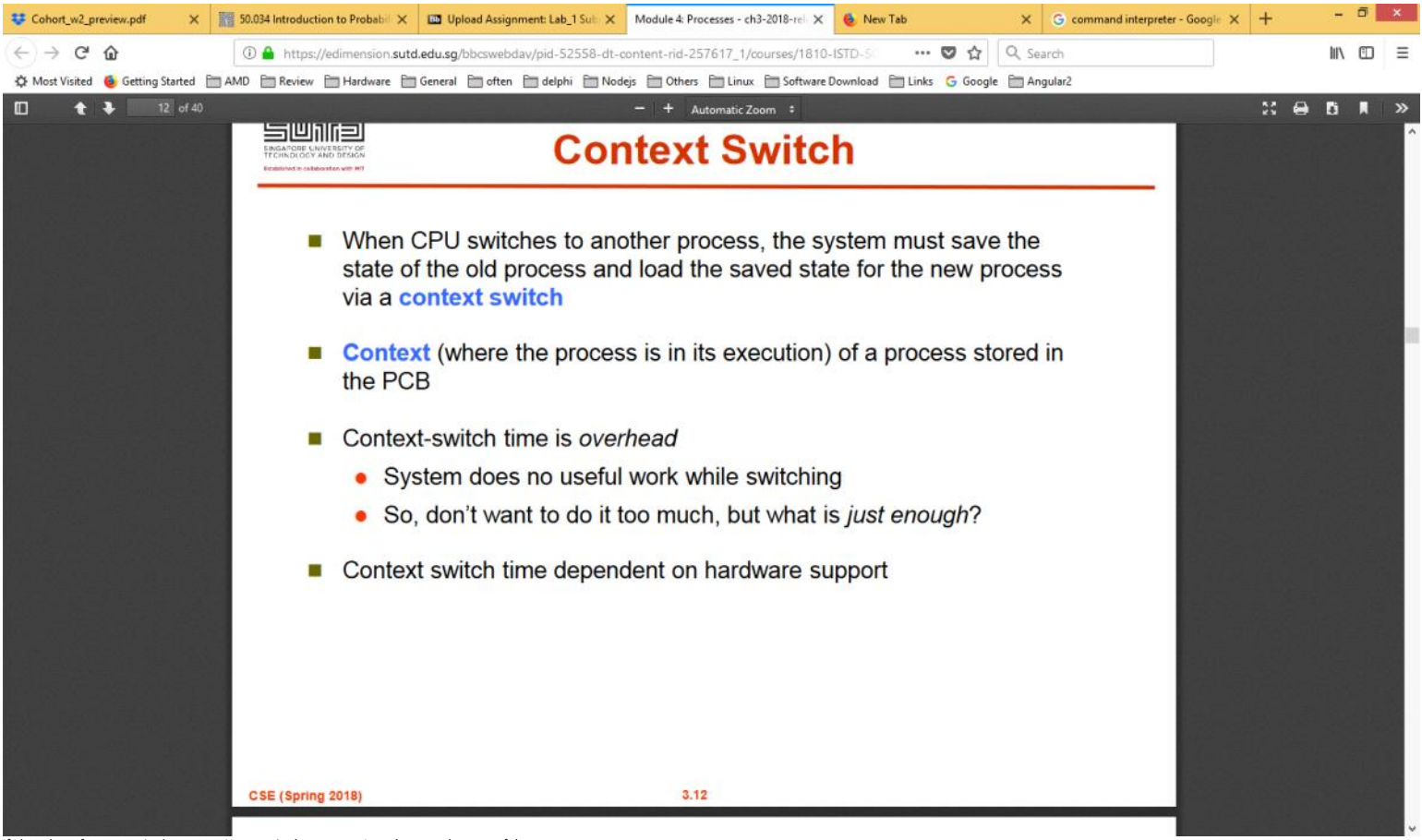
Ready Queue And Various I/O Device Queues



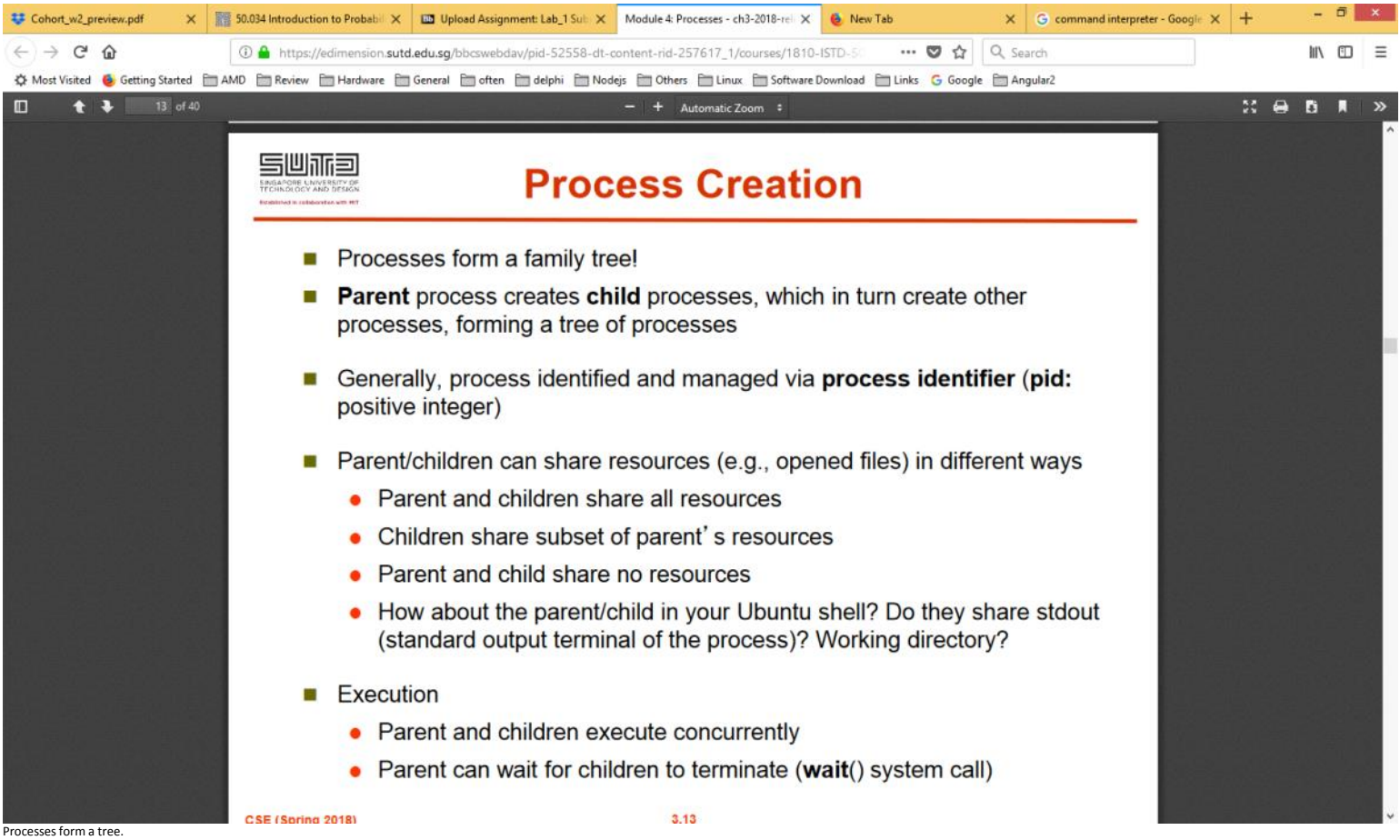
Representation of Process Scheduling



Note: create a process is also called *fork* (created process is *child* of creating process)



If I have lots of processes in the system. No room in the memory, I need to save the state of the process to the secondary process. It's the entire address space. Those parts of memory that has been used, those all have to be saved on the hard Disk. To switch from one process to another process, we call the context switching overhead. Does the government have overhead? By all means. Context switch does happen, and SOLARIS have put in additional hardware to make context switch faster. Can have six to ten types of registers. I can load the state into the additional registers. When it comes for the process to run, just switch to another set of registers.



Processes form a tree.

When process creates a new process using fork() operation, which of the following states is shared between the parent process and the child process?

D. Shared memory segment.

Including the parent process, how many processes are created by the following program.

3 forks running in the main C program

Cohort 1

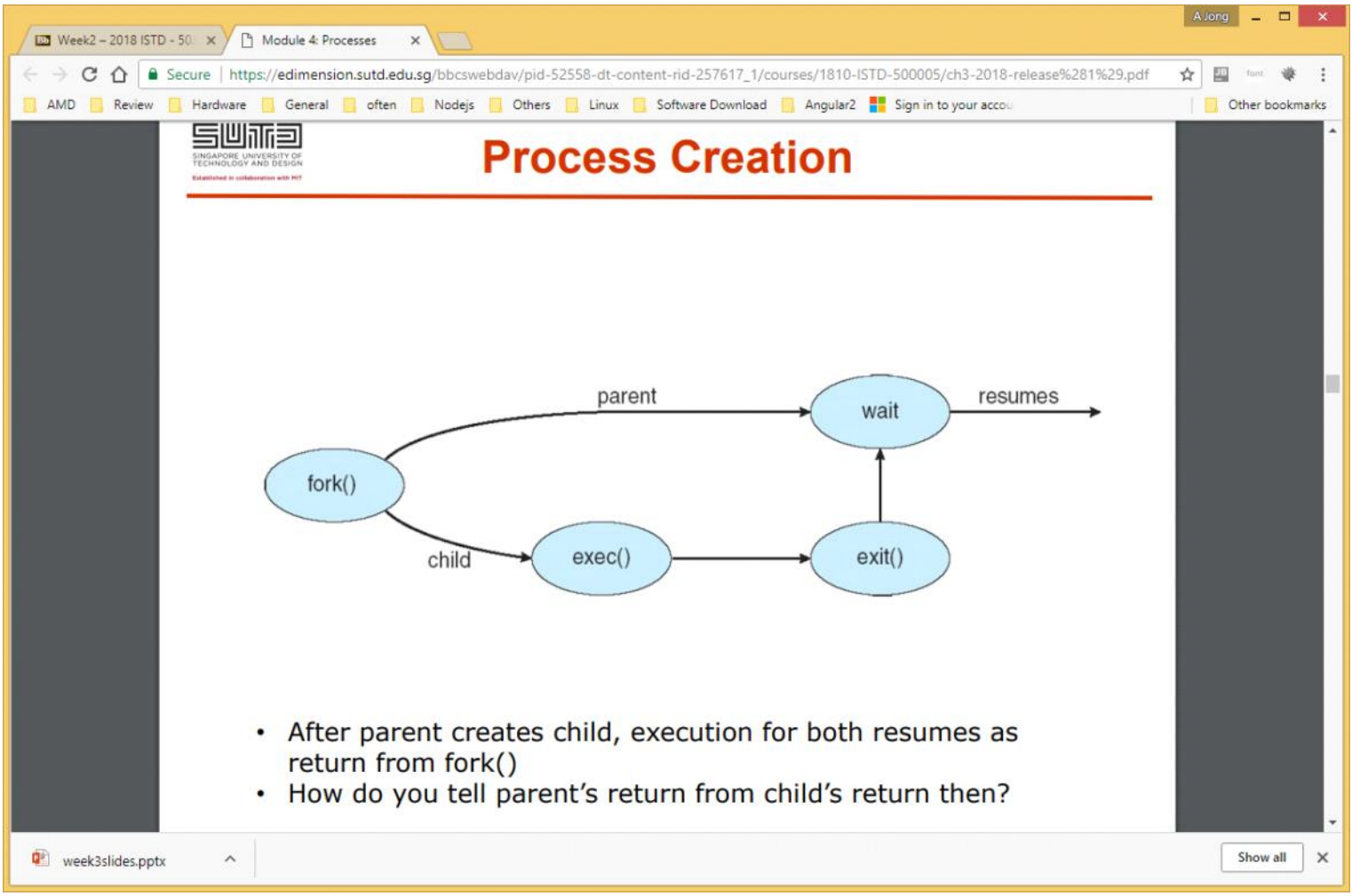
Monday, February 5, 2018 8:33 AM

Process couples two abstractions: concurrency and protection.
Where does the kernel maintain information about each process? Process Control Block. Keeps a lot of info there. In a kernel table. It is not stored in user space but in kernel. The following are the only queues a process can be in Ready and in Waiting. True or False? There's a device queue, and IO queue, and multiple instances of devices where you have separate queues.

Two mechanisms for IPC(Inter-Process Communication): Shared memory and Message Passing. The function fork is that it creates a new process address space. Everything that is replicated here onto this address space. The process ID of the child will be 0. But the parent doesn't keep its own id. Instead, it would keep the child ID. So the code is shared, but value of PID is different. Parent would have an ID > 0, but not 0. What is the value over here? The parent is waiting for the child to finish. There is an implicit where the signal would go to the parent. The parent would print the value of value, which is 5. The same print statement, what would would that be? It would be 20. It would take the old value of 5, add 15 to it. So the Child would have the value be 20.

Parent gets the PID of the child when fork() is called. So in this case, the first line © prints out 2734, the pid of the child. While pid1, having called pid1() prints out 2700

Parent can children execute concurrently. The parent can wait for child to terminate. So it's quite powerful (wait() system call).
Address space: Child gets its own address space. Whose content is initially a duplicate of parent's (including text section that stores the program run). Child then loads a new program into its address space. Shell process creates child process, but child wants to run (say) ls, then it will load ls code



Process Termination: Handled in different ways in different unix systems. Different versions of Linux handle it differently. When the parent process creates a number of children, it doesn't necessarily terminate all the children processes.

In unix, the way processes are structured, job referred to as the whole number of processes created as a process. So a csh that creates ls and cat that originates from csh (pid 1400) is a process. It's a related group of processes. It is not synonymous to the previous use of the word job.

Interprocess communication. How might a process communicate with another process? One process opens a file and another process re-opens a file. We are going through the file system. Open a file, close a file. If you have a protocol, and you have the overhead of going to the file system. Secondary storage is about a thousand times slower than main memory. The whole idea of message passing, whether sending an email, you create a pipe, a channel of communication, where you pass information from one

process to another. There are reasons for cooperating processes:
Share information, speed up computation, achieve modularity (hence protection) in spite of cooperation. Convenience e.g. "ls -l | wc -l" roughly counts how many files you have. There are two basic means of IPC: Shared memory and message passing.

In the case of a host computer. Both of the processes are on the same host. So how would they communicate if they were to use message passing. Process A wants to send a message M to Process B. That is done by means of a buffer. Then it informs the kernel, take the message and give it to process B. Then the kernel would take that, wake the process B and say that here's a message from Process A? So the kernel sets aside some memory from Process A into its own kernel data structures and copy it into Process B space. Then it will wake up process B. The key idea is the copying of message from memory to memory.

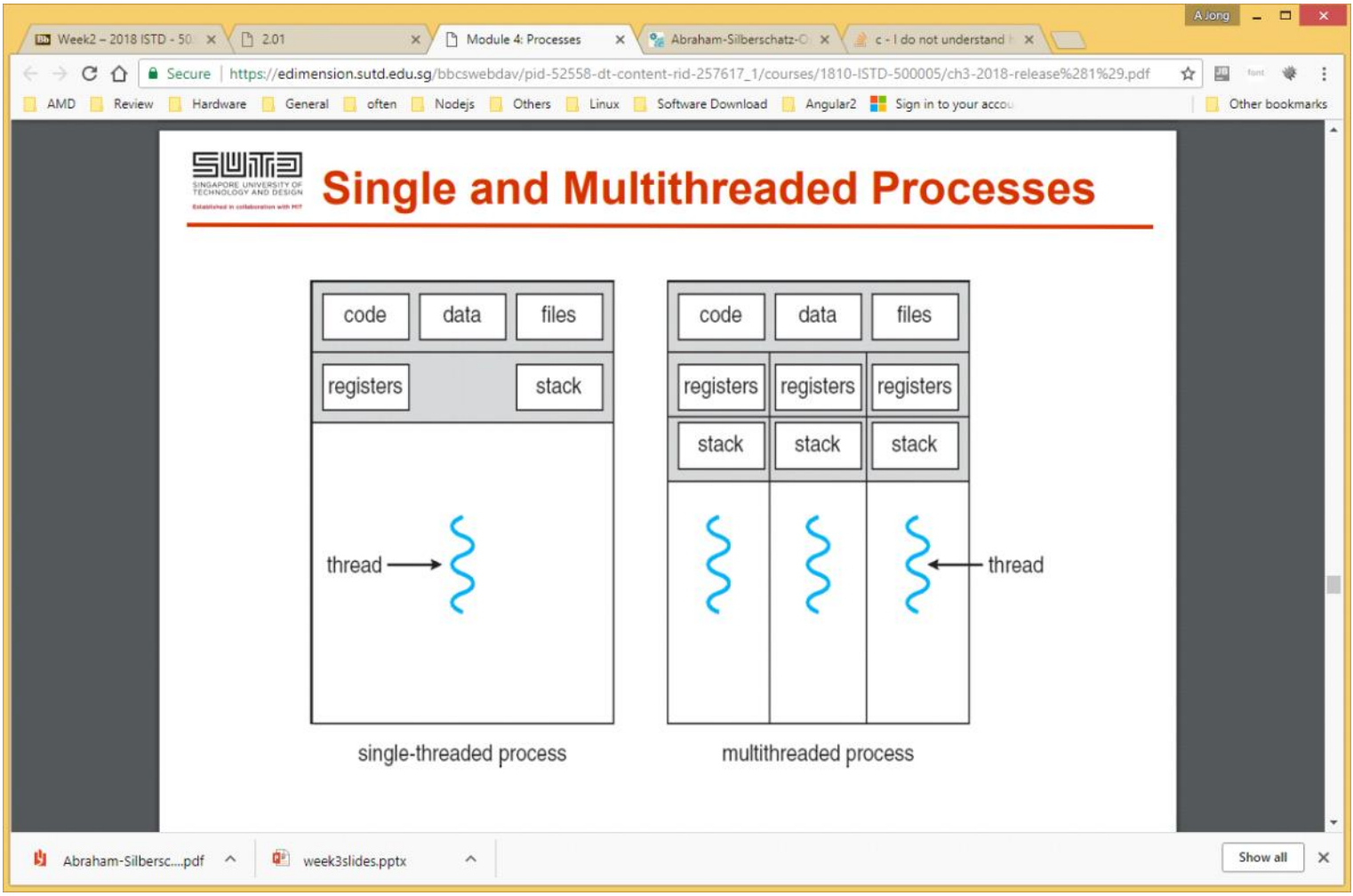
The system call would send the message, and process B would do a system call to receive the message. And the message passing would take place this way.
Shared Memory
In a sense: proces B needs to know that there is a piece of shared memory there. There has to be another channel where process B must be informed that the shared memory segment such and such. Process A can create that shared memory segment but needs some way to tell process B that there is a shared memory. Once these two processes are attached, they can read and write like from their own memory. You don't have to copy the piece of memory in order to communicate. As long as they don't interfere with each other.

Shared Memory: Like a file descriptor. Segment ID = shmget(IPC_PRIVATE, size, s_irusr | S_IWUSR); The process first creates a shared memory segment (w/ read, write access for owner). P1 gets the shared memory
Shmat shared memory attach: Process wanting access to that shared memory must attach to it. Once P2 has the ID, then P1 and P2 can get shared memory access and would read and write from the memory. The two processes have to agree on a way to share this ID?
Shared_memory = (char*) shmat(id, NULL, 0);
How would process 2 know that this is the ID? This is outside the system. I would write it into a file, which is known to P2, read the shared segment ID, and use this system call to get access. It is outside of that system. They could just write this shared memory like writing to a disk.

A socket is defined as an endpoint for communication: Usually for network communication, but also IPC within single machine (e.g. so called Unix domain sockets)
Endpoint specified as concatenation of IP address and (TCP or UDP) port: E.g. 161.25:19.8:1625 refers to port 1625 on host 161.25.19.8.

We saw that the process is a unit of concurrency and protection. But what is the drawback? You incur a lot of overhead. You have to make sure that your address space is unique, don't step on each other. Every time you do a context switch, you have to do copying, and you have to swap out the disk and so forth. When you do a fork, we've been saying that you copy the entire memory content from the parent to the child. While that's logically true, physically, no. It doesn't make sense to copy the code and wipe it out immediately. They do copy and write. Only when that page is written into do you need to write into it.

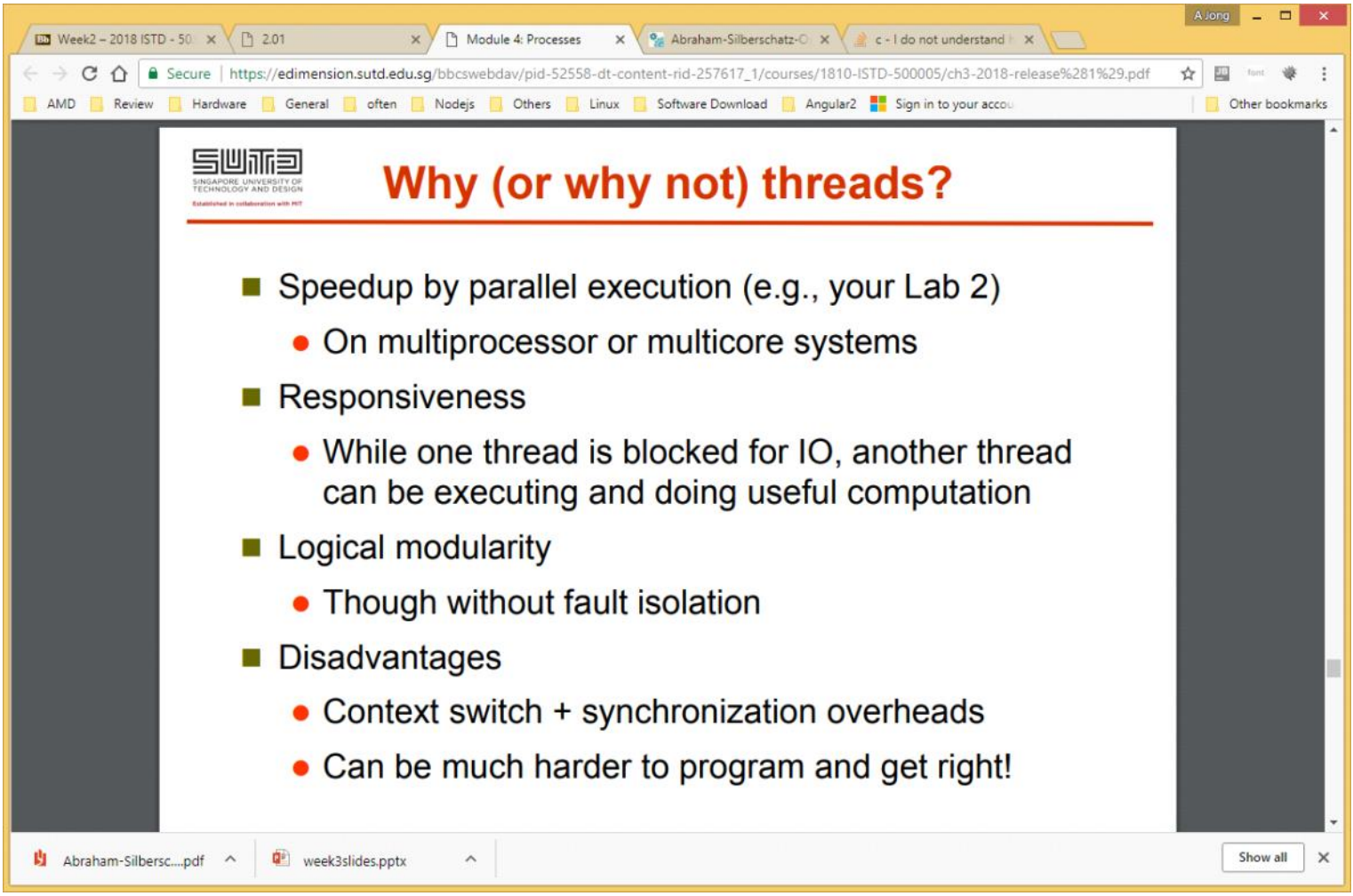
Can I decoupe the two: e.g. have concurrency without protection? Yes, use threads: (thread = line of execution), has registers + stack: Many threads can run within a process, share the process's address space. No protection between them. But IPC (or inter-thread communication) is simple (e.g. no need for shmget, shmat, etc and fast (much less to save/restore at context switch)

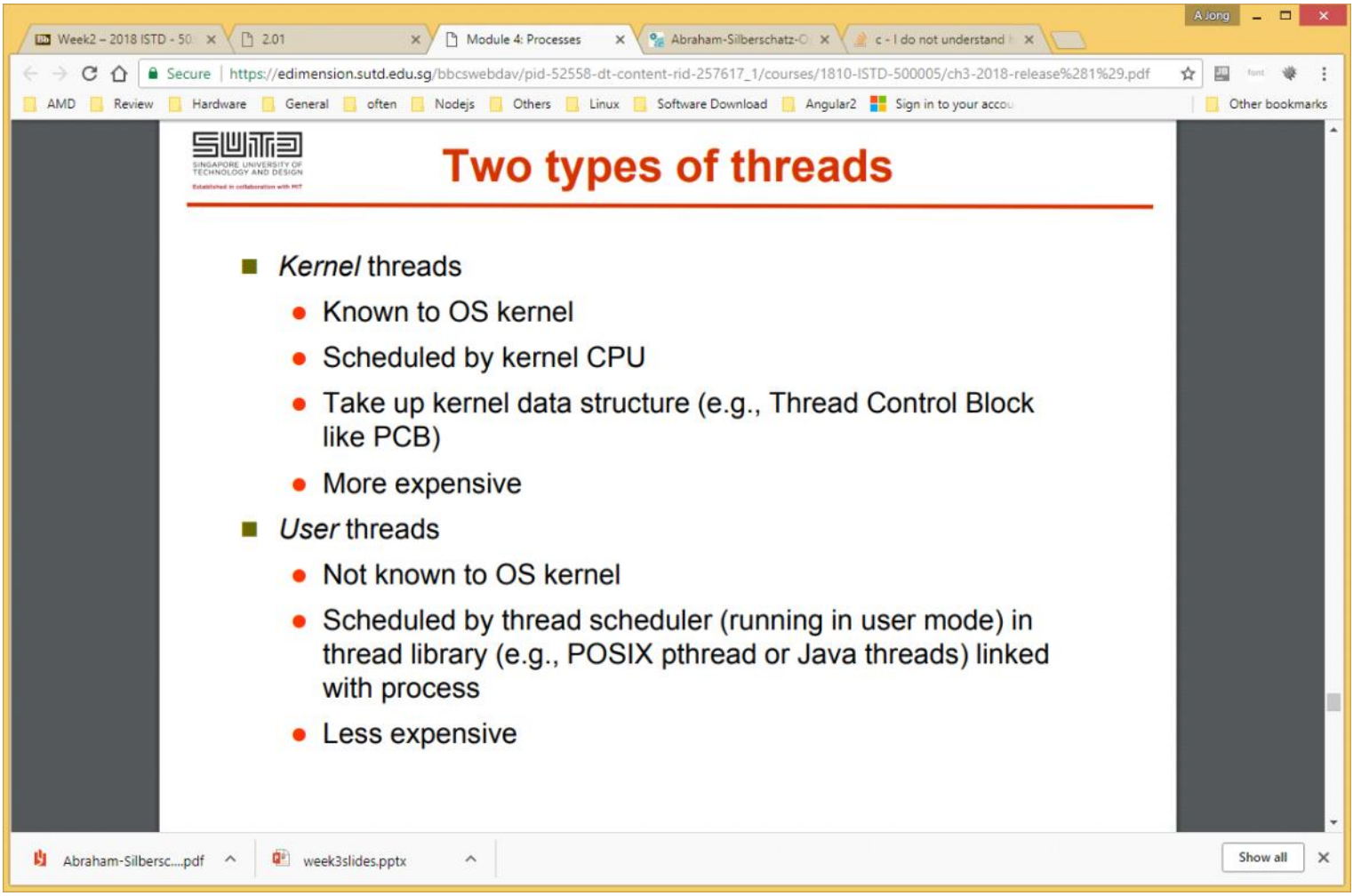


A thread of computation is a sequence of computations that come together. You have multiple sequences that are coming to their respective CPUs. In the case of the web browsers, one thread could be loading the image to one part of the memory. The other thread could be opening up a network to connection to some other website. All these threads would go on in parallel in another process.

Java has a good set of memory for handling threads. The whole Java model is not geared towards shared memory. There are three types of system calls: Windows 32, Linux, and Java. Linux standard adopts the POSiX standard. API specifies behavior of the thread library, implementation is up to developers of the library. Common in UNIX operating systems(Solaris, Linux, Mac OS/X).

Why (or why not threads?)
Speedup by parallel execution (e.g. your Lab2) On multiprocess or multicore systems: Matrix can partition data set very well. With multiple CPUs doing different quadrants. They each write into their own piece of the memory. Responsiveness: While one thread is blocked for IO, another thread can be executed and doing useful computation.


Week2 – 2018 ISTD - 50
2.01
Module 4: Processes
Abraham-Silberschatz-O
c - I do not understand
A Jong
Secure
https://edimension.sutd.edu.sg/bbcswebdav/pid-52558-dt-content-rid-257617_1/courses/1810-ISTD-500005/ch3-2018-release%281%29.pdf
AMD
Review
Hardware
General
often
Nodejs
Others
Linux
Software Download
Angular2
Sign in to your accou
Other bookmarks
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
Why (or why not) threads?
Speedup by parallel execution (e.g., your Lab 2)
On multiprocessor or multicore systems
Responsiveness
While one thread is blocked for IO, another thread can be executing and doing useful computation
Logical modularity
Though without fault isolation
Disadvantages
Context switch + synchronization overheads
Can be much harder to program and get right!
Abraham-Silbersc....pdf
week3slides.pptx
Show all


Week2 – 2018 ISTD - 50
2.01
Module 4: Processes
Abraham-Silberschatz-O
c - I do not understand
Secure
https://edimension.sutd.edu.sg/bbcswebdav/pid-52558-dt-content-rid-257617_1/courses/1810-ISTD-500005/ch3-2018-release%281%29.pdf
AMD
Review
Hardware
General
often
Nodejs
Others
Linux
Software Download
Angular2
Sign in to your accou
Other bookmarks
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN
Two types of threads
Kernel threads
Known to OS kernel
Scheduled by kernel CPU
Take up kernel data structure (e.g., Thread Control Block like PCB)
More expensive
User threads
Not known to OS kernel
Scheduled by thread scheduler (running in user mode) in thread library (e.g., POSIX pthread or Java threads) linked with process
Less expensive
Abraham-Silbersc....pdf
week3slides.pptx
Show all

Wednesday

Wednesday, February 7, 2018 11:33 AM

We are doing a fork over here. What do we know about fork? You have two processes. You have parent and child. Inside of this, we have a piece of code that only the child executes and only a piece of code that the parent executes. If you go and check to see the pid, if the value returned in the fork is zero, we are referring to the child. If the value > 0, it is the parent. If you draw a picture over here:
Child has pid = 0, parent has pid = 1234. But child has id of 1234. Parent has a wait null. Parent is waiting for the child to terminate. What does the child do? The child is doing some thread activity? The parent is not invoking any thread calls. When the parent prints out the value of this integer variable *value* ? Is it 5? The parent and child have two disjoint address space. The only thing that is different at point of fork is the pid, so the value would be the same. But there would be two different variables in memory. If you want to do shared memory, you need to get shared memory from the operating system called shmget that you have to attach using an shmattach (from parent to child) then you get a block of shared memory over here.

Who handles those calls? It is the java virtual machine. The java virtual machine handles the process under the OS. It's just a process under the host OS. All these things you see in the code are executed in the process. Here, it would be the run time library for C for C code. The only C code that is a system call is fork. But the threads come from a library here. The run time library: there's a run time system. Python uses the one for C. Java has a runtime system, called the Java Virtual Machine. This is all handled within the process.

Does the child thread executes joint or the parent executes joint. The child executes runner, and this piece of code changes the value of *value.* That's the only thing that the child thread executes, before exiting. It exits the thread infrastructure. In fact, the parent is waiting for this child to finish asynchronously, as if it is parallel. It is as though they were identical. They behave exactly the same way. The parents run in an address space, the child runs in a separate address space. The parent thread which happens to be a child itself, it runs this thing. This parent process can commence.

For the parent thread to print out the value of this variable, after creating the child, immediately after the child has terminated, but the rest of the thing is the same. You see the code. It prints out the value 0, before the joint, and then it prints out the value 5 after the join. Parent prints out 0 still.

The parent thread had enough time to wait a little while, all the computation had taken place in the child thread, and the value has updated. There is only one variable printed out by the child process, the parent process does not change anything it has its own address space.

There are a lot of things underneath that is hidden off.

Let's go back to the producer consumer problem which is message passing. As we said in the last lecture, when we have message passing: it's a primitive pattern by which all /most communication takes place between processes.

If you have a huge directory, with a recursive ls, you have to set up this message passing queue, and you have wc values coming out and have a huge output.

How do we think of the message queue? Think of it as a buffer. As an element is put into this message queue, the in is put into available memory, and the out pointer is bumped up to the queue to take up to the end. What the producer is doing is manipulating the in part while the consumer is manipulating the out part. What would my producer and consumer programs look like? The producer looks at this count

variable in shared memory. When the count is equal to the buffer size, my buffer is full. The producer would have to stop. When there is an element, account is not equal to buffer size. I am always pointing to the next available buffer size and I am manipulating the buffer size.

If cout = 0 , there's nothing in this queue. When the count is greater than zero, it removes the buffer and decrements the value of count. The only thing that is shared between these two is the count. The producer and consumer can run on their own threads and processes. Threads and processes are synonymous in synchronization. Each of these processes make non zero processes. One process may have its own environment, another process has its own environment. One may be long or short or have a lot of processes, so can't judge the speed.

What do we mean by at the same time. Count is in RAM, but if want to bring in the Alu, do it in the registers. Registers are faster than the RAM. Need to load the values from the memory location to the registers. If there is DNA and viruses are running at the same time, it could happen anywhere at this problem: Race Condition. When you have many lines of codes, you want to ensure that only one process is in the critical section at any point of time. We need to guarantee mutual exclusion to updates of the count variable. If one of the process is in the middle of updating count, no other processes can be updating count at the same time. You can build a mechanism that guarantees no mutual exclusion but may hurt yourself. An example that is almost trivial: I will put the process to sleep to assure mutual exclusion.

There must be overall progress in the system. There shouldn't be a system where all processes are in a dead lock state. So a solution to the critical section problem is Mutual Exclusion, Progress and bounded waiting. If the resource is going to other processes; this process has been starved. Bound must exist on the number of times that the processes are able to enter its critical section after a process has made its request to enter its critical section. Process cannot be starved of its resource.

This process that has accessed the resource should do something in the exit section.

```
While true {
Entry section
        Critical section
Exit section
        Remainder section
        }
```

Remainder could be anything, overall don't care. This is a general schematic for any kind of synchronization problem. Our task is to focus on the entry and exit sections.

Load and store instructions are atomic: first task has shared variables. It can be generalized to any process I want, but I have process 0 and process 1. And they access the (wantEnter in shared memory). This is an array of two shared elements. Before process 1 or 0 enters, it needs to indicate its desire to enter. If process 0 wants to enter, it checks if want enter of the process 1 is true. If it is true, it does nothing. Waits on the while statement. But it becomes false, it sets want enter - true, and enters the critical section and change want Enter to false and allow process 1 to enter the resource.

The want enter is not really visible. There is a big problem in the while loop: efficiency. Takes up CPU time. Would we get into the situation where there would be more than one process than any given time. You have to force yourselves to think in parallel. If think in one process, process 0 will execute all of this code with wantEnter 0. The two processes would enter its want Enter concurrently, enter the critical section at the same time. Both enter when want Enter = False, and change its respective want Enter to true.

Then with have by turns.
While(true)

```
        Wihle (turn ! = 0)
        Critical section
        Turn = 1;
        Remainder section
}
```

Say turn is 1, process 1 turn. But if something stops process 1, then it will never switch to process 0's turn.

Process 1 will do the opposite. If turn not equal 1 will wait, and will turn it to 0. So let the processes take turns; turn variable indicates whose turn it is.
Now we go to the Peterson's solution. Makes use of both the approaches. One with the array of variables and the other with turn. Combines both. Has a variable turn, and a boolean, a set of flags which = wantEnter. Flag establishes that the process wants to enter. So I can make progress.

```
Wile(true){
        Flag[i] = true;
        Turn = j;
        While (flag[j] && turn == j);
        Critical section
        Flag[i] = false;
        Remainder section
}
```

One is flag j = 0, if j wants to use the resource, will set it to true. If both conditions are true, he is about to enter the critical section. So just have to wait for a while. In the exit section, I and J and all the processes doing this set their respective bit to false in that array.

- [x] HW: Need to say why attempts 1 and 2 failed, and argue that Peterson's solution is correct: it satisfies mutual exclusion and progress; Show that it satisfies bounded waiting also.

Synchronization Hardware: Many systems provide hardware support for critical section code. Solutions become easier with hardware support. Mutual exclusion on uniprocessors: could disable interrupt on process. One solution for critical section: if any process enter the critical section, make the critical section non- interruptable. If any process wishes to enter the critical section: the OS will make sure that no other process will interrupt the critical section. We will be looking at two primitives supported by hardware: Test and Set, and the other one is called swap. It's guaranteed by the hardware to be atomic. That means it either goes through or not, but cannot be divisible any further. Test and get and swap.

getandSet is true, somebody is using the resource. It means that the value of the resource is true; somebody else is in that critical section now. It yields the CPU.

Monday

Monday, February 12, 2018 8:34 AM

Non-preemptive kernel is free from race conditions in kernel data structures: True
Peterson's solution needs special hardware instructions? False: It's an entirely software based instructions
Test_and_set() and Swap: Assembly language instructions; system calls; atomic instructions implemented in hardware:
The key word here is atomic: assembly language instructions are software based: It is not an instruction set: it is assembled into low language instructions.

The key thing is atomic: It is indivisible: It can be assembled into a number of instructions
Now" Why is disabling interrupts not a good idea for multiple processor architectures:
1: Disabling interrupts must be coordinated between all processors. This can be time-consuming
2: Just because one CPU is in critical section, all other CPUS are delayed
3: System clock may be affected by too many interrupts. If you have a hardware clock. An oscillator. System clock is registered somewhere. If you disable interrupts; system clocks going to be skewed over a time.
On single processor architecture, system small, can still get by.

Remember the get and set: Does this solution preserve bounded waiting?
You have a hardware lock. Set the lock to false. Which means the first process that enters get access to the critical section. You do a get and set on this lock. While its true; the thread yields. When you are trying to set it to true; somebody has gotten ahead from you. So you yield the CPU to some other thread. This is an initialization thing; you initialize lock to false, waiting for some thread to come in. With a true; that thread will get access. Instead of tying up the CPU, the thread yields. Is the thread still runnable? It is in the JAVA sense. Should it be running on the CPU? Because it is consuming doing nothing; so the thread yields.

On the other hand, some other thread, sets it to false, this thread can proceed. When you do a false after the critical section: This is not a unix scheduler, but a java machine scheduler; but the principles are the same. You don't have device interrupts; but in principle, you have multiple threading, interleaving of different execution sequences. So when you set the lock to false, JVM will wake up some other thread that had attempted to enter critical section and are in this yielded state. One would be woken up and given the CPU.

Question: Bounded waiting? Multiple threads are trying to get that same lock. Critical section is not that bad. But all it does is to yield the CPU. You can have a thread that enters and gets and sets the lock and go again. There's no guarantee that within the bounded amount of time, the waiting thread will get the CPU.

After j enters CS (and sets lock to false) it can quickly loop back (before I has a chance to run) to set lock back to true and enters again. No guarantee how many times j can do that before I will find lock equal to false inside the while loop.

We looked at hardware and Peterson's solution: They all require busy waiting: like somebody pointed out, that's quite wasteful. Processes will be waiting in a loop. If this fails, the process goes back and tries again. It wastes the CPU time unnecessarily. To solve: Dijkstra (shortest path problem) came up with this semaphore.
Semaphore express a solution without busy waiting. Basically; we have an acquire (entered the critical section) and then have a release. Two atomic solutions are on the variable, acquire and release.

Semaphore is something like a lock. You have to acquire semaphore and you have to release semaphore. One is a binary semaphore and one is a counting semaphore. This release here: release implements the value at this semaphore variable. Something like the count ++ and count -- both of those have to be done atomically, otherwise you have a race condition. This acquire is done atomically. This value <= 0, it means no op. this acquire operation does not really do this in a busy way. In a release operation, there is no chance of a busy wait. The acquire operation looks as if it is doing a busy wait. It attempts to read a value, and if the value is less than or equal to zero, it will exit this operation automatically. If not, it will have access to the critical section.

You might say that since you are waiting in a tight loop, it is the same as the Peterson's solution. There are two types of semaphores: binary means only have zero or one. What it does it gives a mutual processes to the critical section. Many processes trying to access the critical section, only one can have access. The counting semaphore is initialized to a value, say n, and every entry into that critical section reduces n by one until it hits zero. And then it stops. Counting semaphore does not provide mutual exclusion, and is useful for more general condition synchronization. Makes sure that the condition needed for a process to continue working is true: buffer not empty for consumer in bounded-buffer producer-consumer problem.

Sempahore sem = new semaphore(1)
Sen.acquire() //critical section
Sen.release(//remainder section
You can have multiple processes that try to enter, than you have a negative integer, rather than a boolean. You have to decrement. If you set value to 1, then the first process that enters the critical section will set it to zero.
The release is the exit section, while the acquire is the entry section.

When a thread executes: if the thread needs to access this resource, then it will go through the acquire method of the semaphore. Every thread will acquire that semaphore: and then execute the critical section and release the semaphore. One example is that these five threads are doing matrix multiplication. Each segment of the array takes a different amount of time. A thread can go get another index. Each submatrix gets a different amount of time, so you can use this structure to implement a semaphore, mutual exclusive. You use the critical section to guard a shared integer in shared memory, which could be implemented in a critical section.

The key thing is the use of the semaphore acquire and the semaphore release. So the worker implements runnable. It repeatedly enters critical section then exits it to do something else. The reason for the mutual exclusion is because you want to update the shared memory in a mutually exclusive fashion. It is a binary semaphore: How would we initialize semaphore? By setting value to 1.

Even though we say that this solves the problem of busy waiting? If the busy waiting was for the entire duration of time for the critical section. The critical section could take a long time relatively speaking. But if you have a critical section inside that acquire; all it is doing is resetting the counter. That should not take a long time. So the busy waiting inside the critical section is not sustainable, but the busy waiting inside the acquire will not take too long because there is nothing else other than value--. If there are ten threads doing this at the same time, nine of them would be waiting until 10 memory cycles because each of them is waiting on a hardware section.

But if the thread fails to get the semaphore, it will not wait in the critical section. Some other thread will try to get the busy section. Problem was busy waiting was that thread exceeds time set, in this case, it will still need to yield the thread. Does hardware guarantee yield thread? It will not. JVM supports semaphore, but hardware does not guarantee that.

The point is that you need to have a mechanism: If you have multiple CPUS; but the critical sections are likely to be very small. You will take the task and put it to sleep to yield and come back after a long time. It's unlikely to wait that long because the critical section is so small. This rests on critical section is small, if not, it is better to replace the while statement in the if.

Busy waiting is okay if you have multiple CPUS but not for uniprocessors (executing instructions without doing anything useful). In those conditions (if critical sections is short), the busy wait is called a spinlock. In general, we can't say that the CS is short.

This brings us to another construct which is used with semaphores" a queue. By integrating the semaphore implementation with the CPU scheduler, and some of the states you have queues (waiting, running). So in essence, each acquire can't complete, like you pointed out just now. If you were to use the thread, you yield a thread. You can actually have a queue and enqueue that process. The two scheduling operations are block and wakeup. For each semaphore, there's an associated queue. If the semaphore operation fails at the acquire stage, instead of doing the spinlock (and at the process level, no question of yield), we give the process to a queue associated with the semaphore. In wakeup, we remove the process from the waiting queue. We don't arbitrarily wake up any process and change its CPU scheduling state to ready/running. Only CPU scheduler is waiting to run.

We can see the implementation of that. The acquire is setting the value less than zero. The implementation can indeed go less than zero. You can block the process, put it on a block queue, and for the implementation of release, you can increment value, and if value less than zero, you can remove a process P from the list and wakeup(P). You can do it in 5,4 order and can have priorities. If it is a prioritized setting, then you have starvation.

So let's just say FIFO, the one waiting for the longest time is the process that is woken up. The semaphore is releasing the critical section problem. I have a semaphore, solving the critical section problem, and the semaphore is making use of the acquire. How is the acquire solving the critical section problem. How is it escaping the circular logic. We claim that this acquire is solving the critical section problem. But then if you look at the definition of acquire; That itself has a critical section.

How does the acquire solve the critical section problem?
We will use the software that do busy wait: like Peterson's algorithm which makes use of turn and flag to solve bounded waiting, mutual exclusion and progress problems. Those are solutions that do have busy wait, and the busy wait, like we saw just now, it is not a long busy wait. It is a short busy wait. It is only the update of a semaphore. There is no arbitrary piece of code that can wait forever.
In single-processor environment, we can solve it by simply inhibiting interrupts during the time the acquire() and release() operations are executing. Once interrupts are inhibited, instructions from different process executes until interrupts are reenabled and the scheduler can regain control.
In a multiprocessor environment, interrupts must be disabled on every processor. Otherwise, instructions from different processes may be interleaved in some arbitrary way. Disabling interrupts can be difficult and diminish performance. Semaphore systems provide alternative locking techniques -- such as compare and swap or spinlocks to ensure that acquire and release are performed atomically. acquire and release have not eliminated busy waiting, but they are guaranteed to be short because they implement semaphores. It is short (no more than ten instructions) and the critical section is almost never occupied and busy waiting is only for a short time.

Right at the beginning, we have a multiple buffer size, and if the queue is not full, you add an element into the queue, you implement an in. The consumer will do nothing if the buffer is empty, otherwise, it removes an item from the buffer and increments out, the buffer_size = 0. This has all kinds of problems. If you do without mutual exclusion, you get into trouble because of a race condition. We implement a semaphore to increment this. How do we extend to multiple processors and consumers. Because you have a single producer and consumer, we do not need to worry about the in and out in shared memory.

Now we want to extend to solution to multiple producers and consumers.
Bounded waiting Problem.
We want to have multiple producers and consumers.
Three semaphores: 1 binary for the buffer; two counting for variables in and out. Initialize producer in variable at N and consumer out variable at out.
int n;
semaphore mutex = 1;
semaphore empty = n;
semaphore full = 0;
mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to 1. Empty and full semaphores count the number of empty and full buffers. The semaphore empty is initialized to teh value n; the semaphore full is initialized to the value 0. Producer producing full buffers for the consumer or as the consumer producing empty buffers for the producer.
do{/*produce an item in next_produced*/
acquire(empty);
acquire(mutex);
release(mutex);
release(full);
}while (true);

do{
acquire(full);
acquire(mutex);
/remove an item from buffer to next_consumed/
release (mutex);
release(empty);
/consume the item in next_consumed/
}while(true)

From <https://edimension.sutd.edu.sg/webapps/assignment/uploadAssignment?course_id=_2358_1&content_id=_53335_1&mode=view>

There is at most n elements in the buffer. That's the second condition, initialized the counting semaphore to n. Third: there is at least one element in the buffer for process to continue. If 0, then it has to wait. Condition is supported in Java, associated with the semaphore. Constructs in Java for supporting synchronization. Semaphore gives you the basic property of mutual exclusion and guaranteeing you: Does not guarantee you process and bounded waiting. Within the semaphore yes, you can write the program in such a way that can guarantee to make progress and bounded waiting. All the semaphore does is to guarantee mutual exclusion.

At the language level, each object has an associated binary lock. It inherits from Object class, there is a lock that becomes available through a synchronize method. This method becomes under this lock. And this lock is released when exiting the synchronized method. The lock is put in place. When you exit the method, lock is removed automatically. Mutual exclusion guaranteed from this method. You can choose if you want to; you don't have to lock it. At most one thread can be inside at any given time. Threads are placed in an entry set.

Java is also a scheduler. It is exposing some of the scheduling issues. You have an object and a lock. Lock can be invoked for methods. Every thread that wants to manipulate that object will go to an entry set. (queue of threads waiting to enter any synchronized method for the object). They will all be sitting on the same entry set. There is a producer thread; producer thread holds object's lock in insert() when calling yield(). Because buffer is full; yield won't give up the lock. In variable is manipulated when buffer is not full.

Consumer is the same: while count is zero, nothing in buffer, yields CPU to another thread. If there is something, the buffer is decremented and does the process. What's the problem? When you do a yield, you are yielding the CPU, you are not unlocking the lock. Another thread can come in but that thread cannot get a lock, because that lock is already locked by some other thread. We will look at the solution tomorrow.

Wednesday

Wednesday, February 14, 2018 11:32 AM

Java virtual machine actually runs different threads in different priority levels. Think of garbage collection: no malloc. It must be doing that in the background.

You have a shared buffer with mulitple producers and consumers. The shared buffer, manipulated with shared variables *in* and *out,* accessed by producers and consumers respectively. Anything in a scope of a program is a shared variable. Let's say the producers have a higher priority then the consumers. We are looking at the case of more than one producer and consumer. Producers have a higher priority than the consumers. There may be a problem: N =buffer size. That means the count == buffer_size.

When the buffer is full, the producers can't produce anymore. Told you that the producers have a higher priority for CPU scheduling. Must it be in a condition where the producers are waiting. It's not a deadlock; it's a life lock. A live lock is when there is apparently progress where the threads are executing; they'll test the condition whether buffer is empty or not. Another producer enters; finds it to be full, and goes out. Consumers may not get the chance to enter and get a CPU and consume resource and allow producers to produce. This is known as livelock. A dead lock is when the threads are waiting for resource; cannot get access to the CPU.

Need to be aware of the nuances. Programmatically you can assign priority levels to your threads. Can launch at higher and lower priority. There's a possibility of livelock: thread not getting a chance to run and not able to make progress.

That brings us to Java Synchronization. What are the various mechanisms that we have looked at. The problem of synchronization. The shared buffer, with a count variable, and you can see that there is a race condition. The way the count variable updates concurrently over multiple threads and processes. Then we looked at some solutions; some software solutions for that. We also mentioned in passing the Decker's algorithm. You need to expose them at a higher level; higher level mechanisms for obtaining mutual exclusions. We looked at the semaphore; we looked at two different kinds of semaphore: binary semaphore gives you mutual exclusion over the resource. Only one thread can be in execution in the critical section at any time. The counting semaphore can be initialized to a value greater than 1. When you have that, you have the possibility of more than one thread entering. That is useful for multiple resource units. We saw this application in the multiple buffer. Multiple producers entering the buffer concurrently. You can initialize that to n the size of the buffer.

We moved a little more to the notion of a Java lock. It is another way of saying that the lock allows mutual exclusion. Allows you to have a software construct where the program does not worry about the semaphore and uses the keyword synchronize where the thread would access the method in a mutually exclusive method. Make sure that this database updates mutually exclusive; don't want it to be in concurrent mode on that method.

Pictorially, we can represent it as threads. A thread has acquired the lock, executing a method in the scope of that lock. Other threads that also wish to obtain that lock would be in an entry set; like a queue, but not in fifo order.

We are looking at the race condition; we introduce the lock. We get into the situation where you might want to yield the CPU to the next thread; because the condition for entering the critical section is not yet met. You yield the CPU without yielding the lock; can get into deadlock situation. How can that happen? Another thread does not have the lock in order to make progress. The producers make this method call to insert, while the consumers entered this call to remove. The producers are still holding onto the lock while yielding the thread. While the consumer has the CPU, the consumer cannot enter here because it needs this lock to enter. Every object has its lock; if you synchronize mechanism over those objects, the java machine will put into this place.

What would the JVM use to put that lock: a lock generates mutual exclusion; so it will use a binary semaphore. To generate a generic lock, we are not considering the bounded buffer anymore. There are no further semantics. When you use synchronize, you are only referring to the lock. What would JVM implement in the background? There would be a binary semaphore. What would be the solution in this case? How would you solve the problem? We looked at it last class. We have an entry set, and when you put that thread back into the entry set it doesn't help. You'll have to release the lock.

Basically, you have two operations: wait operation and notify operation. The thread that is holding the lock will have to move to a waitset. The thread releases the object lock; state of thread set to block; (bit like waiting state in the context) ; the thread is placed in the wait set for the object. What this means is that the next consumer can then lock and enter its critical section. It will lock the object and enter its critical section. While coming out of the critical section, the consumer thread will notify.

Here; you have the wait in the producer and consumer. Consumer able to enter critical section, because of synchronized section, and will invoke the notify call. What does the notify do? An arbitrary thread T from the wait set is removed. And moved T to the entry set (status is set to runnable).

We have a wait set. Here is the entry set. It picks up one thread from here and moves it here. Remember these entry set are all runnable; anyone could be executed. So it removes an arbitrary set and puts it here and changes status to runnable.

We have introduced a notify and a wait. There are two things we need to introduce; this synchronize method with wait and notify solves mutual exclusion problem. Wait and notify solves the condition synchronized problem. A lock locks the entire method. Whatever be the condition; if you enforce the lock, the thread is forced to wait. Whatever the condition like buffer is full or not full (or not empty like for the consumer), whatever be the condition; the lock will enforce that. But if you can put a condition; then you have a final degree of control in the scheduler. You can check to see if the condition is met or not. Each of these threads would be waiting in a different condition. A consumer thread will be waiting for thread condition not empty. You might make a producer and move it back from wait set to entry set. With this condition capability, you might see which condition is actually met. You might move it back to the entry set. So you make progress.

You have a final degree of control over the scheduling and operating system.
? Synchronize enforces a lock. Say you have file methods, you have five methods with synchronize tag line, they can only run one after another.

You are not only yielding the CPU with the synchronize, but with the lock as well. We will not go into interrupt section. Java gives you capability to catch interrupt. Any thread that enters by using the lock will also have a notify. If there are no threads waiting at that point, no action is taken.

The wait is quite similar to the yield. There was a yield operation; when you yield the CPU. There is a difference between the wait and the yield. When you do yielding, thread goes back to the entry set and be scheduled again. Thread.yield, when you see the yield coming in.
You will have to increase the counter. Which you might not do in a lock. You will not have to do an action when exiting from one lock. So basically, the thing is that the yield does not release the lock and the wait does. That is achieved by the use of another keyword: notify all. So in the absence of there being a condition; we would like to have a condition. Or notify upon there being a condition. What that would do is make possible the condition that all threads are waiting for. Waiting for buffer not full or buffer not waiting, and notify all threads that are waiting. We would like to have a named condition.

This is a binary semaphore, and when I launch five threads with this construct: so I have five threads. For each of those threads, I do a new thread to get the object and I start with this. Because of the semaphore, if I have a synchronized access, then any one of those threads can go in at any time. When we have semaphores, it is exposed to the programmer. The programmer is exposing the lock by himself. The programmer uses a private semaphore and the thread or the worker bee acquires the semaphore, goes through critical section, and releases the semaphore.

Now we have a case, using this template, we set up a structure where the nth bee will schedule the next bee. (N+1). We have this concept of turn; and when I am exiting the critical section, I will turn it to turn + 1%5. So I am the first thread, called thread 1, about to finish. Thread 1 wishes for CPU to thread 2. So turn = (Turn + 1) %5. 2,3,4,5 are all in a blocked state, and notifier can pick up any of those threads. This problem can be solved by the notify all. It is slow, but 2,3,4,5 are all put into the entry set.

When we get into the wait operation, we are assumed to be holding a lock. Say a thread invokes a wait without holding a lock.

1. Y checks that buffer is empty
2. X puts item into buffer, makes buffer non-empty, calls notify() (to wake up Y in case Y is waiting).
3. Y calls wait() after checking that buffer is empty in Step 1.

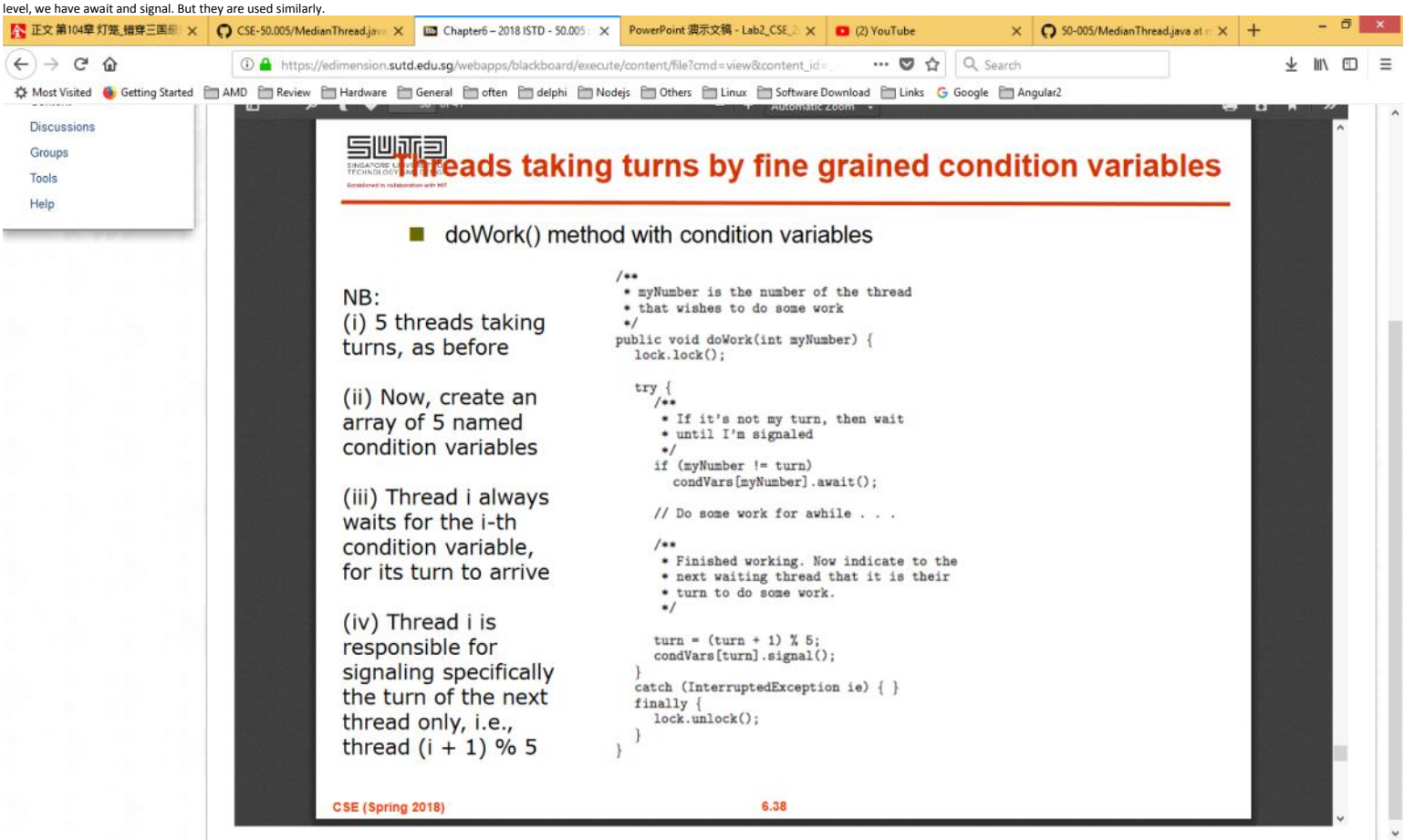
You can call wait without holding the lock. You can do the release before the acquire. Nobody will do it in their right mind. You can do that in many bad situations. You have to acquire the lock first before running the wait if it is synchronized.

You have pre-empted the X, already in Step 2, Y is blocked as it calls wait in step 3.

A re-entrant lock is something like that: when you hold on to a lock, you have the ability to get access to the same lock without waiting for others to release it. So this is basically what is going on. When a thread is put on the wait set. Before going to the wait set it should release the lock. It is eligible to re-enter. It can't make progress, but all other conditions allow it to go back and re-access. No point giving the CPU. This thread can enter again for execution. The program will not be incorrect if it did so. That lock is re-entered. This thread can re-enter again without further tests.

With the re-entrant lock, we can associate something called condition variables. This are objects allocated in java, and when you have a re-entrant lock, you can have a condition variable, which is a condition. It's a special java object called a condition, associated with a re-entrant lock. A lock is reentrant if it is safe to acquire the lock again by a caller already holding the lock. In the above code, note that the condition variable condVar is associated with the lock key; in general, this association makes sense because a thread always holds a lock when a condition being signalled or waited for.

We have two new methods: At the level of the lock, we have wait and notifier. At the level of condition level, we have await and signal. But they are used similarly.



When the turn is reached, then it will wake up the thread that needs to be woken up.

When you implement both by releasing as a semaphore. Semaphore has semantics: binary or counting. When you release, you increment. Do you do that with a notify? No such thing. That's the reason why slide 36, you see a scenario where you get into a race condition because there is no history. In semaphore, it forces you to remember by the count state variable. So that's the difference between the release and the notify.

Semaphore solution does not need the count variable.
One final topic: Java lets you use the synchronized construct on any chunk of code. Why would you have to lock the entire thing? You take any piece of code, and put around it a mutex lock. In the middle of a method; you don't have to synchronize method, then have critical section within that. Java would enter in a synchronized lock. Any synchronized call using that lock will be serialized over that piece of code.

Deadlocks

There is no difference between programs in an operating systems. We say resources in an operating

system, what would that be? Would a CPU be a resource for an operating system? Yes or no? You can have multiple CPUs, each of those a resource. From the process point of view, you can only think about processes and resources. Processes obtain resources and that's granted by the operating system. Resources are taken away. If you have a time shared operating system. You will have time slices of the CPU. Sharing the CPU with many processes. CPU is granted to a certain amount of time and taken away from the thread. A chunk of memory, goes through virtual memory, but in essence, process has a certain amount of memory for a certain time. Every thing that you're talking about is a resource. Whether process level or a thread level. You have access to that resource and can do different things with it. Physical resources, logical resources (Disk Drive, reformatting disk drive) and the same disk drive could house a certain number of files in a file system. Files would be the resources. Can get into a deadlock system with file access.

You have two processes, P1 and P2. P1 tries to open file 1 and then file 2. And P2 does the reverse. Would either of the processes make progress? No. That would be an example of a deadlock within an operating system. Multip-threaded could have a deadlock. Or we can have within the OS context possibility of deadlock with files. To get a handle on this problem, let's look at a model for deadlocks.

Essentially, they operate in pairs. If you acquire B and A, you could have a deadlock between two binary semaphores A and B. You might be able to get away if P1 proceeds and finishes with B and A, but you could have an interleaving.

Deadlock doesn't come into computers. Bridge crossing example. Each section of a bridge can be viewed as a resource. If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback). Several cars may have to back up in general. Starvation is possible (cars in one direction only keeps going).

There are types of resources, R1, R2, Rn. R1 could be CPUS, R2 is memory segments, IO devices etc.

They are multiple units of the same type. The operating system has Wi instances (or units(of each resource type. Ri that it can allocate to requesting processes. Each process utilizes a resource as follows: You have a request stat where you request one instance of a resource from OS e.g. get access to printer. And then use that instance (use the acquired resource privately to do its work). And releases it (return an acquired instance of a resource back to the OS e.g. give back the acquired printer so the OS can give this printer to another process. A process can have any number of resource instances but only one at a time.

The deadlock may occur if you have these conditions.

1. Mutual Exclusion: one process makes a request at a time. Let's take the case of opening a file. Lot of times, when we open a file we don't need to read the file. That open can proceed quite normally because multiple readers can be reading at the same time. Mutual Exclusion only happens when want to write a file. When want to read a file; no problem
2. Hold and Wait. A deadlock takes place when the process must be already acquired a number of resources and waiting to acquire some more.
3. No preemption. Preemption is when we kill a process in deadlock. Or let's force it to give up some of its resources in some way. That is preemption. So the resource can be released only voluntarily by the process holding it. After process has completed its task
4. CIRCULAR WAIT. What does it say? Circular wait says that you have a collection of processes 0 to n and form a ring. P0 is waiting for resource P1 is holding on to, P1 waiting for resource P2 is holding onto…Pn Neither processes are all waiting and all deadlocked. Doesn't mean no process waiting in a system. But these set of processes are deadlocked. Doesn't mean system is deadlocked. There could be other roads where traffic is okay, but at that road, jam.

These conditions are necessary but not sufficient. You may come out of a deadlock with these conditions.

Let's try to construct a directed graph with a set of vertices. Nodes of a graph can be of two types: Either squares or circles. Squares are resource types, circles are processes. Instances of resource type are represented by squares but big square is resource type.

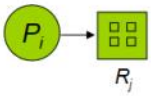
Having represented the nodes you have two types of edges. Request edge could go from Pi to Rj. If process p is waiting for instance of resource type. So you represented that by an arrow. An assignment edge goes from a resource instance to a process. Not from type but from instance.

Resource-Allocation Graph (cont'd)

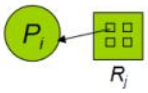
- Process (first type of nodes/vertices)

- Resource type with 4 instances (second type of nodes)

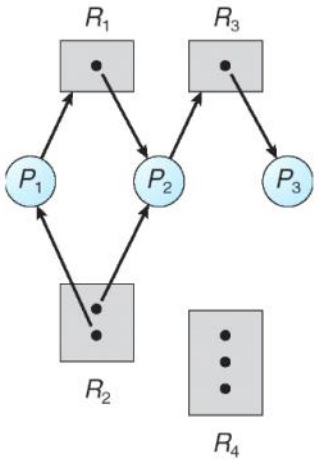
- P_i requests an instance of R_j (first type of directed edge)



- P_i is holding an instance of R_j (second type of directed edge)



Example Resource Allocation Graph



Is this system deadlocked? *Hint*: Is there circular wait?

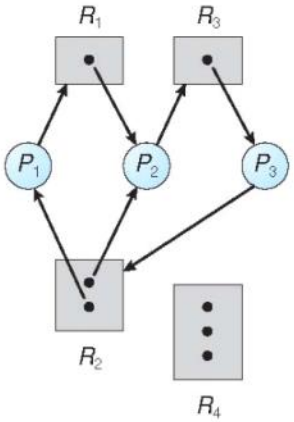
Can we see if there is a cycle over here? There is no cycle. So there cannot be a deadlock in this graph. Can we say why? There's this process P3, holding onto a resource R3, there's only 1. R3 is not in waiting. P1 is waiting for instance of R1. P2 is waiting for instance of R3. So P3 can finish, R3 becomes finish. Edge established for P2 to R3. P2 finishes execution releases resource R1, and given to P1. So all of them given to progress.

Discussions
Groups
Tools
Help

Resource Allocation Graph with a Deadlock

P1 holds an instance of R2, needs an instance of R1 to continue execution so that it may complete its task later

This graph *has* a cycle

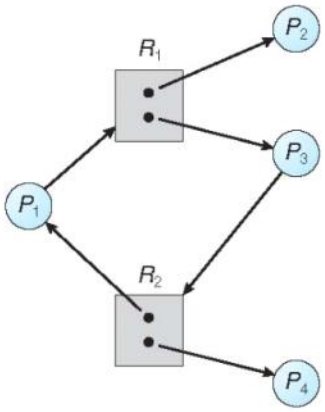


Can any of P1, P2, P3 acquire the resource it wants to complete its task?
Hint: The processes won't release the resources they are holding until they can continue execution and complete their respective tasks.

Possibility of deadlock, but not necessarily a deadlock.

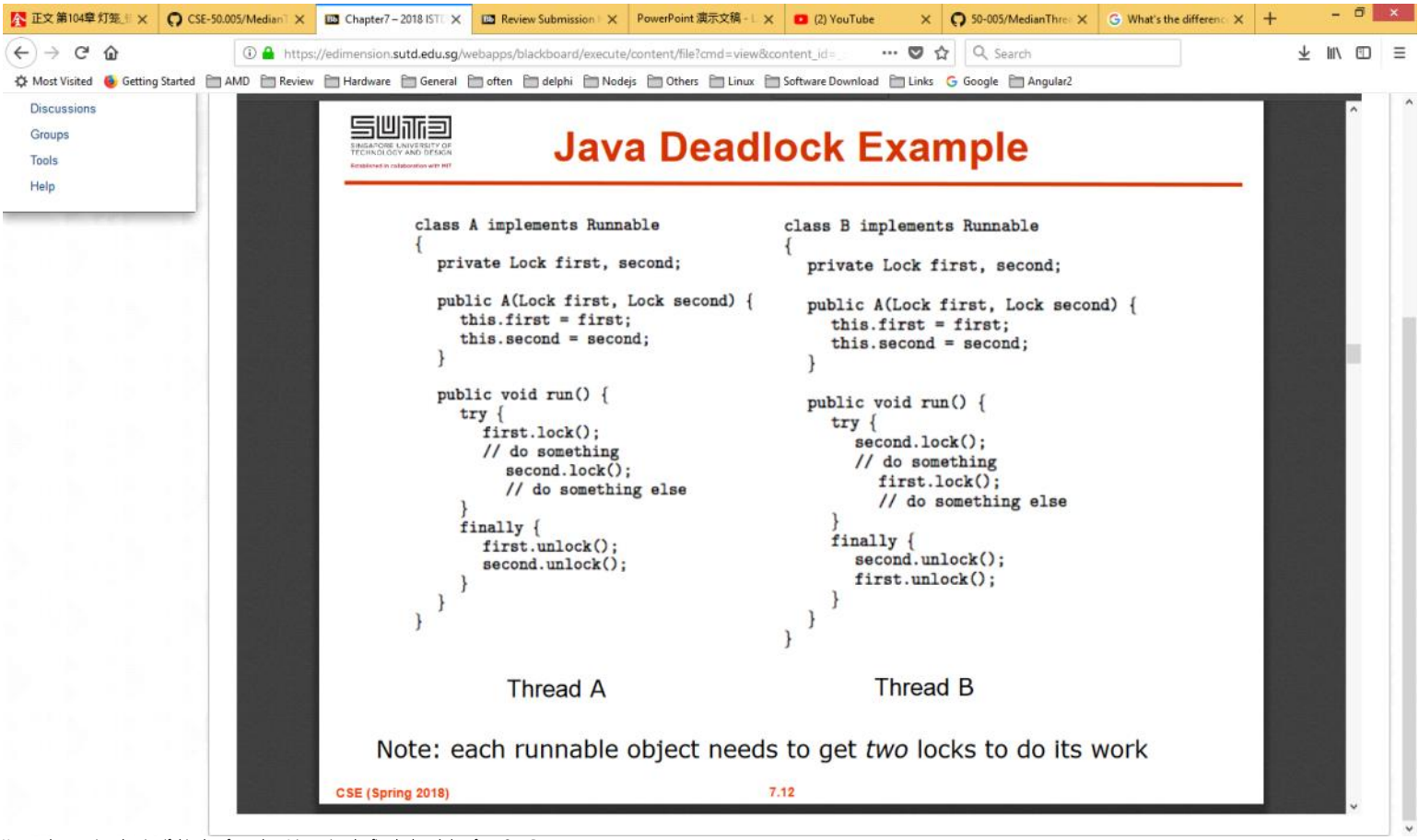
Discussions
Groups
Tools
Help

Graph with A Cycle but No Deadlock



Multiple instances of resource: Cycle is *necessary*, but *not sufficient*, condition for deadlock
This graph has a cycle. But what is a possible completion sequence of these processes?

Because P4 can finish, P3 can get that resource and finish as well. This is an example of cycle but not deadlock.
Now, instead of two copies of R1 and R2, have one copy of each. Since that's the degenerate case when we started out with semaphores A and B in the very first slide. So maybe P0 can acquire A and B.



You can have an interleaving if this class from class A is getting the first lock and class from Cass B gets the second lock.

Thursday

Thursday, February 15, 2018 1:41 PM

Deadlocks consist of three actions we need to design. The first one is you need some information about the future; how is that information encoded, according to this? The maximum number of resources you will ever need. When we start the program, in the beginning, it will tell the operation: I will need them in the future. How many of them I will need? I will need 5. Declare to the operating system. At any point in my execution, I will need 5 resources. The operating system will hold you accountable to that. If you need just five printers. If you have already four, you ask for two more.

You check if I have enough units of the resource to give you. Will wait at least. The other check I need to do if I violated the contract. At most 5? Or 6? Also deny it because the request is not legal. Once these two checks clear: By granting the resources now, based on your knowledge on how much they need them in the future, is it okay to give the resources to you. By granting the resources to you in the future, do I have a chance to get into a circular wait.

Just the possibility of deadlock would be sufficient to deny it. Because you are looking into the future

Resource allocation state is defined by the numbers of currently available and allocated resources, and the maximum number of resources that each process may need in the future.

SAFE STATE
A state of the system is to determine how many resources you will need and so on. My job is to determine if the state is safe; whether there is no possibility of a permanent circular wait. Safe state :if there exists a sequence of all the processes in the system such that for each Pi the resources that Pi will ever need can be satisfied by the currently available resources plus the resources held by all the preceding Pj with j < i. If I have enough resources to satisfy it, it will be put into good use. In a deadlock, I won't be able to get my resources to continue. So you are worried about not getting the resource that you need.

When they are running before you, then they are getting resources ahead of you. By letting them run, you are sure they will terminate. You can't just take something without returning. You must return what you have back to the system. Then the p1 to pn, what kind of order are we defining.

If Pi's resource needs are not immediately available, then Pi can wait until all the preceding Pj processes have finished. When the preceding processes all finished, Pi can obtain its needed resources, do its job, return its allocated resources and finish.

When Pi finishes, Pi+ 1 can obtain its needed resources and finish and so on. <P1,P2, Pn> defines a feasible for all the processes to finish.

If

Monday

Monday, February 19, 2018 8:35 AM

You have two threads implementing runnable. The first lock is implemented, and the second thread is locking the second lock and then the first lock. Since they each take one lock, and they are waiting for the other lock that the other thread has. But you would see the non-deterministic aspect of this. A followed by B. If you introduce a random sleep, then you can sometimes see this is A followed by B, and B followed by A. We say that this problems are non-deterministic. Function is a mapping of an input to an output. Operating systems don't behave as functions because there is a lot of randomness. Sometimes the order varies, so the deadlocks may exist or not.

Methods for handling deadlocks. We let the system run and we somehow come out of it. The three methods we'll study are first avoidance, detection and then prevention. When we try to avoid a deadlock from happening by allowing processes to declare in advance all the resources that they will need. The intuition came from banking. Think of several projects that are looking for capital, and need money in certain periods of time. In the first quarter they need ten thousand, fifteen thousand. You'll have a sequence of schedules by which projects need to be funded. So you have multiple sources of funding as well. Investors funding sources. Banker needs to decide whether to honour the request or tell the project to wait as there could be a situation where you run out of money and no project could make an advance.

It is rather conservative because it is risk-adverse. You can let the system run, and even if they could put into a deadlock, you might not be in a deadlock. If there's a likelihood of a deadlock, we'll let the system wait. Second is deadlock detection and recovery. Let the processes run. If there is a situation if the process gets sluggish, we'll suspect there's a deadlock and CPU utilization and process would run detection algorithm. Once it has detected that there is this cycle, then it recovers one by one by taking away resources. Then there is deadlock prevention. We have seen in the last lecture; four conditions. If one of the conditions doesn't hold then there is no deadlock. Because we put a lot of constraints of the processes, we put it last.

We just let the processes run. If the situation gets unbearable, you just reboot. In a shared computer system, that's done. Most of the time that's what's done in the real world

Deadlock avoidance: the way is to known apriori the maximum number of resources needed. If the process breaches the contract (asks for fifteen instead of ten), we'll not talk about that. At the time of resource request, avoidance algorithm examines the system's resource allocation state to ensure that granting the request will never lead to wait condition later. Resource allocation state is defined by the numbers of currently available and allocated resources and the maximum number of resources that each processe may need in the future.

So say these are the resources.

n	A	B
P1	5	4
P2	2	2
P3	3	4

Capacity <8,6>

Allocated	A	B
P1	4	1
P2	-	-
P3	2	1

How can we compute the number of resources of A and B is needed.
Initially, P1 needs 5 of A and 4 of B but only has 4 of A and 1 of B. We need to compute the available vector the number of units of A and B still available. So we know six units of A, 2 units of B still available. We subtract the summation of this column vector so that's how we compute the variable. We need to compute the need which is an n by n matrix. Anything else to compute? There is nothing else to recompute. Everytime we have to recompute this matrix. Capacity is a given. This is a given because we know that this is a maximum declare.

Safety algorithm: we assume that request is granted and modify the needs matrix tentatively. We see whether we get into a safety issue. If there is, we do not get into the request. What is the safety issue? Safety issue says that if the system has a certain number of processes. If there exists a certain order of completion of these processes that does not get into a deadlock. If there is an order in which these processes can complete, it is a safe state. Let's check to see if this is in a safe state or not.

After P1 is finished, it will give back the resources. After p1 finishes, we will get back everything that has been allocated. 4 1 will be added back. So available resources go from <2,4> to <6,5> and then P3 gets back. So by going P1, P2, P3. By going in order, there is a number of processes that can go in order.

Banker's Algorithm: Multiple instances of each resource. When a process has a number of units of this resource, it must return that in a finite amount of time. It's not a deadlock situation so it's a reasonable solution. The data structures are a matrix. With a maximum, and an allocation (a snapshot). Maximum is static. And then we have a need, which is a max - allocation. Need needs to be changed every time, and then available changes as well.

You don't assign resources to the process, but create a vector called work and set it to available. You copy it to a vector called work. It is basically to show you whether the process has finished or not. We need to find an unfinished process such that it's resource needs has to be accommodated by available resources. We need to find a process that its needs can be met by need (2). It'll have to come sometime later.

? Isn't it possible that other processes keep getting in, such that a process with a large amount of resources needed, with the possibility these process keep interfering with mega process running? Yes. That happens.

Sum up this column (1st) which is 2 + 3 + 2, and then we get 7. And then 10 - 7 = 3. So we have A = 3 resources. Allocation is 2 for B and 5 for C. So 7 - 5 = 2 for C and 5 - 2 = 3 for B. How would we know if this is safe? Can P 0 be given all that it needs? It needs 7 of A, 4 of B and 3 of C. P0 cannot be given what it needs because it requires four more instance of B and there are only three of B available. We have skipped one step, computing the needs matrix. So needs matrix looks like this.

From <https://edimension.sutd.edu.sg/webapps/assignment/uploadAssignment?content_id=_53601_1&course_id=_2358_1&group_id=&mode=view>

P0	6	4	3
P1	1	2	2

P2	6	0	0
P3	0	1	1
P4	4	3	1

How would the OS take this decision?
Will go with P1 first, with available <5 3 2> after P1 released. P3 finishes with releasing 3 of A, 2 of C. So this available resources will go up to <7 4 3>. Then P4 goes ahead. With returning 2 of C. So new vector becomes <7 4 5> Can P2 go ahead? Can P0 go ahead? Both can. If P0 goes ahead, it can return one unit of B, and becomes 7 5 5 . P2 goes through and returns <10 5 7>
We have established a safe state and this resource request can be granted. We don't know why we are making this enquiry.
Can 1 0 2 be honored or not.
We check to see if available or not. We don't know whether safe. We pretend that we have given it. We modify the allocation and needs and we invoke the safety algorithm. If there is a sequence, then it is safe. P1 is safe because there are a few orders in which it can be sustained.

Now, can we further a grant by 3 3 0 by P4. No. Request 0 2 0 ? No.

When a new request of 1 0 2 from P1, available resources came down to 2 3 0.
1 0 2 is stilll sustainable, as the system will still remain safe after granting 1 0 2 to p1. But for I, we are requesting 3 3 0. This request is unsustainable because there aren't enough available resources. We put it into a queue. Is that sustainable. From what is here, can we make a tentative allocation. Because we have 3 units of B and we are only requesting B.

But when we check the safety, 0 2 0 is allocated to P0, we reach an unsafe state.

Deadlock Detection: Available: A vector of length m indicates the number of available resources of each type. Allocation: Defines the number of resources of each type currently allocated to each process. Request: an n x m matrix indicates the current request of each process. We don't have a maximum but a request matrix. We have a set of processes not making any progress. Some of them are running around merrily. So the OS suspects that there is a deadlock. What are the set of resources that P1 and P2 are waiting for. Very similar to our matrix in the banker's algorithm.

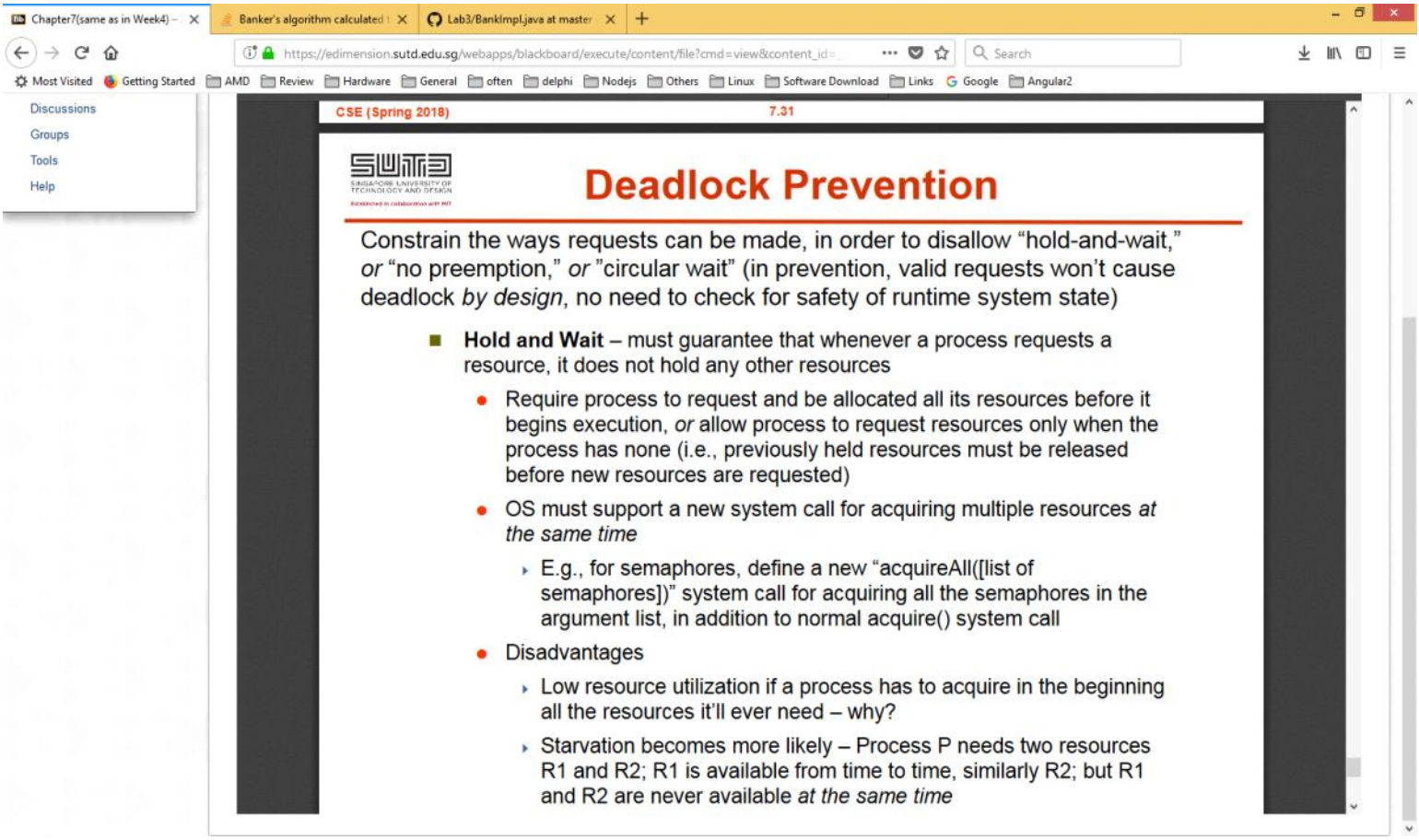
We create this data structure when we want to invoke this data detection. We create a tentative data structure of work. If the processes has not finished; if the allocation of I is not 0 then finish is false. Maybe the process has not actually finished. Because it is not waiting for any resource. This whole thing is because the process does not know if really deadlocked or not. It sets process to true, maybe it is waiting for user input. Then you come to step 2. Find that process is not finished and can be accommodated with available resource.

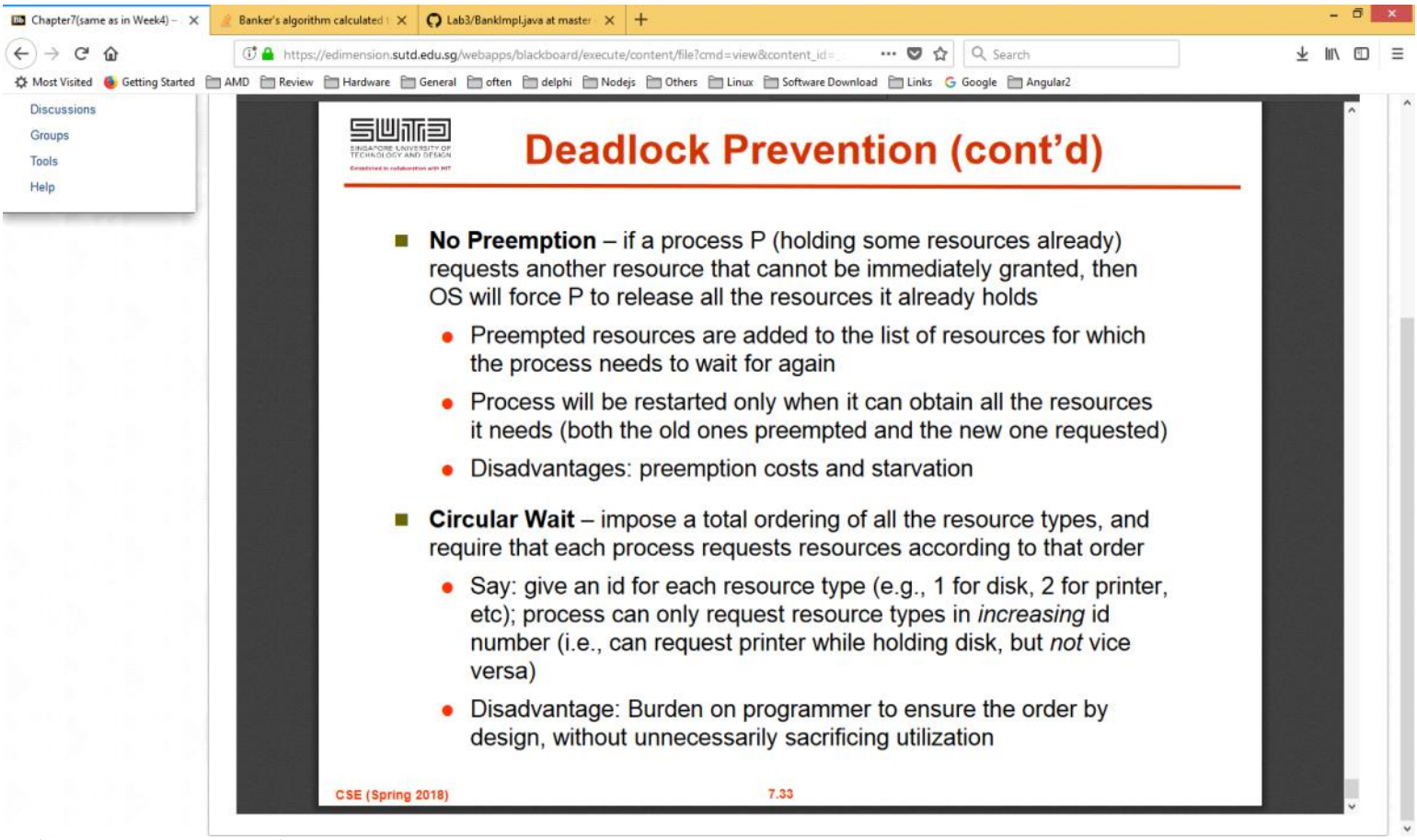
This is optimistic because we are pretending that it will not request for more resources. But we are optimistically saying that we give all the resources it needs, it will complete and this step (2) of detection algorithm is complete.

Now we need to figure out which process to remove. One is to abort all the deadlock processes, you can destroy so many things. Processes may be writing into files temporarily, no commitment, so you have an inconsistent state. But sometimes you have to resort to it. The other one is abort one process at a time until the deadlock process is eliminated.
There are operating systems that allow you to revoke afterwards. Is the process interactive or batch?

This brings us to the topic of deadlock prevention. If you remember, right at the beginning, we look at some criteria. Necessary conditions for a deadlock. Ensure that one of these conditions is not true. For example, mutual exclusion: can we ensure it is avoided? In certain applications; if you have a read only application; need not be mutually exclusion. From a file or from a browser, these can all go in concurrent mode.

Can remove hold and wait requirement? Say there is a certain number of resources, process needs to require everything and get them all at once with one sys call. So what are we doing? We are averting the wait. You don't have to hold and wait. You either get everything or nothing. If you do need some more resources, you stop holding and give it up.





Cost of preemption: updates to database; and you'll study that you have to roll back and have a consistent state of a database. Only then can you preempt the process. When you preempt the process from resources. Starvation is a distinct possibility because you have a situation where the process is forced to relinquish the process.

Can you relax/enforce that there's no circular wait. If you are one process waiting for another process and vice versa. If there is necessarily to wait because for resources. Most of the cases, it's about process vs resource. If you impose an ordering, the tape is before the disk before the printer. Then you broke the circularity. If you impose an order on your request, it's a way of avoiding deadlock. First it acquires the take and then the printer. It must release everything and then enter in the middle.

To break a deadlock: get a grandfather lock. Just declare a lock shared by everybody and then allow everyone else to get that deadlock before other locks.

Tuesday

Wednesday, February 21, 2018 11:33 AM

We can think of an operating system that provides an interface to software at a higher level. This software could be something that an end-user can write. Most end-users sit at software level, and we make use of OS by system calls, delete and creating files. Using the operating system to get into the hardware. That's not what we ever do. In addition to our end users, we also have various programs that are also executing on behalf of the users. For example, a compiler has something that you ever write. Most people don't write compilers. They use compilers. Those are considered to be system software. They use the operating system to operate on computer hardware. That is secondary storage and mapping to primary access memory. We think of the computer as an operation store. The first that comes to your mind is that is my computer, has my information.

Sometimes, as a means of obtaining information. Like for example: Skype call, need computer. Means of computation. If there is information within the hardware, I can't access it like other types of hardware, need means of accessing that information. Most logical way that has been standardized is that of a file. It's a logical collection of information with a name associated with it. That has persistence. What distinguishes it from something in memory, with a variable or RAM, is the term 'persistence'. Persistence across power failures; it resides there and will stay there. My File .txt

What is a file system? We also have a directory structure. All the files in my host computer and a directory structure. Once I have a file and a directory structure, I can do a lot of things. On Windows and Mac, you get a subset of this. On a Linux computer, you will see these commands and a lot more.

Cat: displays the content of a file. There are some files that would show up as garbage; binary files. OS will not allow you to do that.
RM: remove a file. Deletes. In Linux, it's a good idea to with your shell to rm -i. With alias account, you can set up aliases for your command that asks you every time you ask for a file.

CD: changes directory: implemented in your first assignment.
Period. ..: Goes up one level in the directory structure.

These are supported by the directory to manipulate files. We think of the OS as a program that supports system calls. You have system calls that operate each of these operations. Of locating a particular file on your harddisk, on your file system. It creates a file system that creates a hard link and a softlink.

What is a hard link and a soft link?

Symbolic link. When it is going through a garbage collection, it is a softlink that is broken. It can span across other conditions.

Hardlink is a copy of the same file. It behaves exactly like the soft link, except now it is an ASCII file. It is treated like the file I have linked it to. We would look at the INODE structure. It is pointing at the same INODE as softlink is. But if I remove file2, the hardlink is still around. In every way, it is behaving just like a file.

It can be like an object file (output of character). Or output of a loader. There are so many other types of files. If I have matlab program, it's a file. According to various application programs or software, the files that I use are given different file types. At the basic level, the files are a collection of bytes. They can be used to store your pictures, like a JPEG file, or some kind of picture formatting, email, music extentions. The file can be organized to directories: namespaces. A directory is a space of names.

There is a structure to it. In unix, we have a root folder, and from the directory, you have subdirectories that are predefined. Like bin. They are not used by user. Under /user/home, your home directory and so on.

Data in a file system is stored in secondary storage. Especially this: Secondary storage need not be disks. Historically, there were no disks. It started with paper tape and then magnetic tape. We access the file in a sequential access mode. We read sequentially using a file pointer. But that's not always. If we have a data base, you're requesting a piece of data from a data base. I'm speaking of a commercial database, you'll be looking at TB of data. Through indexing mechanism and so forth. A database fetches record for you. Would it be sequential? Let's say you are working in a bank, with hundreds and thousands of customers. How do you think a customer would retrieve record for you? Would it be sequentially or some other way? Is there this humongous file where you get to record n?

You can look at it through python data structure: you are looking at it via directory, and random access. We still have this sequential structure because of UNIX. IBM came up with ways of structuring data and still in use by IBM in its mainframe computers. We also have RAM that behaves as harddisks. They no longer have RAM or disks that has this quality of persistence. It gives you this abstraction of whatever you see in an object is whatever survives through power failures. It is just like secondary store. That's non-volatile.

With this abstraction, you still need to have performance. If you were to support, then very good. Every time I write something into memory, it is flushed to the secondary store. Is that the way it is done normally? We normally cache the memory in secondary store in primary memory for performance. If you want every write to memory to be persistent, why would you need to flush to the hard disk every time(like calculating I to the power of 10000). Value of I would be cached to the memory, and if there is a system failure, or application has ended, or closed, then it is flushed to secondary storage. This whole area of caching and virtual memory that is also supported by the file system. You don't get to see it but you have some way of supporting the secondary store. The caching of the memory and your virtual address spaces.

It is treated as a sequence of bytes or words (File definition) This bytes need to be stored in some fashion. In the older days, IBM came up with various structures of storage on the secondary store and allowed users to have a look at that. If I'm a database programmer, I can see it in secondary store, build index mechanisms and so forth. Over the days, that kind of end user perceptions has moved out. You may still be working on software teams that use that kind of structure. You use the file as a system of bytes. Still a record structure, but record is just one byte. You have fixed and variable length records. It is just a sequence of bytes. You can then have different types of bytes. Different types of files. You can have files that are text, executable, and matlab files that matlab understands. You can have your own.

So who interprets the format? Those extensions, types of file. End user applications: Matlab expects it in a certain type of format that the program language understands. OS is extension. OS kernel would need to. I want to display on the screen something that is a sequence of binary. Just binary words. Is it possible to do that? Or I want to print out on the computer a binary file. System programs know ELF or executable files. User application programs (e.g. browser understands HTML).

As I were to say, there are hundreds of file types distinguished by their extensions. There are executables exe .com or bin. Or no extension, OS treats it in another way. There is something called a permission bit, which denotes it as an executable file. Linux: directories are executable so you can go up and down a directory. Then you have objects; compile machine languages but not linked. Source code like java which has its own extension. Then there is batch file. Which came from the Windows bootup. Then you have your shells. Shell files are for stuff you want to execute.

Then you have text files like doc. Word processor files like wp, .tex (processing system written by donald canute in stanford in 30 years and improved to latex). Then you have rtf and doc which (rtf supported by people away from Microsoft Word). Then you have libraries that are dynamically linked in Linux and Windows and various extensions. Lib, a, so, dll. You have extensions that are for the purpose of documents like postscript. Or viewing, like jpeg. Acrobat came up with the pdf extension. Files that are prepared for compression. Zipping, putting into large directories like huge files. So zip and file and tar. And then multimedia. You have hundreds and thousands of different types of files.

So, file can be thought of as an abstract data type. This term is actually a precursor of object oriented programming. It's an object. An object has data and methods. Which is native to the object, and methods to manipulate the object. If we look at it like that, how can we think of file as an abstract data type and the operations we can do on file. We can refine it further and give it semantics. At the basic level, thinking about hardware and the OS and the whole slew of application packages. They will have the same set of methods to access the file. To think of a file as an object, you need to create, delete, open, rename/move (unix move is rename, it is assigning file a different name. MV is actually a different name). You need to be able to rename a file and you are changing the attribute to the file. You need to be able to write and read to file.

What kinds of metadata do you have? Last modified, properties and you may not include its name but look at directory listing in chronological order. Making use of metadata that the file system has stored about my file. Information about usage. This brings us to the interface.

Let us continue with the attributes. We need a name. We can't do anything about a name. We remember all the numbers of this file. You can't remember the numbers. We need convenient names to identify our files and store it in a structure. In a tree structure. If you think of a directory as tree structure, folder within a folder. How would it not be a tree. What would you have a symbolic link that goes from one directory to another to access a file within that other directory, then you have made it into a graph which is not a tree and not acyclic. If you have a non acyclic graph somewhere, you can keep going round and round in circles. OS has to do with that.

We have name, identifier (tag, internal to the system. System based identifier, identifies file within the system.)Hundreds and thousands of files, medium size within the file system. Unique tags. And some kind of indexing mechanism across that. We talked about different file types from the storage perspective. Types would not make much of a difference. Would you store different files from binary files?

You can have special kinds of ways, like swap files. You have huge address space. You can have a swap file. But you can treat it as a storage, doesn't need to structure any differently. Location is a pointer to file's location on a disk. It's a pointer to a file's location on the disk. What you would see is a virtual pointer. You have a pointer in memory which is a reflection of its location ona disk. You never see its real application on a disk. A device driver, manufactured by the company, would deal with the physical address on the disk itself. Size, will increase as you keep writing. Protection: information about who can read write and execute the file. There would be rings of protection and so on, but they've done away because of the simplicity of UNIX. At the OS level, but you can do more but that's on the application level, handled by companies with software. You need users, groups and the world. UGORWX. You can grant privileges to the whole world to RWX. Or I can grant it to the group or keep it to the user. Those are capabilities to deal with protection for a file. Because they treat files and directories in a very similar way. They even treat devices/ files /directories in a similar way. Dev is given similar abstractions. Finally, you have user ID, date and time. And you say, who use this file. Date and time it was last accessed and so on. Metadata about files is kept in the directory structure, which is maintained in the disk. As you can see, all this information is stored in the directory structure. You can run into megabytes just to store the directory structure of a file system.

Let's go to the topic of abstract data type. You go to a file and need to have interface to a file by methods which you have access to a file. You need create, read/write/ also need to reposition. What is reposition? In the case of a database, you want to get a particular customer record, doesn't make sense for sequential order. You need a position. I need to rewind the tape to where I can find it. It is a horrible way to store it in a tape sequentially. So I have a hard disk, with a set of platters like a CDROM, and I have these arms that go back and forth accessing a sector of the disk. When I have that particular sector under the read write head, I can retrieve that order. This is called direct access. The operation to do that is lseek. Because read and write are sequential. The operation is lseek. But it is good to know even if you hardly know that. If you have an index to google map, you have access to data map of the world but you won't store it sequentially. You have an index to that file which could use a KDTree. Two Dimensional tree. 3 dimensional, and that structure would have direct access, with sector addresses to the storage. For that data which you are trying to retrieve into data map. So then that's reposition. Memory map has to do with mapping of processes address space into access store. So you would hardly do anything like that. It gives you abstraction of data memory. Your program is waiting, and other programs come to the memory, your programs would have to be swapped out to the memory. When my virtual memory gives you huge address space, then again, that page needs to be fetched from the disk to memory through a sys call.

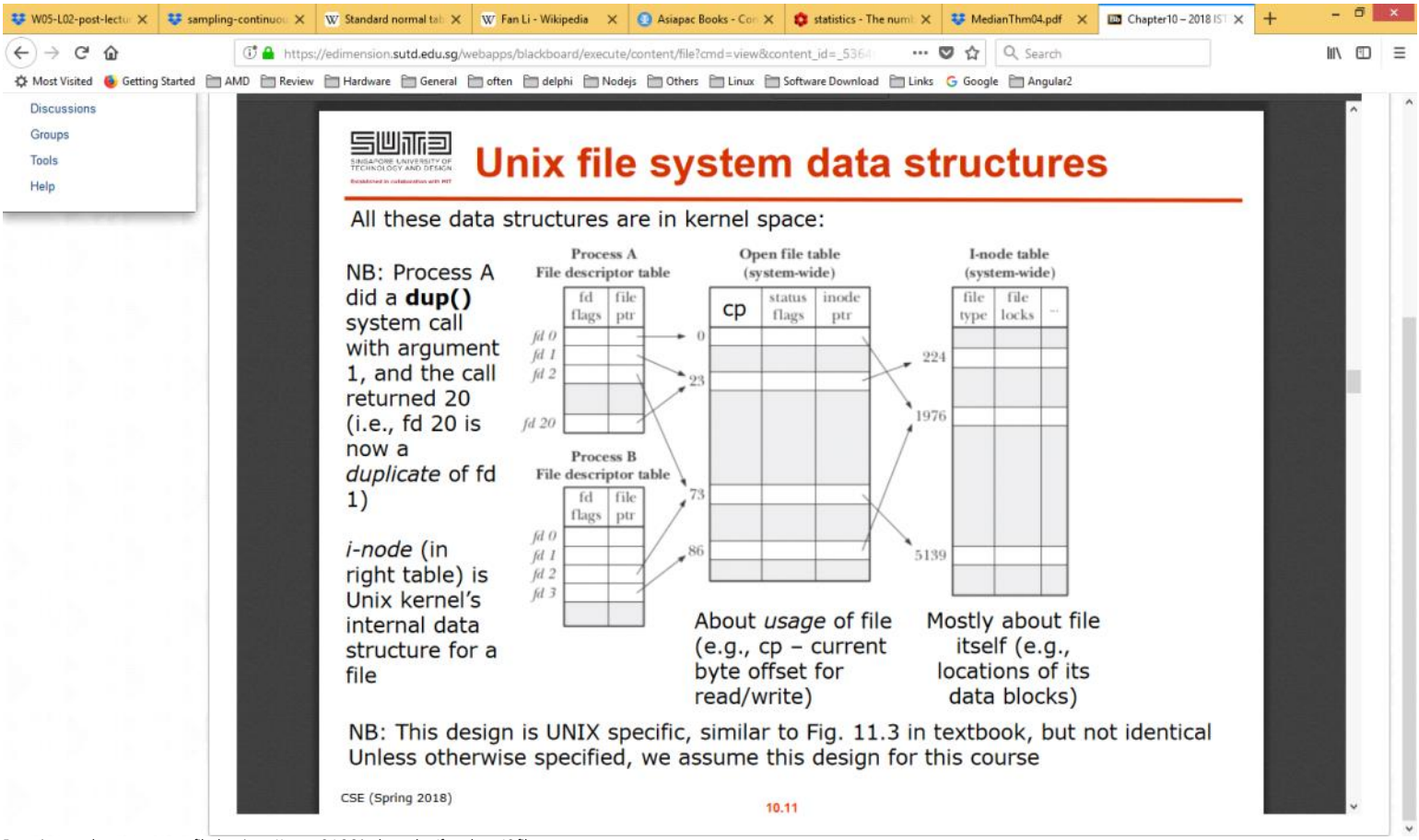
So the important ones are create, read/write/reposition, delete and truncate (removes data from the file and keeps the info for archival purposes. It keeps attributes.)

So what is the reason why we distinguish between create and open? What should create do? Create says I need a new file, need some place in the disk. If there is no place in the disk, create should say that should have it in the disk. We have that file and need to start some session in that file. We have concurrency. Other people could be wanting to use the same file. Operating system wants to know how to prevent bad things from happening. The key word is a session. Open means this user wants to begin a session, so begin reading and writing from a file. We also make sure that user has access write. Often, by explicit open() system call, but can be other methods too. After you have used a file, you should close it. Some users don't do so, OS has to do some housekeeping operations to do that. If the program crashes, OS needs to detect that and close file automatically.

Need an identifier (which is not long). So you need something that the OS can relate to? A number. Even if you use 32 bit number or 64 bit, you can have a huge number of files. If you were to give each file a 64 bit, you need a number. You need to map from names to that internal number. This is done through a file descriptor table.

This is the way it is done. There are basically three kinds of kernel tables, in the kernel memory. Not on the hard disk or on the secondary storage. This inode table contains an entry for each file that is currently being used by some process in the operating system. If there is a file that nobody is using, it won't show up in the inode table. It is a dynamic entity. It is system wide.

The other table is called an open file table, also system wide, and that open file table, is associated with the processes directly. A single process may have two file descriptors pointing to the same entry of the open file table. The third thing is the file descriptor table, unique to every process. This is the basic structuring of the mapping from the process to the secondary storage. Which table has the access to the entity storage? Open File table points to INODE, file descriptor points to open file. The only table that has the sector store is the last one(INODE). There is only one entry in each table for each file. There may be one more entry in the open file or file descriptor table that points at the same entry in the corresponding table. Physically, they are the same file.

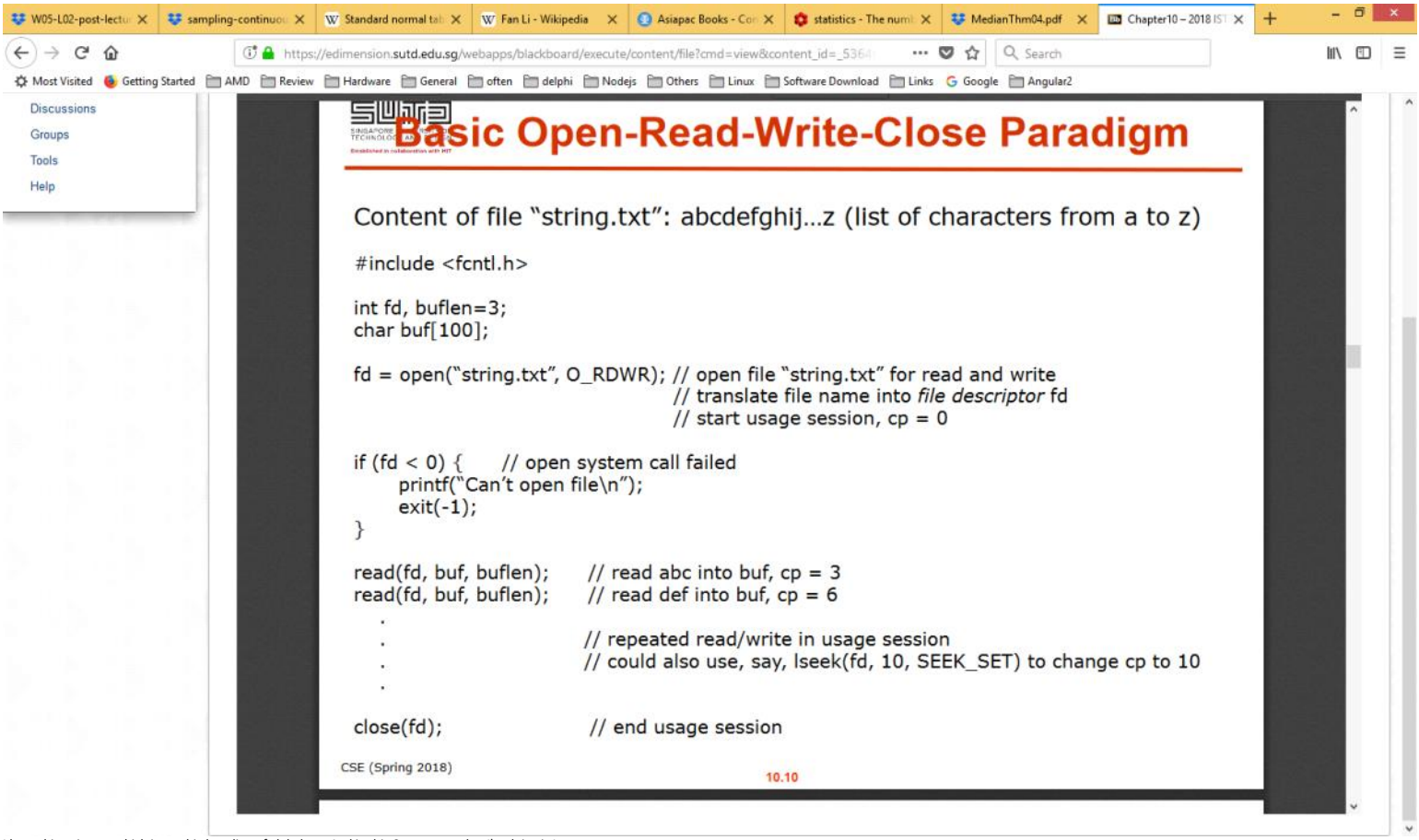


Every time you do open, you get a file descriptor. You get 0 1 2 3 in that order. If you have 10 files, you get around 13. So when you first open a file, you get number 3. Because it is unique to a process, you go sequentially and get the next number in the file descriptor table. As is shown in that picture, your fd goes from 0 1 2 and so on. Process A has 20 files open so it has gone the way to 20.

Fd is the integer index into per-process file descriptor table. So within that table, the index has meaning only in the context of this process. Each fd table entry points to system-wide open file table about usage of the opened (see Slide 10.11). Current file offset (cp): pointer to current read/write location. In the secondary store, there is only one entity. Need only one entry in the INODE table that keeps information about where it is stored and other things about that file. In that open file table, what other information should we keep? By default, file access is always sequential. For every usage, we keep something called a current pointer (a cp). Every time you write, the cp would advance by that many bytes. And when you read you read from wherever that cp is pointing at. When you do the seek, you are repositioning the cp.

This cp is very important for the current usage. Here, this file descriptor is pointing to a file here (23). File descriptor in this address space is pointing to that same open file table entry. On that count, they are both the same. They would have different current pointers because they come from different usage manners.

Access status: mode granted like read, write/append, execute. Open count: how many fd table entries point to it == e.g. can't remove open file table entry if reference count is positive.



I have this string.txt which is one big long line of alphabets. And in this C program, what I'm doing is I open that file string.txt in read and write mode and print out the file descriptor number, then I do a dup command. Dup command duplicates file descriptor. And gives it a new place in a new table entry and duplicates that file descriptor. After that, this is simply an error handled. It prints it out and I will read from that file descriptor and read buffer length which is 3. I will read three bytes. And then take the second file descriptor and read three bytes into its buffer. And we'll see what's happening behind the scenes.

The value of the fd is 3, so third entry (stdin,stdout,error), and because it is duplicate, it is opened at the same file entry. If I read, I am advancing the cp, and read a b c and my cp is pointing at d. Notice when I do the same for the duplicated file table entry, that's reading from d, not from a. Which would not be the case if I do a separate open. I do the same process opening it again. You will reset the value from A. The cp for that would be a.

So this is not exactly this. This only shows you what happens if you do a series of reads. But if you have a duplicate, the duplicated file descriptor would see the advancement of the current pointer. An lseek is a repositioning, and set the pointer to any valid location within that file.

What we have seen is that multiple file descriptor table entries can point to same open file table entry. Many to one mapping. A process can have two or more file descriptors referencing the same open file table entry (after dup() as illustrated on previous slide).

Different processes can also have their file descriptors point to the same open file table entry.

- Child inherits parent's file descriptors after fork()
  - Child gets its own fd table. But this fd table initially has the same content as the parent's fd table
- Fork copies everything except return from the fork. Return is the only thing that is different. 0 in the case of the child and the process id of the child in the case of the parent.

Of course, after you do the exec, everything else is different. Pointers to the open file table will be the same. Even unrelated processes have the different system directories. This has communicated over some socket the fd. I can acquire file descriptor and use the software communication to pass it to another processor.

- Unrelated processes can pass file descriptors to each other e.g. using "UNIX domain sockets." I can pass the descriptor from p1 to p2.

If two file descriptors fd1 and fd2 reference same open file entry, they share usage (e.g. cp) of the file -- read/write through fd1 will advance cp seen by fd2.

In order to get a better understanding, let's do an activity.
Consider the snapshot of Unix open files shown on Slide 10.11. Redraw the tables after the following sequence of actions.

- B forks process C
- C closes fd2
- C opens file 1976
- NB. Open() system call allocates smallest fd that is not currently used

So what would happen when B forks C, everything would be copied. C closes fd2. C closes it, it is saying that I do not need this association goes away. So iahve this empty space where fd2 was not pointing at 73. Now we look at 1976. It is already open in the INODE table. And it is pointed to by fd0. But fd 0 has another OFT entry over here which points to 1976. At this point, 86 was also pointing to 1976 (before open file 1976). So what would happen. Another fd which would get a new CP because we are opening a new Association. Let's call it 88 on open file table. So it would point from fd0 to 88, and this 88 is now pointing to the same inode table entry : 1976.

Does the closing delete the entry in the open file table? Open file table entry deleted when no

references to that file. So c closes fd2. So this association from fd2 to 73 will not be removed because Process A is still using that open file table entry. There are all these things pointing in the case of 1976. This file is different from this file. This file is 51 39. This file is 1976. But if they are sharing the same file. These are different points within the same file.

An open file table entry is created in two ways. When I do a dup I get that. Fd20 is a duplicate of fd1. They are pointing in the same way. I do a fork, I do a child process here. I am actually passing a pointer to that of the child.
The 0 entry would have taken place when it was first created. I pass a file from fd0 (usually std in) and I can pass in some file name Maybe this process, after it was forked, would close that file. This process still maintains the parent association, and pointer to this OFT. Since fd3 parent does not fd3, and this child would have opened a file and gotten this fd3.

Monday

Monday, February 26, 2018 8:34 AM

The open file entry would be new each time you issue an open statement. If it is something that is inherited from a parent, or duplicated, or even passed through a socket descriptor, then it will be the same oft entry. As with all things, try it out. So write a small program, preferably in C. Do a fork, and then try to access that entry, write a couple of bytes into that file and see the current pointer.

So today, we'll move on to the topic of directory. In the beginning, you had a flat file structure. All the users of the system had files in that same flat directory. But that's not a good thing. File system as a collection of files, and a collection of pointers into that. The pointers are the directory structure. In the file system, you have one directory per file system, and all of them are stored in secondary storage. All of them are cached for speed. A good practice, is that backups of the two structures are done on tape as you can have several generals of backup.

Thinking of the files as operations. Another name for directory is folders. The first thing is that we would like to create a file or folder, or create a folder within a folder. We can create a directory. We can delete a file or folder; list a directory (list the contents, folders within) and we can have a -r option where it does so recursively. We can rename a file, and search for a file. We can traverse a file system. Basically, trying to look for a particular file that matches a particular pattern. Find all the names written, and given a directory given a java.

We are just looking a bit too much into storage aspects. Basically, how does a file system map onto a secondary storage. The most we are talking about are disk; not about CDROM, stuff like that. You can have partitions of the disk. Why is that? We would like to limit the size of each file system. You can have several gigabytes and you may not need that much space on one file system. Sometimes you want to partition the disk into multiple file systems and have different OS on each. So you might want to partition the disk. In general, you can think of it as partition.

A lot of times, we would like to go for more reliability and that is done through the use of RAID (Redundant Array of Independent Disks). These are large number of disks, and the idea is redundancy. If one or more disks should fail, could go to an alternate disk because these are replicated across multiple disks. There are a lot of structures; mechanisms to consider.

Operations in the directory. We can create a file or folder. Find just gets me all the files starting from the directory. Find . -name '*.java' will take me all the files named java. -exec grep hello{} \; -print: Looks for files with the word hello, grab between the files for hello and print it out.

Purpose of the directory structure: we would like to have efficient search. It is an efficient way of searching through the file system. This sort of data structure is sorting across multiple data sites. But an undderlying way, there is a underlying structure that maps it across the data structure. Naming: These names need to be persistent, not just within programming language but into the storage structure. Users can pick the same name for user files without clashing. How do you know with only a flat directory structure, people would have only one username, file name restrictions. Same file can have different names. Why is that? Maybe it is a common file shared between users and some users think of it in different ways.

The directory structure is a good way to organize. Logical grouping of files by various properties. People may want to share those of the same type, should be clustered under the same directory. Or I want all my files in one folder.

Single Level Directory: In the main computers. Of course you know that this kind of structure is no good. Two users want to name their file the same name. If Tom and Amy want to have a test file, call it Tom Test and Amy test, and this can get more and more unwieldy, especially with eight characters per file name. This can be even more unmanageable. How do you group certain things? It is not possible. Because there is no logical grouping or organization. A glaring drawback is that there is no protection at all.

Two level directory: At one level, only user names, at the next level, you have files which belong to a particular user. This is an improvement because you can have files of the same name. Both user 1 and user 3 have files called a. You need to know the path. So there are notions of subdirectory (not just one level) and path name emerge. (master file directory and user file directory). You don't see the fact that you could have files at master file. Because apparently master file is only allowed to have directories, user file has only files. So each user has his own separate name space. So no clashes and more efficient. What are the problems with this structure? Very rigid. This allows you some level of flexibility in data, but where are you storing system data? How?

We could copy it into the user directory. All system data could be copied into every directory. I have my own version of the compiler, of the linker. Not a very good idea. So it is very rigid. Some people have proposed a separate user directory; but all the users can see, and you can store all your system utilities into that directory. Is that going to be enough? How many additional user directories do you want to have? What happens when you delete a non-empty directory. Main problem with this is actually sharing. If you are sharing the file with multiple users, how do you share.

The idea of nested directory is totally absent. User can't create their own directories. This brings us to the tree structure directory. Any number of levels: at each point in this directory, the square; rectangular shapes represent directory names. All file names. For example, stat here is not a directory, it is a file. Stat is the name. But spell is actually a directory, mail is a directory, but dist is a file. So this is a good way to store; how would sharing be achieved. What do you share? Normally you share system utilities, system software, so you can just have a programs, sharing a path. As soon as you share a path like /bin/count, that becomes shared. You can have one arm of this subdirectory which goes to the user level directories. You can have more sharable directories across users.

These directories can be traversed quite easily. DFS or BFS. What's lacking over here is a flexible way to share. Or name. A little while ago, it is good if the same object can have different names. Or visible to different paths. I can come down from this path, and it is all mail. But mail is a different directory. I could want my mail/prt/first to poitn to this programs/mail. But this structure doesn't afford me this directory. We can use symbolic links.

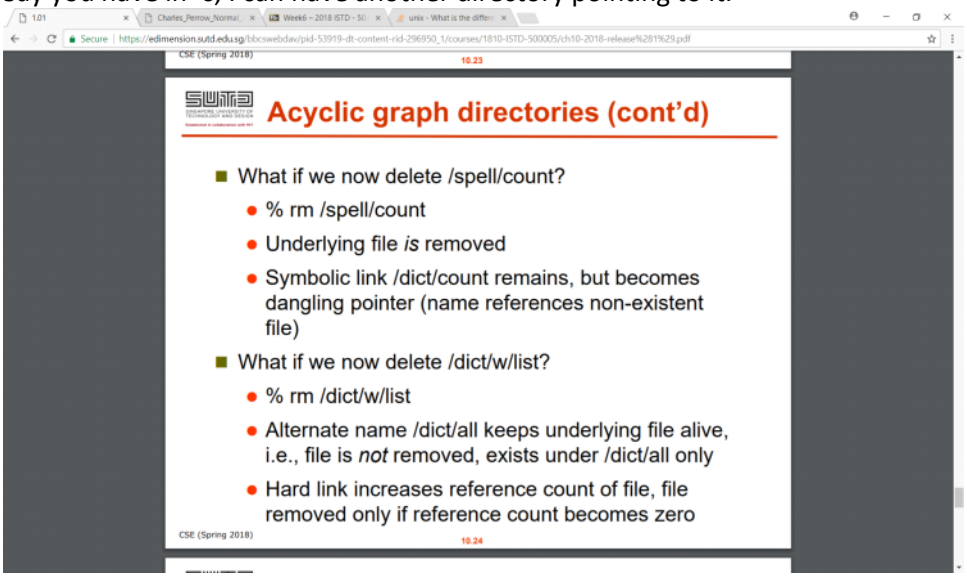
Full path names can become long. You don't always have to use that. Because you have studied linux, so when we have multiple levels of hierarchy, you can have relative paths. So we have this notion of current directory, home directory. The first lab is on this, you can move back and forth and you can move back to home. You can use this name which refers to system utility or binary you want to invoke. I want to look for a particular version of python3.

The system says I don't know where that is? You add to the environmental variables. Variable tells you which directory to search in. You can keep appending to that list of directories, and OS would look for where to find these executables. /bin is natural. There are many user defined directories that the user would want to look in. Shorter relative path names. Now when you do not give that relative path name, it is assumed you are looking at the directory. You can give relative … Notice that each process has its own current working directory. So far you can create a new file or subdirectory within a directory. You can't point to an existing directory/file You can't give an existing file another new name. The way to do this is through the use of symbolic links.

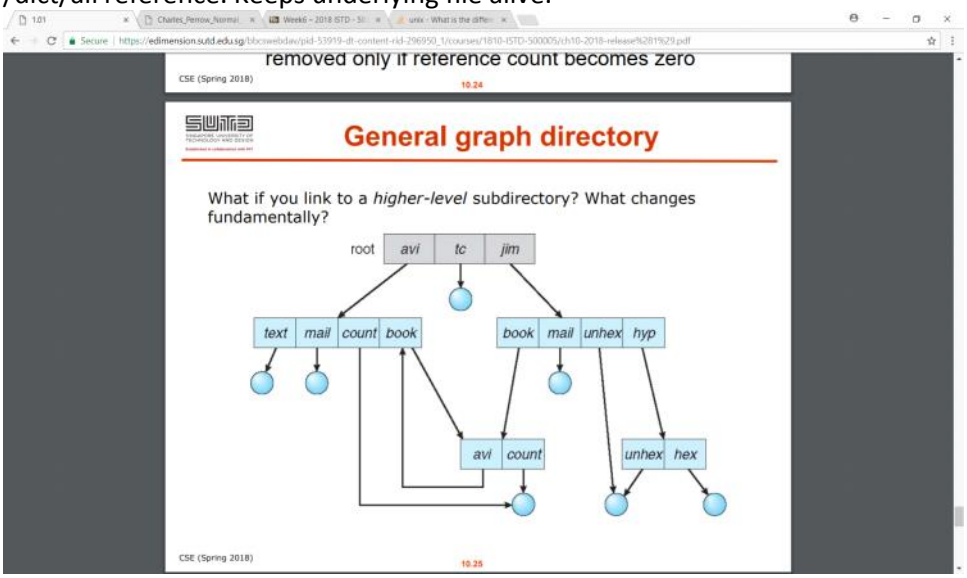
FILE LINKS: We introduce a new type of file system objects in addition to directories and files. This has path name in name space, just like files. We are introducing name space as well. You must hold onto that path name. If that is not deleted, somebody somewhere is referring to that path name. But name links to another existing file system object. Same object can now have multiple names (aliasing). In unix, we have two flavors: Symbolic links (last parameter is name of link): same file now has two names /spell/count and /dict/count. & ls -s /spell/count /dict/count
Hard link is established without the -s : it is a copy of that object. Unless this link gets deleted, the space is not reclaimed. In the case of the slink, the name is just dangling. If the user goes that dangling pointer, there would be no such file.

This now introduces acylicity. Although it is no longer a tree, but still, because we are not going up in the structure, it is still acyclic. It is still manageable because it is acyclic. There is a symbolic link to this object called count. This directory, spell was created first, and an object called count created. Hard link created by ln /spell/words/list all.

Think of it as the hard link does not remove the data until that file is deleted. The hardlink points to the inode while the softlink points at the file. If I deleted a file, s link would not be pointing at anything, while hardlink is another name for the same object and with the same contents.
Say you have ln -s, I can have another directory pointing to it.



Here, there is an example of two links: Symbolic and hard link to the same object. The object is the same. This is a symbolic link from w list and dif you remove this, this object is still accessible to the /dict/all reference. Keeps underlying file alive.



There is a cycle here. How would the cycle be created? Book and avi. So you keep going between book and avi and your path keeps getting longer and longer. Every time you delete a folder, you need to check if this is going.

- General Graph Directory: If you are not careful, you get into difficult situations as cycles can be tricky. Traversal by DFS, BFS etc may not terminate. Reference count won't work because avi points to book, book points to avi. So we need Algorithms to prevent from going to a name file system. Or doing operations like traversal.

Wednesday

Wednesday, February 28, 2018 11:35 AM

An OS is on a particular host, there is not much interaction between types of entity in the hosts. The solution to the challenge of interoperability is the protocol. In an OS, we are sharing. We are sharing disks. But when it comes to a network, what is it we are sharing?

We have several entities on a network, bandwidth. We are sharing bandwidth. Whatever it is, it needs to have a limit. Turns out most of the motivation is happening, but at the core, whatever solutions that are solving. You look at any of the protocols. Telnet, TCP/IP. They haven't changed. But of course, there has been a lot of innovation, happening at the edge. Web access, web based protocol.

What is a network? We sometimes confuse a network to mean the internet. But internet is not the only network. We have computer network. When we say internet, we are talking about the global internet, but we can also have smaller internets. If you choose to use the same protocol as TCP/IP, that would be that internet. Two phones connected by bluetooth: that's not looking at protocol.

When I say there isn't a global internet, there could be military networks which are secure and need to be separated from the internet and have a different set of protocols. The internet was one of the different contenders at a point in time, till early 1990s. There was a bitnet, and a decknet from digital, and IBM had other networks and these were all global. Internet has emerged by and large as the global. There are all those design challenges very well. It could deal with challenges, with interrupt ability, and we could have two computers talking together. It becomes a networks of networks, recursively defined and structure. We still continue to operate the remainder. With the kinds of caveats, of worms, this is questionable. So we have cybersecurity. So coming back to the internet, what essentially are the components of the internet. What is a host?

We use the internet all the time. We use it on devices not necessarily computers. How do we define a network? There is no clear definition.

The first view we have : we'll be using about the internet when we say network: there are two views we would take. First is a handson, hardware oriented. Nuts and bolts component view of the Internet. Second would be a service view. When we look at the nuts and bolts view. There is an access network. What are the entities that would be accessing the network? Predominantly, hosts. We will not talk much about that because the manner in which devices access the internet is similar to hosts. For purpose to the theoretical model, it's enough to talk about hosts as your accesses.

What happens in your hosts that distinguishes them as entity on the network. This are the periphery. So what happens there? If you think about your laptop, or maybe your server. For example, we have a laptop here. And we have a server here. And the laptop is going through the network to access the server here. But inside the components that are within the network, they are not hosts. They are switches. What happens in the end host they are running applications that are network applications.

There are applications that are not network; those are the ones we studied in the first half of the course; system, user applications. The browser actually invokes certain applications that are network applications. When a browser is running, it is presenting information to the user. But then another part of the browser that fetches information from the server. There's something else on my laptop, which is presenting information that has nothing to do with the network. It fetches a page. So those are the network applications. What is the protocol we normally use? HTTP. It's an application layer protocol because it runs on the same level on the other applications; but it is concerned with the network. Hyper text Transfer Protocol.

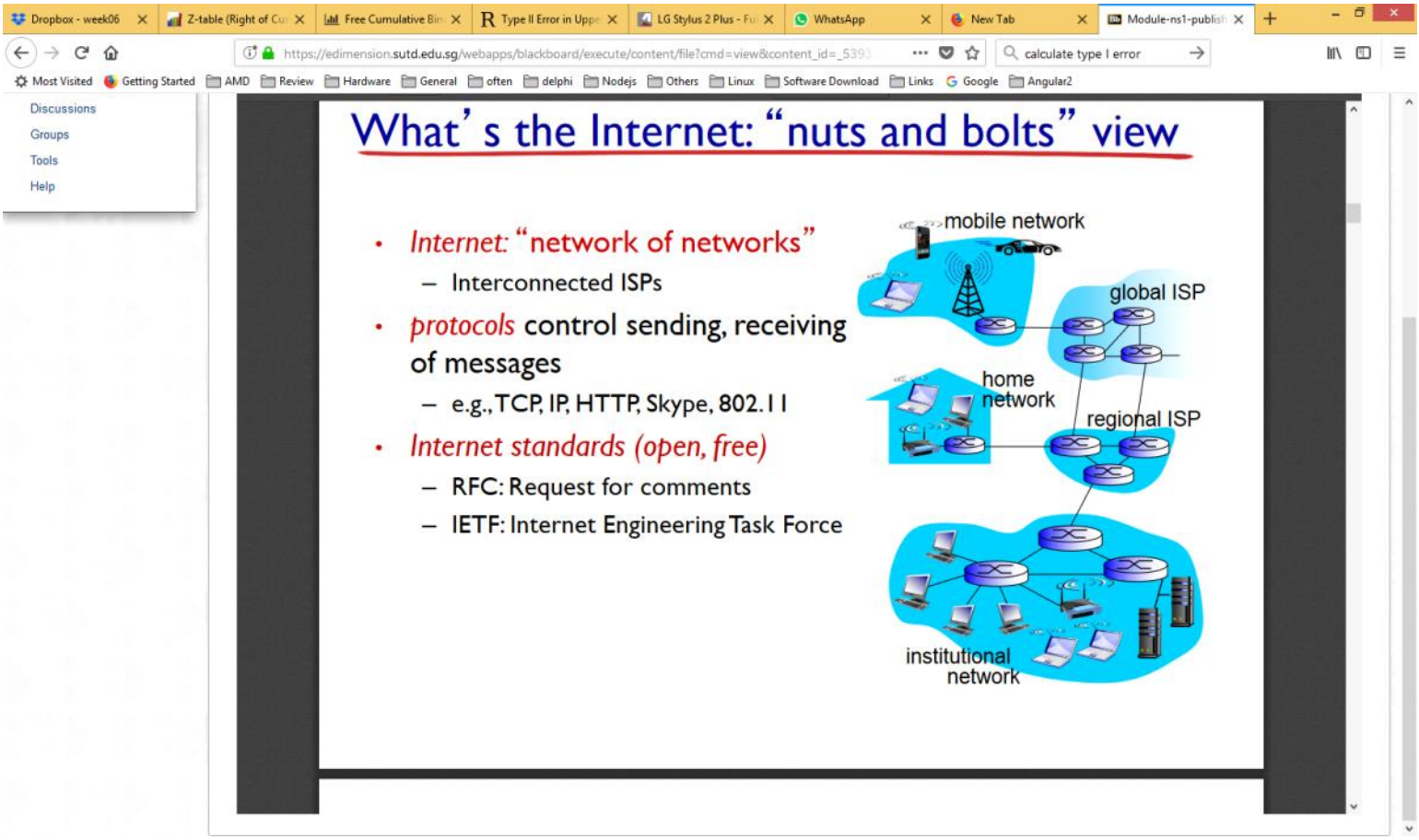
They may be built in a layer fashion. The perspective taken in this course is a top down approach. This book was introduced and immensely successful because previous courses were talked from bottom up: physical, data link layer. Now there are only five layers. That was not interesting. Now we talk from application down. Why is that interesting? Because we use applications all the time.
In order for web browser to work, we need to see what's happening under the hood. Coming back to the nuts and bolts view, there is the view of the hosts as components of this network. There are communication links. What are the communication links in the network? Many of this terminology, as well as abstractions come from our day to day life. For example, hosts as cars or as people. People need to go on a high way, they need hosts. They are interconnected. Without roads, we cannot go from place A to Place B. Roads has intersections.
Routers are interchange points, because they direct traffic. So that's the second component: communication links. In fact, free space is also a communication link. We have transmission taking place through whatever electromagnetic waves you use to transport the signals. Then you have guided media; copper and signals can wave to guided media but with better ways for navigation. So we have fiber, copper and unguided for radio, satellite.

Bandwidth: most important measure: because given the bandwidth, what's the given level of the media. That's training media of your communication link. Looking at the nuts and bolts picture, we have our switches. We need our switches: the intersections on our highway. We would consider routers and switches to be the same: They take information in packets and switch them from source to destination. In the beginning, when communication was taking place, from across the world, it started with Telephony.

Telephony was largely analog, for the most part. For the next seventy eighty years, telephony was analog. The big thing was packet switching: information was converted to digital so that it could be packetised and put through a computer. So a router behaves like a computer, it takes in digitised information and decides where it goes, behaving like an intersection and forwards it to the next hob which could be the server in that case. So we have routers that can route packets.

It's a big step forward in a sharing of our media. So this is the sense of nuts and bolts: these are the pieces of hardware like the hosts, the communication links, being various types of guided media where signals can propagate, signals are digitised mostly but fiber optics are still light. At the very bottom, where the tyre meets the road, everything is still analog but we make a deal about digital because it is easy to convert signals to digital, maintaining quality.

Store and forward is analog. But let's not forget at the end of the road, we are dealing with analog communications. So let's move on. I mentioned hosts but there are non obvious hosts. An IP picture frame: runs internet applications because it downloads a picture from the internet. You don't log into it. You can have a toaster to access the internet and toasts a nice pattern on your bread. Then you have the refrigerator that knows when to order eggs or milk. Then you have internet phones.



How they have another design challenge of scalability. Today's internet is huge. It is mind-boggling. It is billions of users. The first generation of IP devices, that was only 32 bit addresses. The next generation is 138 bits. This internet is running with 32 bit addresses. We will take a look at subsequent lectures. Second thing is this notion of protocols. How did protocols come into being? Protocols help us to look at the problem of interruptability. This is pretty much the same as human communication. We have certain protocols that are open and without any restrictions. You can write your own version of TCP. There are certain protocols that are closed like SKYPE. It is a completely closed proprietary. By notion of how the skype entities behave. There are others such as MPEG. IT is a data compression protocol for videos but has a lot about networking. Because when you send it across the internet, data compression is closely related with standards.

Then again, you have 802.11. Some of those standards have licensing fee. It covers ground with many types of wireless access. There are standards discussions going on because there are newer ideas coming into being. The way protocols come into being is through RFC. It's through a request for comments. You may have encountered that before. You need to approach the engineering task force, containing the idea of RFC, joins the bandwagon and soon you have the internet standard. Many of them are used all the time like TCP and HTTP and so on.

When it comes to a service view, we are talking about services. We are a service provider. When Singtel offers you to the network, Singtel gives you services. You take it for granted that you have a quality of service. This notion of the network as a provision of services, it is something built upon. The reason why you have voice over IP as a service, you can have electronic mail, a user of electronic mail, may not even have a network behind it. These are all various types of services, of infrastructural services. The second thing as the service view is the view of APIS.

For a long time, when the network was being built, the capability for a high speed internet was there all along. People could build networks with GB speeds but there were no killer apps. Killer applications are so bandwidth hungry that it needs a lot of bandwidth you need to engineer your network to meet that bandwidth demand. The killer apps didn't arrive until we write our own apps on mobile devices. This caused a huge surge in demand for network capacity. So why did the apps take off?

If you think about it, what are the key things that led to the capability of you and me writing apps. Browser set applications and so on. The fundamental API is that of a socket API. Going back to operating systems, you need system calls. We studied system calls in great detail. Many system calls we study what would be the equivalent of system calls in a network. There is none. There is no notion of a system call for network.

The socket primarily has to do with communication. When you have a socket, you get the ability to reach out to some other hosts somewhere else in the world and deliver packets to the host and read packets to that host. So this basic ability to communicate across hosts is established through the socket api. Once you have the socket api, you have the ability to have TCP connection, you have HTTP connection, and the web interface and other APIS on top of that. Without that API, you would not be able to build apps and then we would not have those killer apps. So what essentially has happened is the power to produce applications is given to the end user.

ISP give you interconnectivity and services, but no killer apps. But the people who make killer apps were the end users, writing apps and making use of services provided by the internet. We are given these nuts and bolts, so how shall we do security.

So we have two challenges, interop and sharing. What are the principles to overcome these two challenges.
The first challenge: that of interoperablity. When it comes to solving design challenges, it is good to look at how we do thing on a human plane. For example, we want to innovate in building airplanes. We have to have some way to communicate, and that's a language. But is it capable of communicating meaningfully; they must convey meaning and they must also convey intent so to speak. We might

gesture to him to get his attention. By virtue of the fact I have language, I must have convention: a protocol. You must recognize the word 'hello' means something, and embark on a kind of communication. If you are from a different culture, you would probably absorb me.

Our communication has not gone very far. Computers can become too easy, go to an infinite loop. You need etiquette, manners and things like that. We don't communicate on a human plane. Similarly, when it comes to computing, we also need to have a language. What would be the equivalent of language. It can't be human natural language, so what would it be?

Sometime of coding. So as I said, everything is analog signals. When one send analog signals, it means nothing unless you encode information through analog signals. There has to be an agreement. This sequence of 1s and 0s means such and such. Not only agreement, but agreement on every intermediate node, whether a packet switch or it is a core router in the heart of the internet, they all have same understanding of what ones and zeros mean. If you don't, you can't have a network. Coding scheme. Which is well understood and well accepted but all entities are connected to the internet.

After having fixed the language, how do we fix the conventions. What is a convention? Set of rules. It is a set of rules which is accepted mutually. Geneva Convention: peace treaty between nations. They all agree to the rules, it is a mutual agreement. You can break those rules and you don't have peace. So a convention is a set of rules that all entities adhere to. So for the case of computers, we have protocols. Again the word protocol is given from parlance. When I am visiting this dignitary. We have interdoctal protocols. Otherwise, we don't have good interaction. Similarly, we have protocols in the case of computers. If I need to communicate, I need to send a packet and wait for acknowledgement, and the receiving entity needs to send an acknowledgement back.

There is a beginning message, and an acknowledgement, not in all cases. So there is an acknowledgement, and when that acknowledgement is accepted, the message is sent. There is a standardization body that looks into the protocol that are being proposed through a procedure note as comments. There are protocols we have a voting procedure and once everybody decides to go ahead, it goes in. HTTP, TCP UDP IP ARP whatever-P are all approved to go in.

You can also think of a protocol as something that attaches a header to a payload. For example, at the human level, when you send a letter to a recipient, what do you do? You put the letter in an envelope and you write the recipient name and address. The postal department will not accept the letter and the important thing here, the postal department does not interpret contents of the letter but the address of the letter. In a similar sense, the network does not interpret data we are sending but the address we are sending it to. As an end-user, we don't say that this is a letter for the united Kingdom. First send to this postal department. The postal department knows how to figure out the routing information. Similarly, in the case of routing network, I as the end user does not need to provide routing information but there is a set of network within the network that provides routing information. When I send through the domain name server, we know which is the end address of the end user and the router knows how to route through that IP address.

The key thing is the packet has the H and the payload which is forwarded to the next layer. For example, I ask what's the time, or I ask a question and you could have some introductions. Communications need to be known to whatever. The communication activity takes place. Rather than humans, the machine forwards a packet or makes a request for the networking service. The protocol defines format in order of messages received. And takes action. I first introduce myself before asking a question. In the case of networks, the protocol must first order the messages in a particular way and define format, and actions must confirm to that protocol.

This person says Hi and the other one responds with hi. And we take that to be a consent that we are willing to talk to each other. The person could have ignored it and walked on. So once we had that acknowledgement, we say do you have the time, and you say two o clock. In the case of a computer communication, the web browser starts a connection request, because it is an HTTP connection and the web server says yes. And then I send my request with a get request with a particular web page, and the web page that contains this file sends it back to me. The file could not have come at this point. On the other hand, I could not have said TCP connection and appended the request at this time. The protocol has been defined to set up as a connection first, and submit a get request from the server.

Any other human protocols?
Sharing:

You have many manufacturers of computing requirement, people assign meaning to different things. Without protocols, you can't have communication taking place. The second challenge is that of sharing. We have very simple model here, it's a powerful model, this model is that of a switch, you might think of it as a router, and managing the traffic from many users and putting it to an outgoing link to the internet. Say that this link is 1 MB per second and this 1 MB is shared by many users. As I said, it's a simple model, but it is an important model. For example, you can see the traffic that is coming on that link has to be processed here.

IT is called processing delay. There is also a queuing delay associated, And an forwarding delay forwarding packets out onto this link. Coming back to this 1 MB per second link how can we look at the problem of sharing that link by all of these users. Let us look at day to day life. For example, you have a meeting room and we want to use the meeting room. How can we do this? One of the ways to do it is I just come into the room, and I grab the room for myself. I reserve the occupancy of this room for a certain amount of time.

I can make the reservation for a particular time slot and nobody else would use the room. I can make a booking of that time slot and I have ownership of that time slot. How would that map? I can divide up the time on that link so that say for every 10 seconds, I would have 10 input services and each user has 100 miliseconds. So that would be a way to divide up this time between my users. This way of sharing is known as TDM (time division multiplexing). Each N user takes 1/N times of available bandwidth.

Is it a good way for the kind of trafficking I am talking about. Time division multiplexing is suitable for phone calls. Fixed rate for each established call. We can argue that the traffic is going this way or the other way. This amount of bandwidth is always kept busy. So for that it is okay. However, when we talk about internet, we are talking about data traffic, not voice traffic. There is difference between data and voice? What is typically data traffic? I make sure that data traffic is cancelled. Do I cancel my voice? No, because they charge me differently because data traffic goes over the internet. I would need to provision separately. Data traffic is inherently burstly. If we use the internet, we use it for social applications, for electronic mail or for web browsing. Do we have constant traffic?

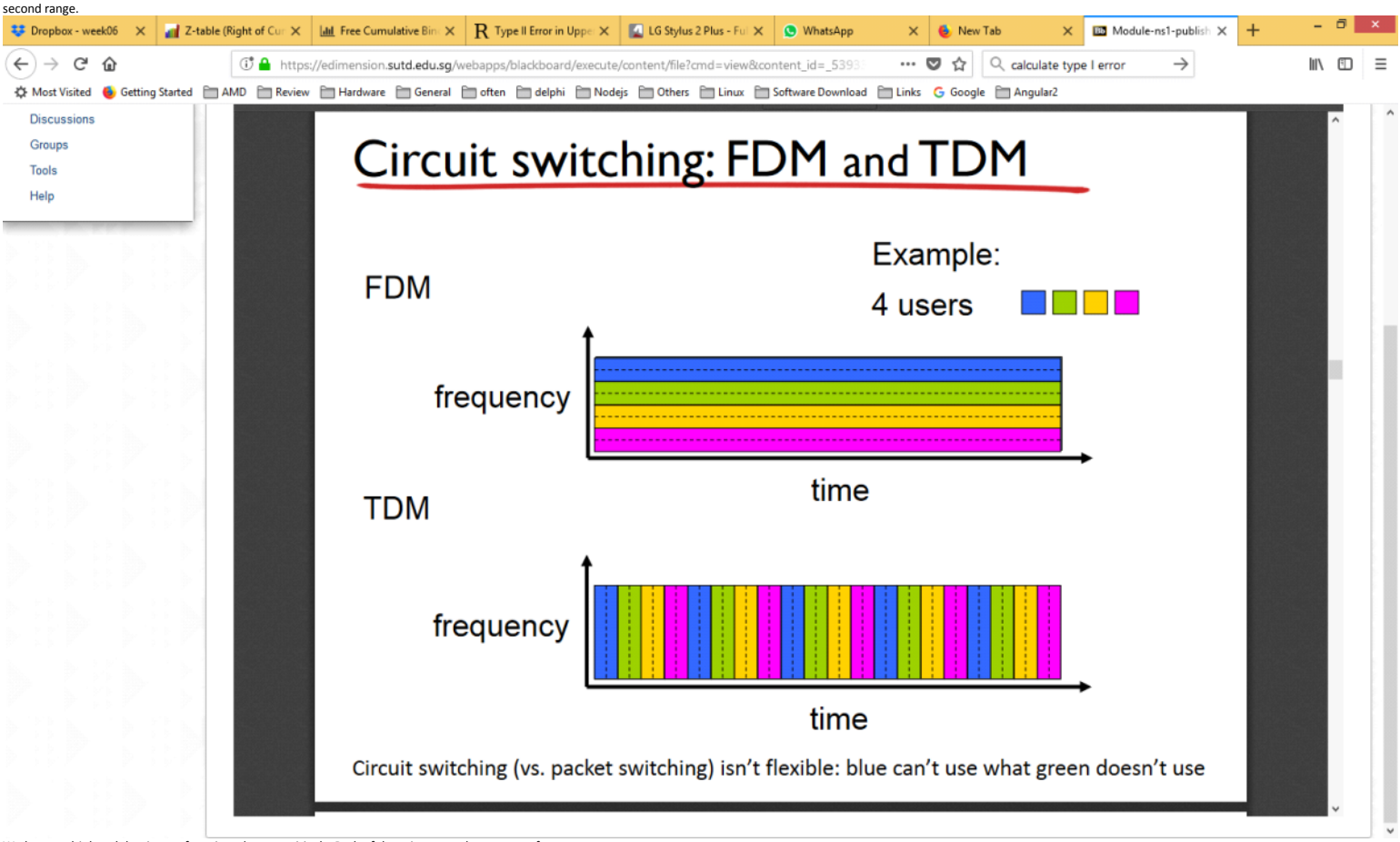
Let's take social media. I'm sporadic; sometimes I'm thinking or doing something eles. It is not constant traffic. Stored video is when I am taking it off a server, and live video is when I am taking things life. If I'm browsing a web, that is bursty because I visit a website, I do a mouse click and I do an object and network gets some time spending some on a web page but I stare at a web page for longer. So when I mean by bursty traffic: How do you surf? It doesn't make sense to provision through a time division multiplexing. I am tying up 1/10th of MBPS all of the time.

The big difference is between circuit switching and packet switching. From the time of AG Bell, to fifty years ago when digital switching started. We have to establish circuit from source to destination, before we start. But with packet switching, we have to propagate the traffic and packet goes from hob to hob and hope it arrives in time.

Now, let's look at how circuit switching works in real life. You have two ways, one is TDM, the other is

frequency division multiplexing.

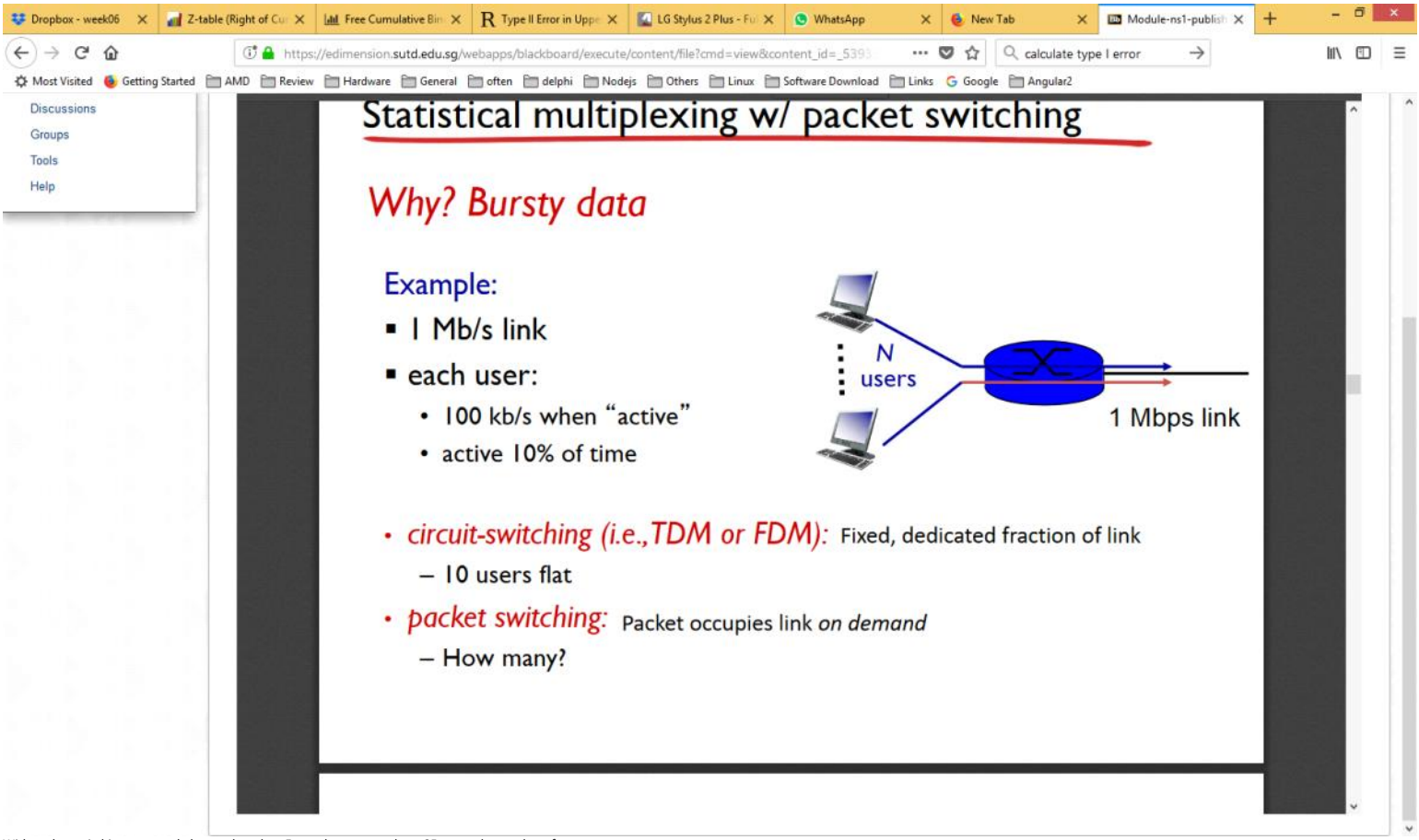
For FDM, you have the entire time to you but you don't have frequency range of communication channel. That may be enough for 4 KHz of range. But for TDM, you have entire frequency spectrum, but only for one time slot.
Say we have four users, you have all the time available for each user but the frequency has been divided to four colours, and the first user gets the first range, that many KHz and the second user would use the second range.



We have multiplexed the time to four time slots repetitively. Both of these importantly are ways of circuit switching. Analogy is analog telephony. In those days, circuit has to be set up. What we would do is raise the phoneset, dial the connection and the connection is established from source to destination throughout the network and only then would voice communication take place. Both of these are data switching.

In the case of packet switching: Packets are being multiplexed, but time and frequency not being multiplexed. Let's have an example, MbPs link. With 10 users, and each user would have 100kb/s when active. Active 10% of their time. Each user requires 100 kilobits per second. But we don't know when the user is going to be active.

With circuit switching, we can only support 10 users. Because we have a fixed, dedicated fraction of link.



With packet switching, we can do better than that. Even when you can have 25 users, the number of users being active is still smaller than 10 because of the statistical multiplexer. It is unlikely that all 10 users are using at the same time. P = 0.1. As you just said, what is the probability that all users are using at the same time. That happens to be very low. All 10 being busy at the same time? 0.1^10. All users at the same time is unlikely. That's where we get the advantage of statistical multiplexing.

35*34*33*32*31*30*29*28*27*26*25 = 1E16
1E16 /35! = 0

Monday

Monday, March 12, 2018 8:34 AM

Last time, we looked at the view of the internet, nuts and bolts view as well as the services view. How the internet provides services to the end user. We think about the internet in terms of the design of the internet. One of the biggest problems is security. Network security has now isolated to cyber security, and because of the basic decisions taken at the time of its design. Because at the time of its design, it was a completely trusted network.

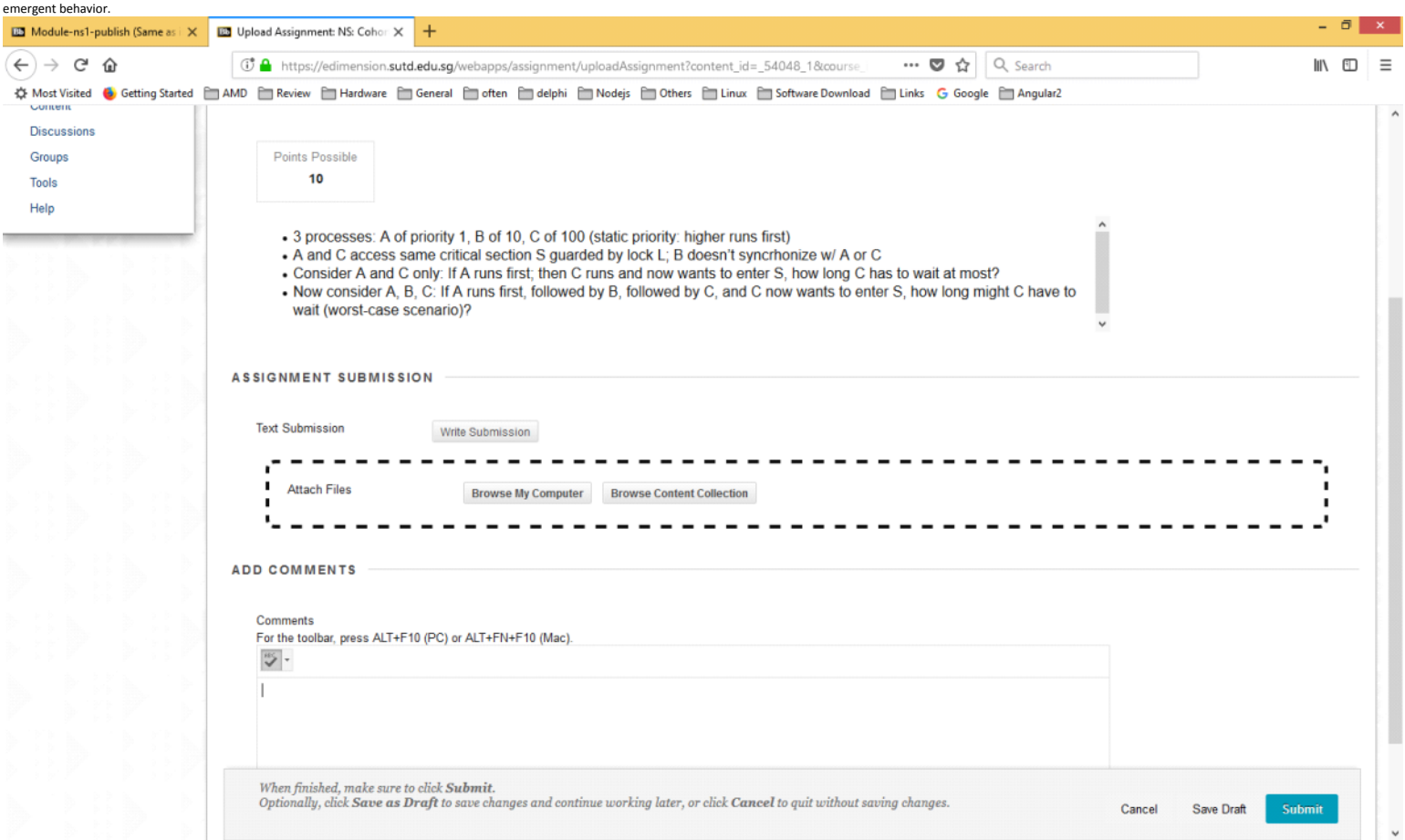
When we look at the network today, let's look at it from the point of view of its design. The first challenge was that of interoperability. How do we operate between different manufacturers? We introduce protocols: an accepted methodology, pattern of interaction, steps and so on. The second challenge was that of sharing. What are we sharing in a network? Transmission media. Share devices and not. The primary thing we are sharing is the media; we are looking at circuit switching (traditional way with telephones) it was an end to end line that was created and naturally, when computers start interacting, that was the model.

But we discover that circuit switching was not very efficient, because the way computers interact and the kind of traffic that goes between computers is packetized and bursty. The circuit switching model tends to waste resources. We did an exercise to see how 35 users. We can go 10MB on one line and the probability of collision, of more than 10 users accessing these lines at the same time. It was very low. That's the reason why internet works today in such a nice way. Because of this phenomenon of statistical multiplexing, of bursty traffic. Today, we shall look at two more challenges.

We will look at the challenge of complexity and we will introduce layering; how the internet networks have solved this problem and addressed it through layering. We will look at scalability, because there are hundreds of devices. It is a huge population. Accessing the network: how would you address this issue? The internet: it is a network of networks and we will move into the network performance. We will look at three aspects of network delay, loss and throughput.

Next challenge is that of complex interacting components. What do we mean by this? If we look at the network, there are so many components involved in the network. Like routers servers and so on. This is extremely complex. To get a handle on this complexity, we look at the way we solve complex problems in the real world. We have a large problem, divide into a smaller problem and we go into smaller problems. We do that in algorithms: we break it into modules. In software engineering, we have modular approach. We have smaller functions. There is something that happens that does not happen at level of modules. To give you an example, say we design a large piece of software, we go about through good engineering principles and each of the modules happen successively. But when we put it together it doesn't work.

That's not only in java and in python, it happens in the real world as well. Sometimes we would have catastrophic failure.
When they integrated the old system, there were parts that interacted and failed. Interaction between modules produces new behaviour which we call emergent behavior. It is something that happens when we put modules together. Such as putting new chemicals together. Let's look at a particular example of emergent behavior.



Let's say A takes k units of time. So C will wait till A finishes process in the critical section. It will wait however is happening, C has to wait until A releases the lock.

Second step. Consider A B and C. B is in the picture. A runs first, followed by B and then followed by C. C might have to wait in A + B time, but will cut in after k units because B does not synchronize with A and C. Remember priority to one is the higher priority. So maybe in this case C cuts in after A (As C > B in terms of priority) and the program runs A B&C. Cr

Because B has higher priority than A, B would continue to supercede A, and A would never be able to release the lock. C is unable to enter its critical section. Even though A is of higher priority, holding resources C needs, B has entered the picture. This would never happen if there were interactions with mutual processes. Static priority scheduler has created emergent behavior.

Potentially forever for C.

This is to show with complex interacting networks.

Back to the question of networking. Think about the problem of transporting bits from one place of the world from another. There are hundreds of thousands of airports, how are you going to solve that problem? We are trying to design a means by which bits maybe across the world. Let's look at the problem of airline transportation. We can look at sea transportation. Airlines is a bit better defined. When we travel by air, we purchase the ticket and we can't proceed to the next step: baggage check. We can't proceed to the next step; loading at the gate. Once the gate loading process has finished, the runway process has taken over. Once the airplane is in the air, there is the problem of routing the airplane from one to another. This airplane routing takes place. On the receiving end, the is a reciprocal set of event sthat take place. There is routing, landing in the runway, the gates (where the unloading takes place) baggage claim and ticket complaints. We can see that the travel that sort of application can be looked at as a sequence of steps whereby one makes use of a service that comes below it.

Coming back to the problem of networking: without any bias to the network today, how do we ship millions of bits from one part of the world to another: we can think about it in this fashion. As opposed to how do we take something on an application layer and then make it go through some bit manipulation at the low level at the other part of the world if we were to look at it as a series of steps to go through at every point. Steps of services at different layers. You begin to get a handle at this problem.

Instead of an n square process where each n level interacts with n-1 levels, every level, the entity on the sending end interacts with the corresponding identity. On the other end. So you only deal with entities on the same level of priority/application interface.

There are seven layers to five layers in the internet. The original side had a presentation layer and a travel layer.

Each layer is like a module: it implements a service: via its own internal layer actions. You may or may not: You may not have the other services because the routing takes place at intermediate sites: makes use of services at the airports. What does that: they travel over well known places. They travel across well known places. Because they need those intermediate sites where routing is available.

Layer I uses service of layer i-1 to provide service of I + 1. The question here: N modules: how many possible interactions if they interact freely? If the ticketing level doesn't know about the level here, it would need to check with every level to make sure that the customer's travel is as smooth. By layering, we bring to the O(n) time.

It has five layers: fifth one which is physical is about pushing bits. The basic entity of that level is bits and purely hardware thing. So we will be mostly concerned with application, transport, network and link layers. So the most of the activity that takes place in networks is at the applications layer. Applications layer would run on n devices; on computers and on laptops. Alll the n entities would run application layer services. As well as applications. As we said before, a user would make use of services at the application layer. We have look at hypertext control protocol. There are also simple layer transport protocol that handles electronic mail. File transfer protocols: travel from one station to the other. There are literally hundreds of protocols in the layer in the internet.

At the application layer: what are the entities that are transported. They are arbitrary sized layers. Transport layers provide TCP and UDP. Transmission control and user data protocol. TCP guarantees delivery of packets. Say there is a huge segment, 1MB of information to be handed over. The transport layer takes it and makes sure it has been put to the other side. That's why it makes use of guaranteed delivery. I will handle the reliability issue. I want my packets to handle as fast as possible. The UDP is a service that is no frills service. It is no frills, it goes and dies and I don't know anything about it. The reliability has to be handled by the application layer itself. You need to make sure that there is acknowledgement on the other side.

The sender at the other end can blast. This case, the packets are going to the destination level and they will get lost because it is just too fast for the receivers to display on screen. The receiver says that you are sending too fast for me. Sender slows down and signals transmission between sender and the receiver. The flow controller balances the rate of the flow so that there is no packet loss and it is comfortable receiving at the other end.

Congestion control has to do with the congestion of the network. You should not be congesting some more. Receiver can receive very fast. If I blast more into the network, it becomes more congested. Flow control, congestion control, fragmentation. They will need to be chopped up into smaller segments because network layer cannot handle. These are the basic functions that the transport layer takes care of: it uses segments/packets.

At the network layer, what's the main issue? We back up: application layer is transmitting large messages from one side to the other, could be mp3 music and maybe skype. So transport layer is reliable or unreliable. There are other functions like flow control, segmentation and fragmentation. Network layer, what is the main purpose? Intermediate points tell you where to go. It is routing that networks takes care of.

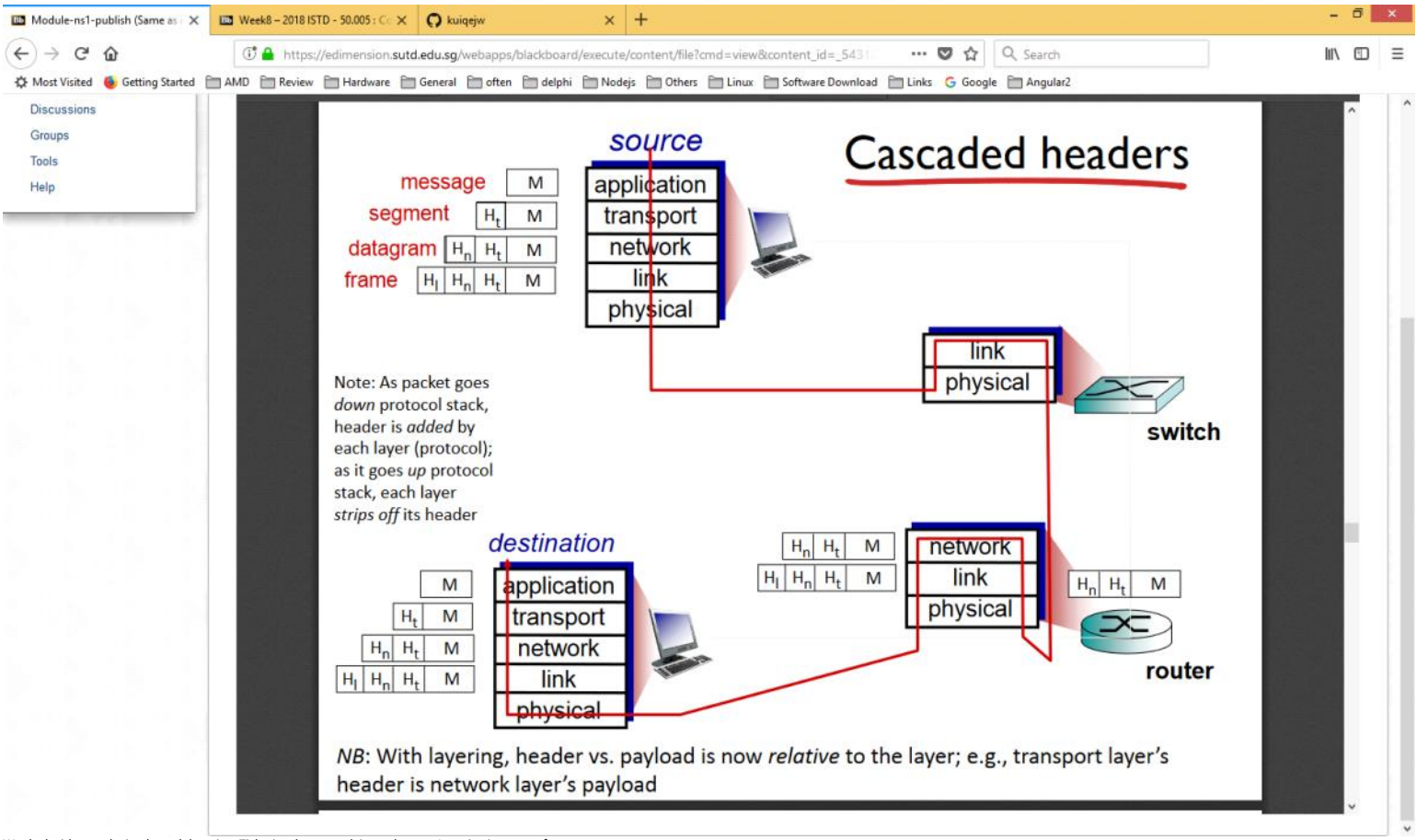
Huge variety of network protocol. You can have multicast: routing protocols are at the heart of the network.

Then we come to the link layer. At the network layer: it is end ot end routing. Link layer is the next hop. Hardware and software comes together. Transport layer is on the software layer: chip does not have TCP protocol. On the network layer, they start coming together. Link layer is the network interface layer. You can get transceiver, network adapter. The main concept of link layer is to take that frame and push it across to the next link and uses a different set of layers because we are not using IP addresses but MAC address. We have 48bit internet/wifi addresses. Mac addresses used at the linked layer. And then finally we have the physical layer; bits on the wire. You can have end to end transfer where links travel on different protocols. One hop over wifi, then over ethernet. Then ethernet over fibre.

The delay would be slower over these segments. How internet traffic gets shipped across back and forth. That's the physical layer. Once again, at the application layer, we have arbitrary sized segments. And then at the network layer, we call them datagrams. And then linked layer we have frames. Linked layer between hop by hop and then bits. IF there is any protocol issue, bits have gone malfunctioned, the bit layer gets taken care of at the media control layer. If we have a linked layer control, and these are two sublayers within the linked layer.

We talk about the envelope; when we send a letter from one to another. And I send it off. Something similar happens at every layer of the internet protocols lab. In order to send that message, the application layer would need to send it to the transport layer, and the transport layer would put a header on top of that transport message. And address information and so on. So that's your segment. And then the transport layer sends it off to the network layer. Network appends its own header and the segment becomes a datagram. Then the datagram goes to the linked layer. This becomes the payload of the next header. There is a header attached to the payload. This becomes a linked layer frame which is

passed down to the physical layer. At the level of the switch, it bypassed the link layer, strip off and examine its new destination before forwarding to the next switch. Difference between router and the switch is that the routing functionality happens at the next link, and at the next step, it would strip network and link. Looks at its destination address and will add a new header to it. And push it over to the physical layer.

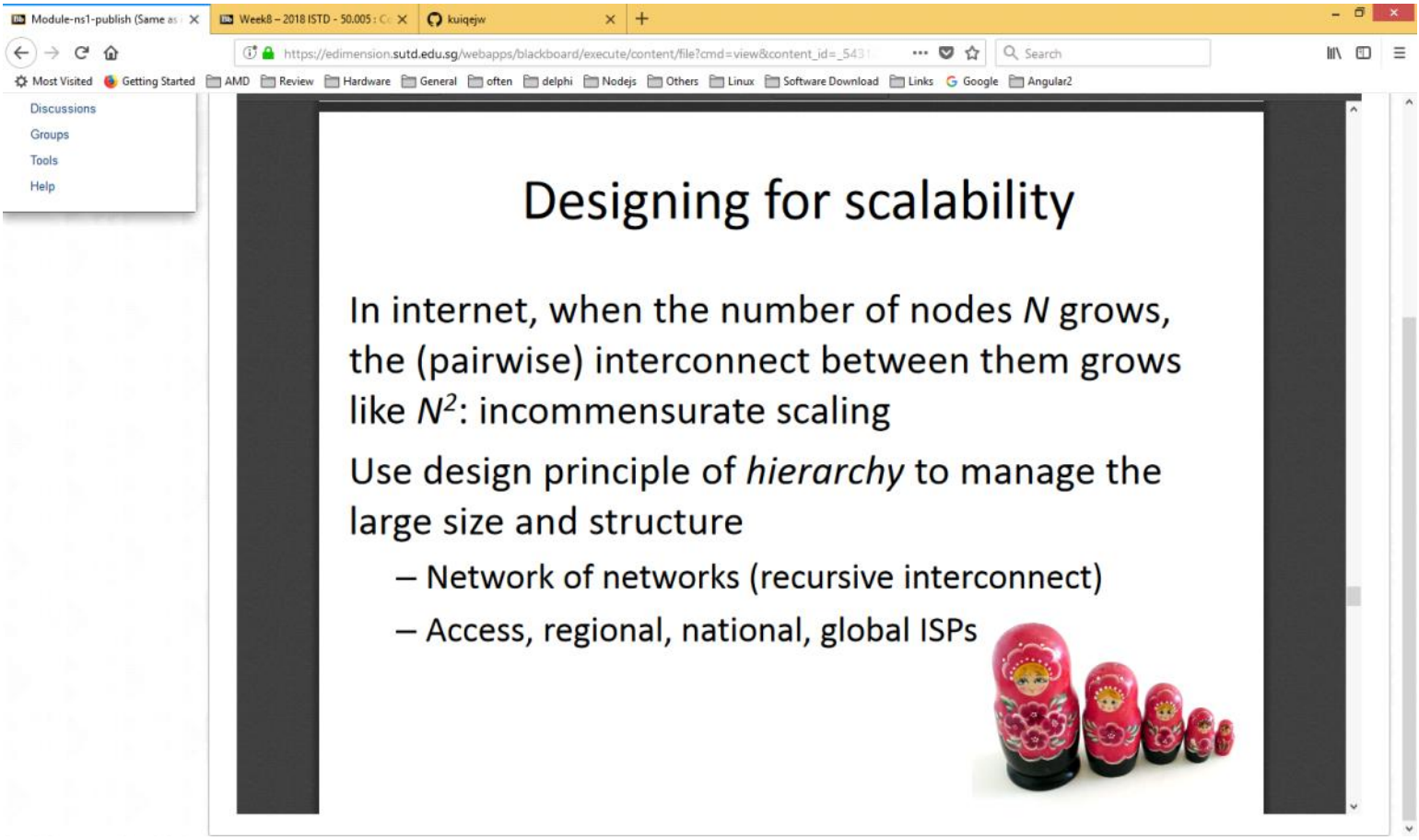


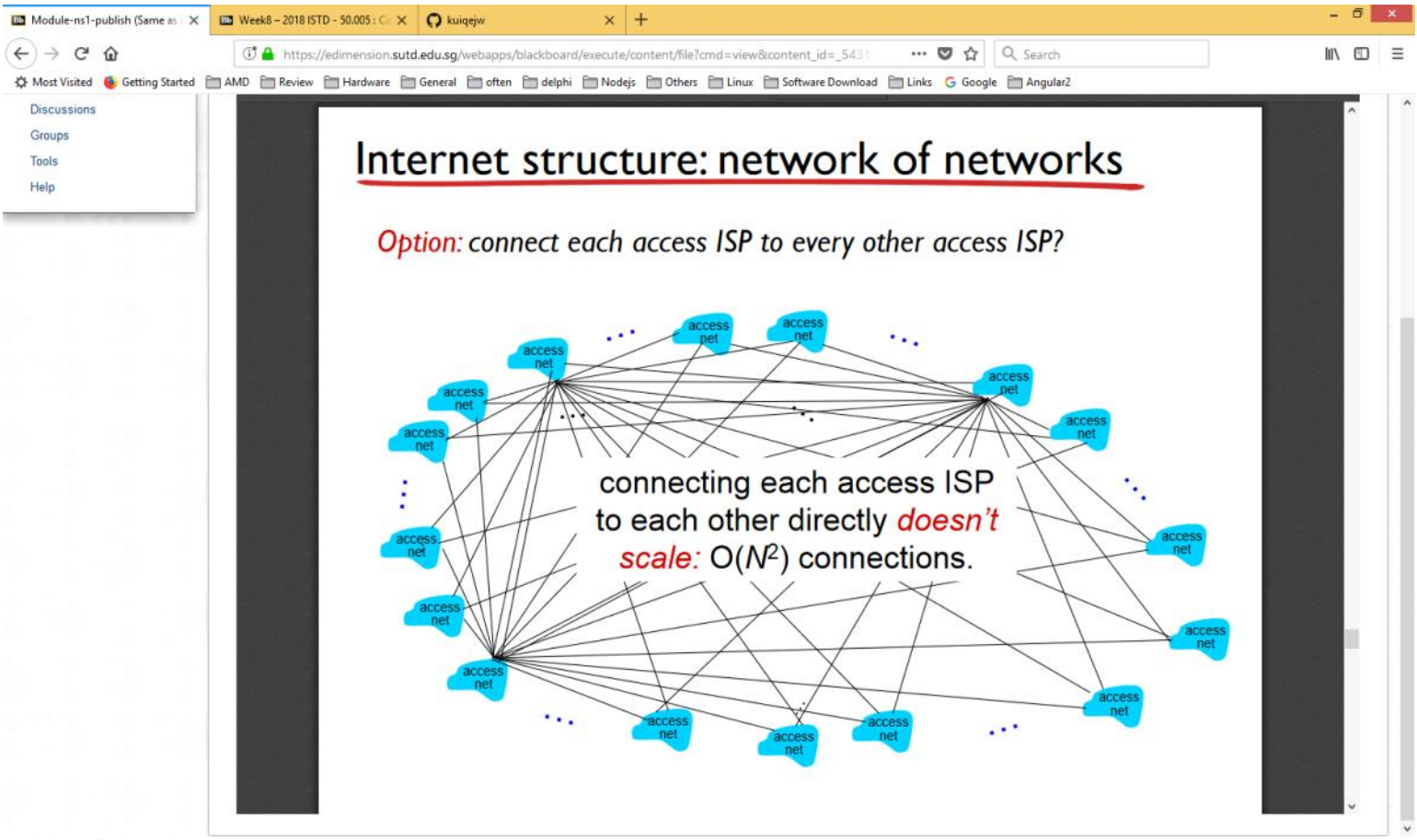
We deal with complexity through layering. Ticketing, baggage claim and on on. Layering in terms of layers of the network. Transport network, datalink. The next challenge is one of scalability. Especially in the 1990s, the internet has grown. Scalability means how the system starts to grow.

Scalability means how to manage system as the system size grows. Bigger and bigger size of internet: sustained fast growth. Increasing structural complexity and thus incommesurate scaling (different aspecs of network grow at different speeds). The internet is such a simple concept that it can evolve bottom up and become as large as what the internet is today. Because it is a network of networks.

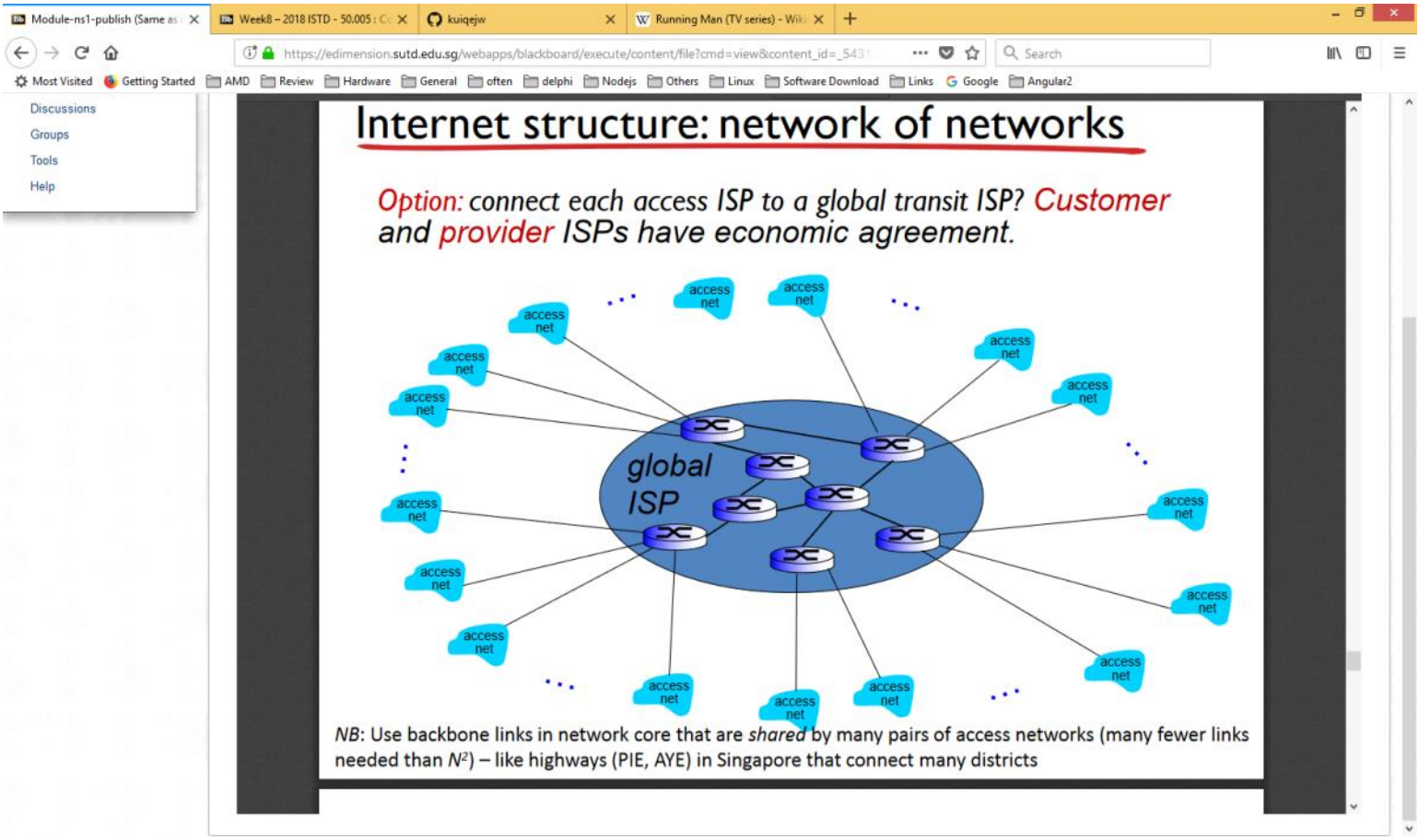
How to build bigger pyramids: commensurate scaling(if something grows order of n^3. The weight of pyramid grows with size of baze. If volume is proportional to the weight.

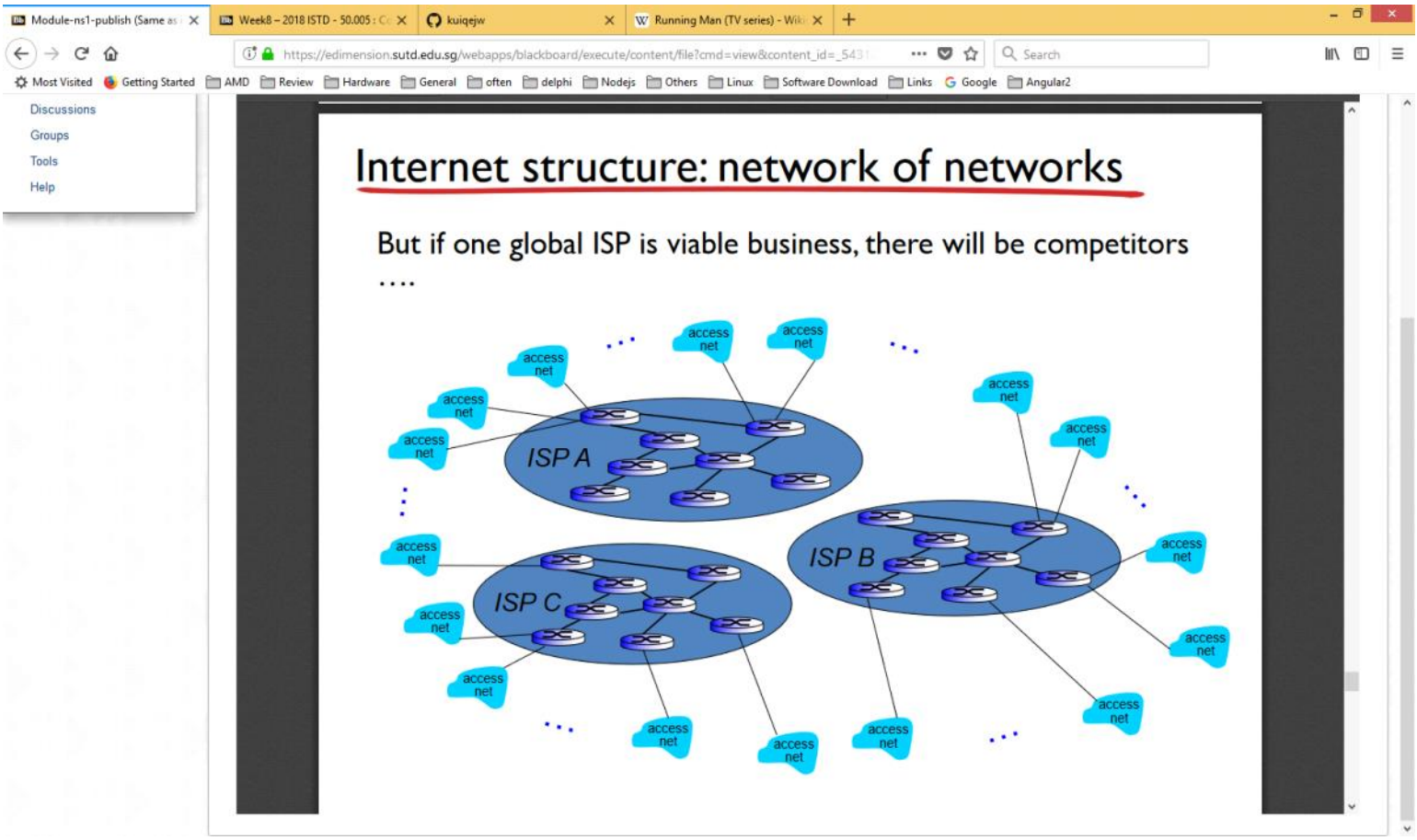
In internet, when the number of nodes N grows, the pairwise interconnect between them grows like n^2. IF you think of n points as access ISP.



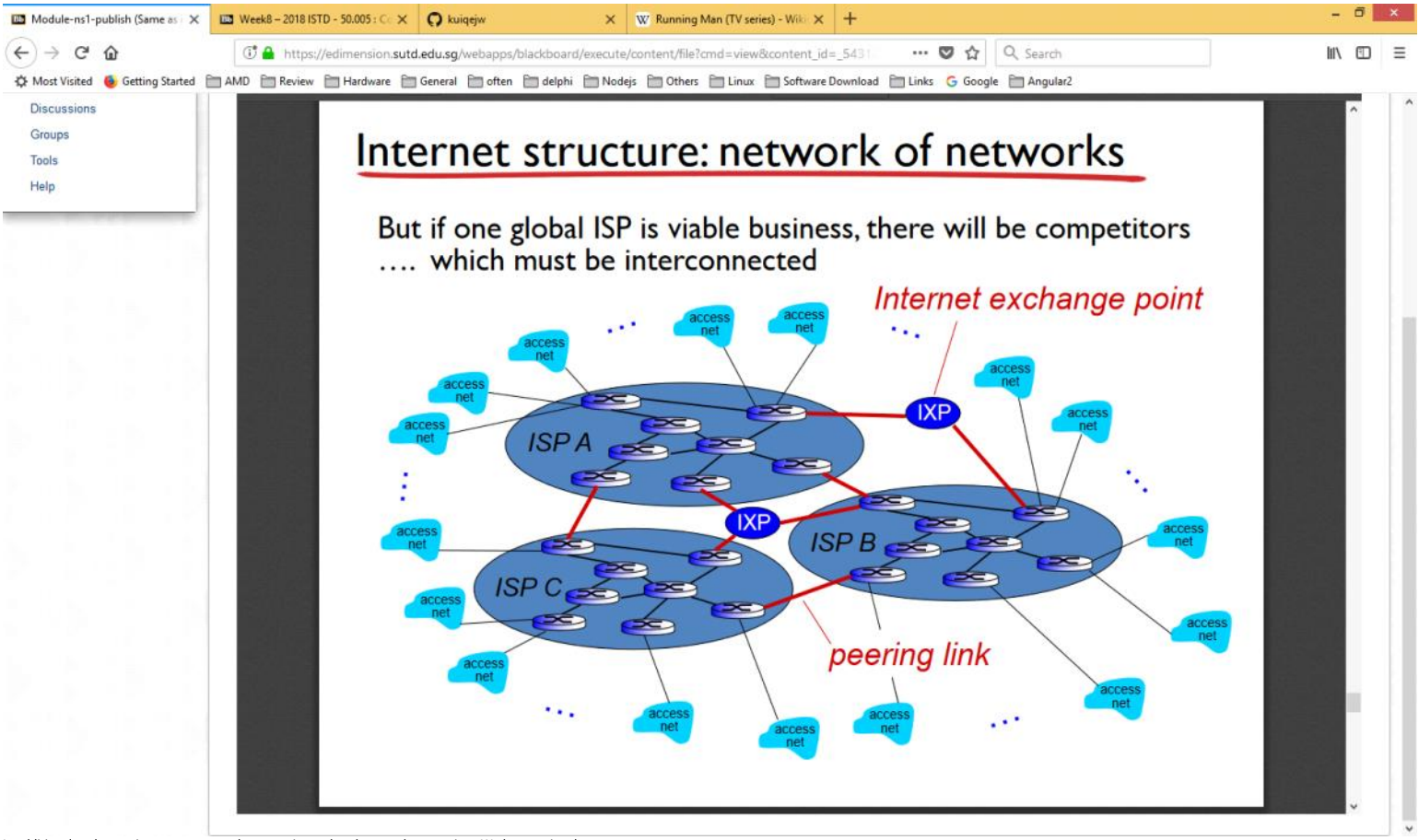


Is this access ISP feasible? It can be SUTD TELCO or COOPERATION or a small company. That is all it takes. To have a connection to each and every one of those, this is not feasible.
There would be n minus one connections and it is just inconceivable. So we introduce the notion of global internet service provider. A global ISP.

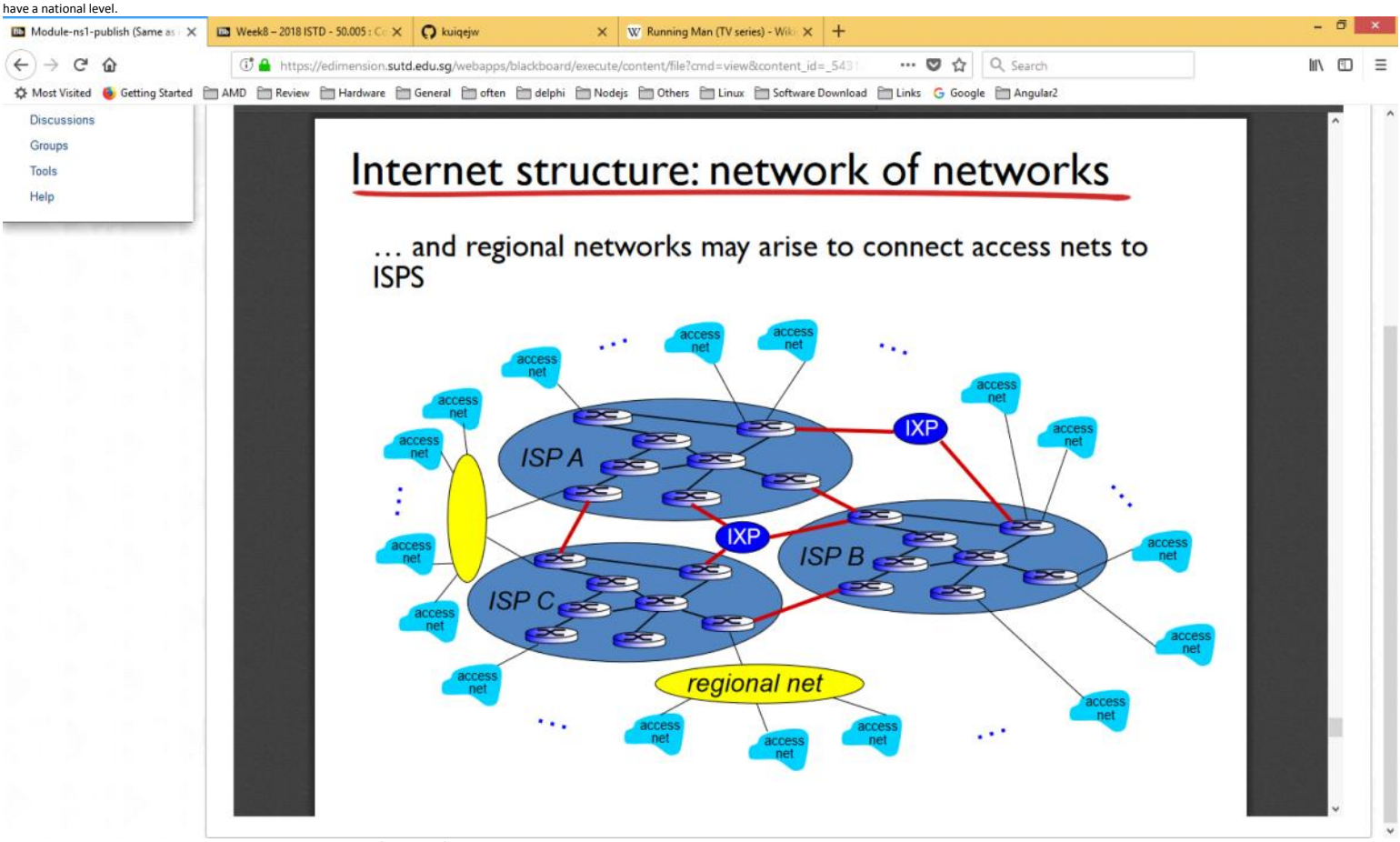




Peering agreements. You have third party entities: New company which is called an IXP: an internet exchange point which is a huge building of routers so that whoever wants to make use of routers can do so in order to carry one or another's traffic. About four hundred ISP in various continents all over. The IXP is purely between ISP (internet service providers)

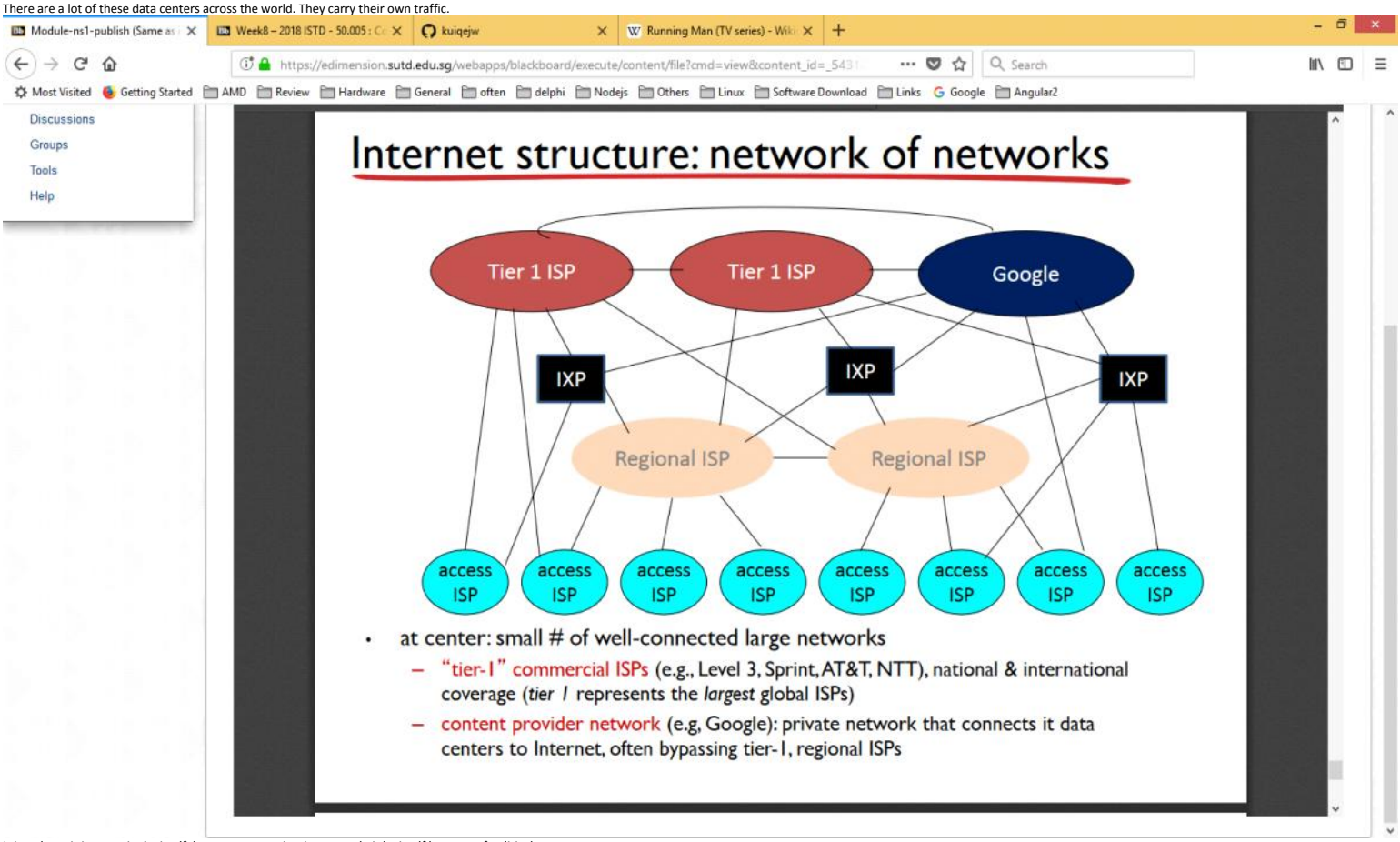


In additional to the previous structure, we have peering and we have exchange points. We have regional networks that are close to the region. At the level of that regional networking, we have a structure. We have a national level.



There are hierarchies within the ISP and within the regional level. As if this weren't enough, there was another level of complexity.

Content providers: provide structures for themselves. Provide content by their own network. About fifty to hundred data centers provided all over the world. Some have only a few hundred or a few thousand. There are a lot of these data centers across the world. They carry their own traffic.



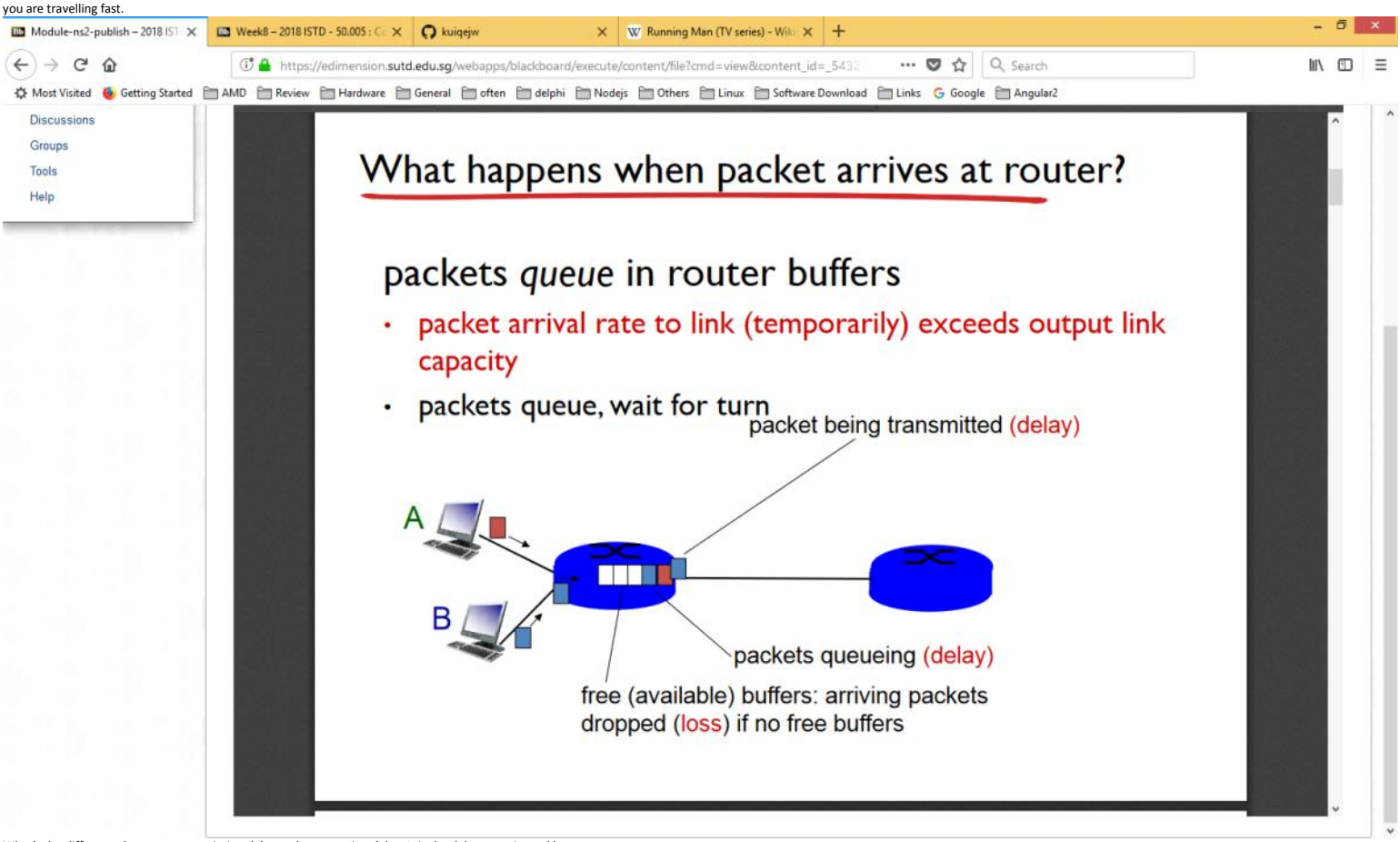
It is so large it is an entity by itself, but some countries, it cannot do it by itself because of political reasons, so they use tier ISPs.

Network Performance

Monday, March 12, 2018 9:41 AM

We talk about network performance: areas researched very heavily. Basic issues are delay throughput and loss. How to reduce delay, from one packet to another, and without loss. We need to make sure that the throughput is sustainable. For other people's packet flows, they are stuck while I get through. We will look at the definitions. We will look at ping and traceroute.
What happens when packet arrives at router? The packets queue in router buffers. I can have a hundred bit or MB transceiver, and it is dependent on access technology. This cannot be ignored. There is this transmission rate, r, which is low. And Arrival rate is very fast. The queue will develop within the switch. Queue is developed within this particular link. I can have several outgoing links for different destinations. There is a processing delay, a transmission delay. Is there another point of delay? There is processing delay, transmission delay. Processing, queuing and transmission. But what is the other delay. Propagation delay.

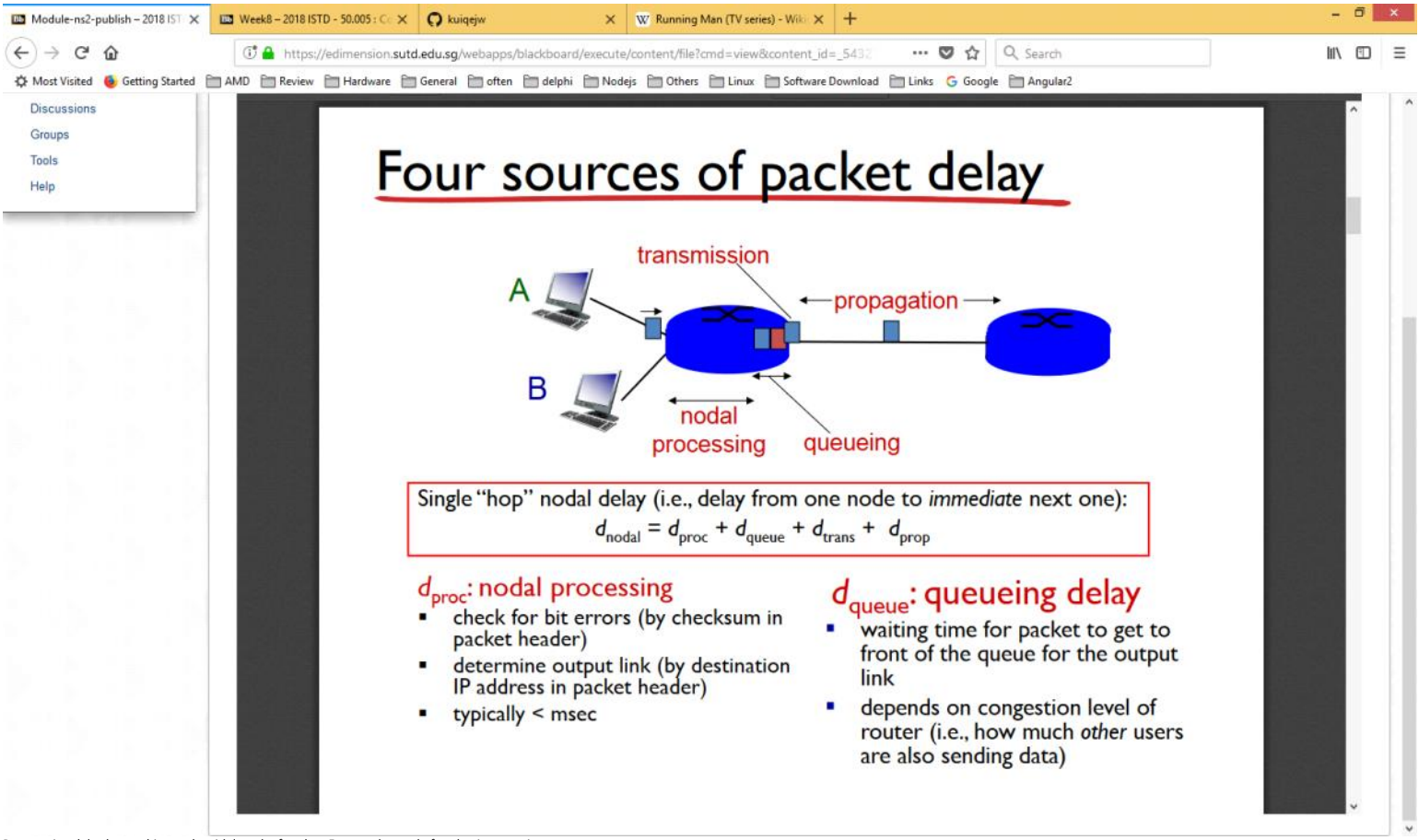
If it is Singapore and Los Angeles, it will take a significant amount of time to get to Los Angeles even if you are travelling fast.



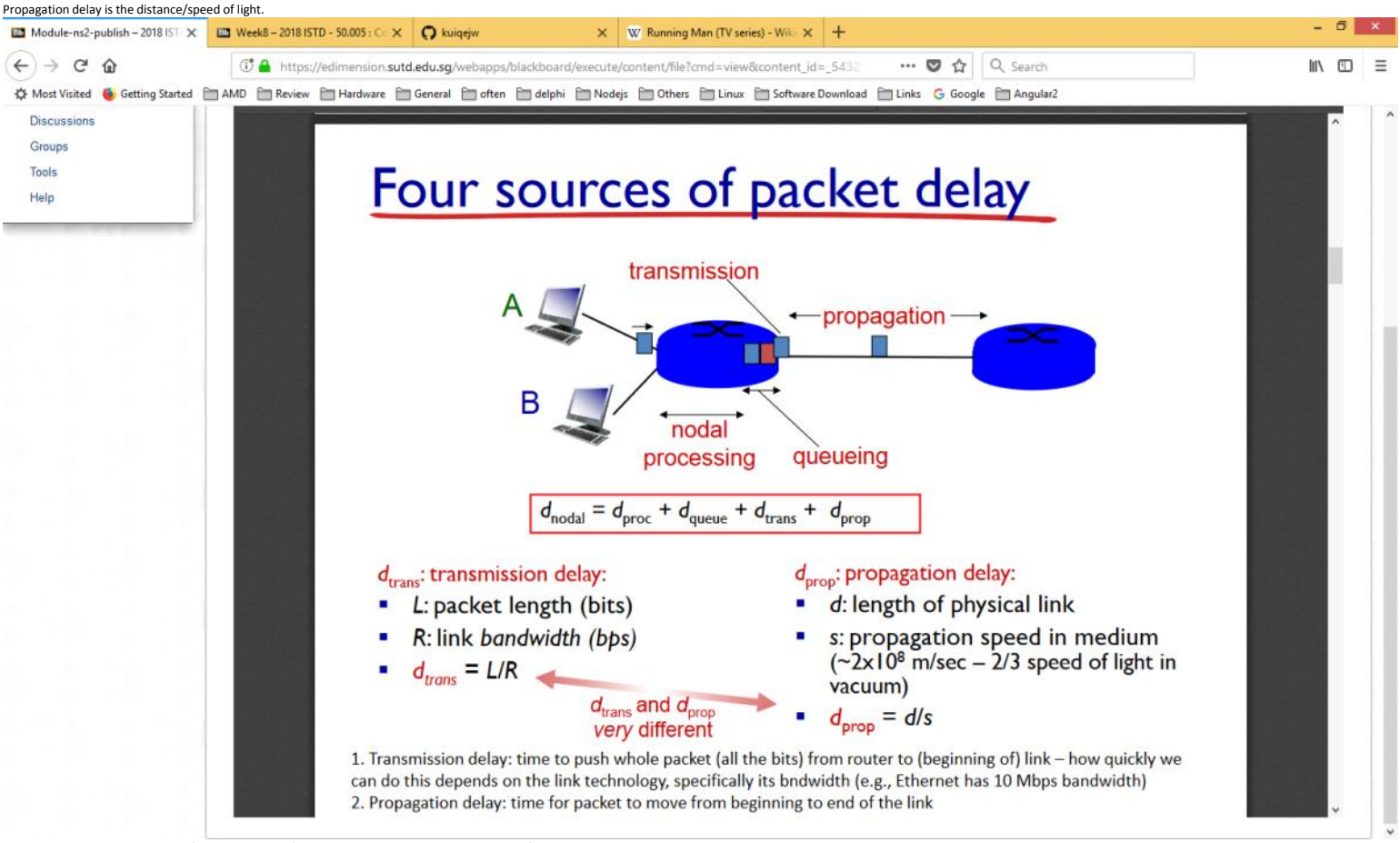
What's the difference between transmission delay and propagation delay: It is the delay experienced by the packet as it is being pushed out through the network. Determined by speed of the transceiver. Say you have a transmission delay (determined by rate of transceiver) bits pushed by router through the network.

Once packet has been pushed out to the network, it will experience delay as it pushes through the network. Then it experiences propagation delay.

What would happen if six packets came but I have only five buffer spaces available? The sixth packet would be dropped.

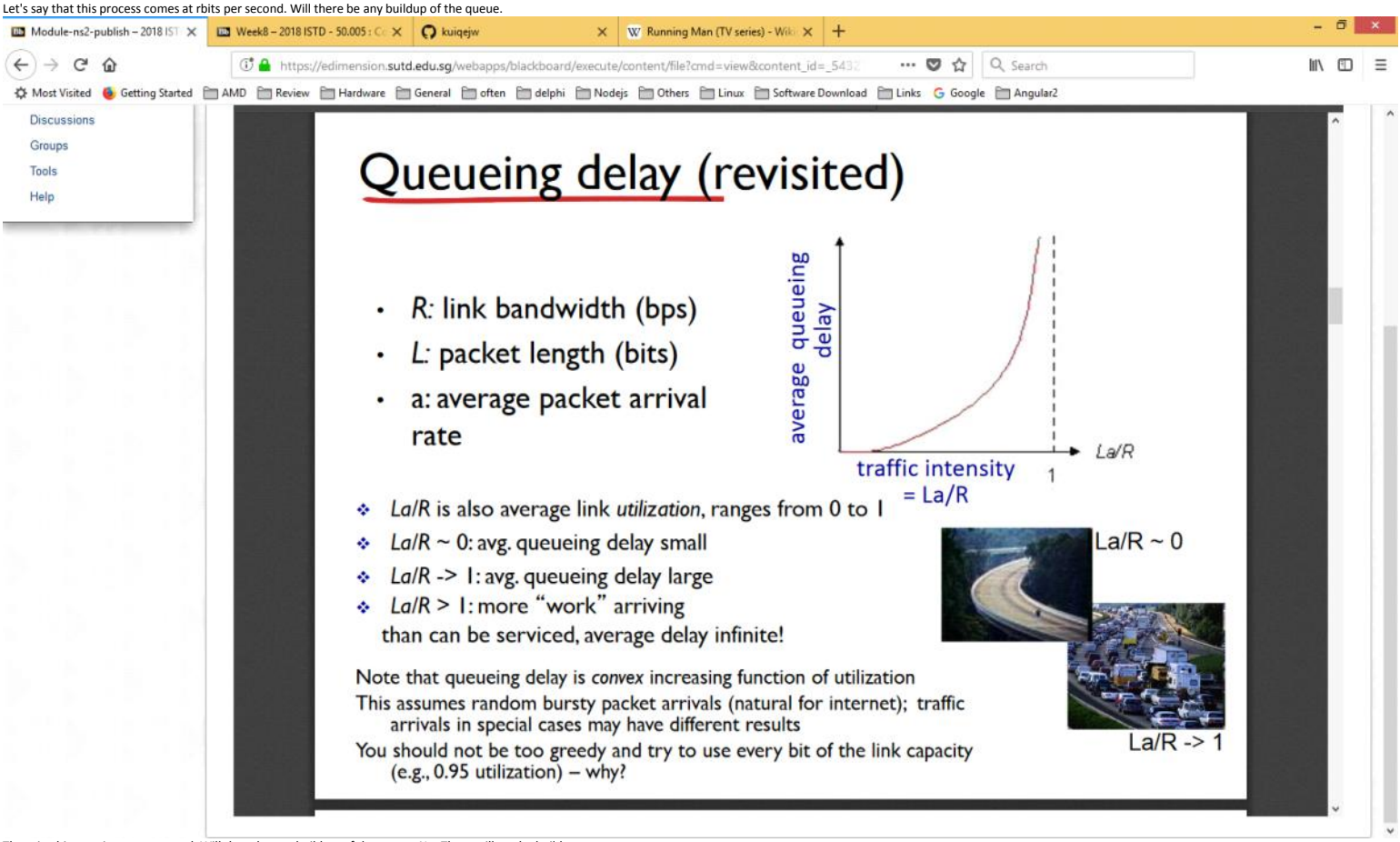


Propagation delay has nothing to do with length of packet. From end to end of packet is not an issue.
Propagation delay is the distance/speed of light.



Another way to look at this issue of transmission is : if you look at our experience on a highway. If you notice, when you are in a highway, there is a ratio of La/R. Average packet arrival rate, and packet arrival in links.

Let's say that this process comes at rbits per second. Will there be any buildup of the queue.



There is r bits coming every second. Will there be any build up of the queue. No. There will not be build up because I know that there are r bits coming, and if traffic is well behaved, every second incoming queue holds r bits. There is no queue.

Exam

Wednesday, March 14, 2018 11:42 AM

One question true and False
One question short answer
6 longer questions
- 1st is Thread and Process
- 2nd is Synchronization (weightage is x2)
- 3rd Deadlock
- 4th File System

No access to calculators or electronic devices
You have to use some major functions; how to use a certain function

For Thread and Process; What's a thread, what is it used for?

System Call:
- Fork and exec
- Summary of main solution approaches: For synchronization
  - Mutual exclusion vs Condition synchronization.
- Semaphore Binary (acquire release) Counting sempahore(acquire/release)

Default anonymous java synchronization	Reentrant binary lock (synchronized method)	Anonymous condition variable (wait/notify or wait notifyAll)
Named/Explicit Java synchronization object	Named reentrant binary lock (binary lock)	Named condition variable (await/signal)

What are the conditions you need to prove. What are the properties you need to prove. Three properties: Autowaiting, Progress, Mutual Exclusion. Really understand (one liner) including the assumption. Criteria assumptions and so on.
Deadlock: Talk about avoidance and prevention.
- How to detect deadlocks: 4 properties.

For File system: It will be the unix design. There were some specific commands that we look at. Not a lot of memorization required.
OpenFileSystem call. You roughly know what to use it. Parameters (File Name and get back a file descriptor)_
Read and Write system calls. That's a handle to file you want to operate on.
Then there's a read and so on.

How does using the lock provide hardware waiting. You have to check whether the thread is wanting to access.

Fork would not return an id

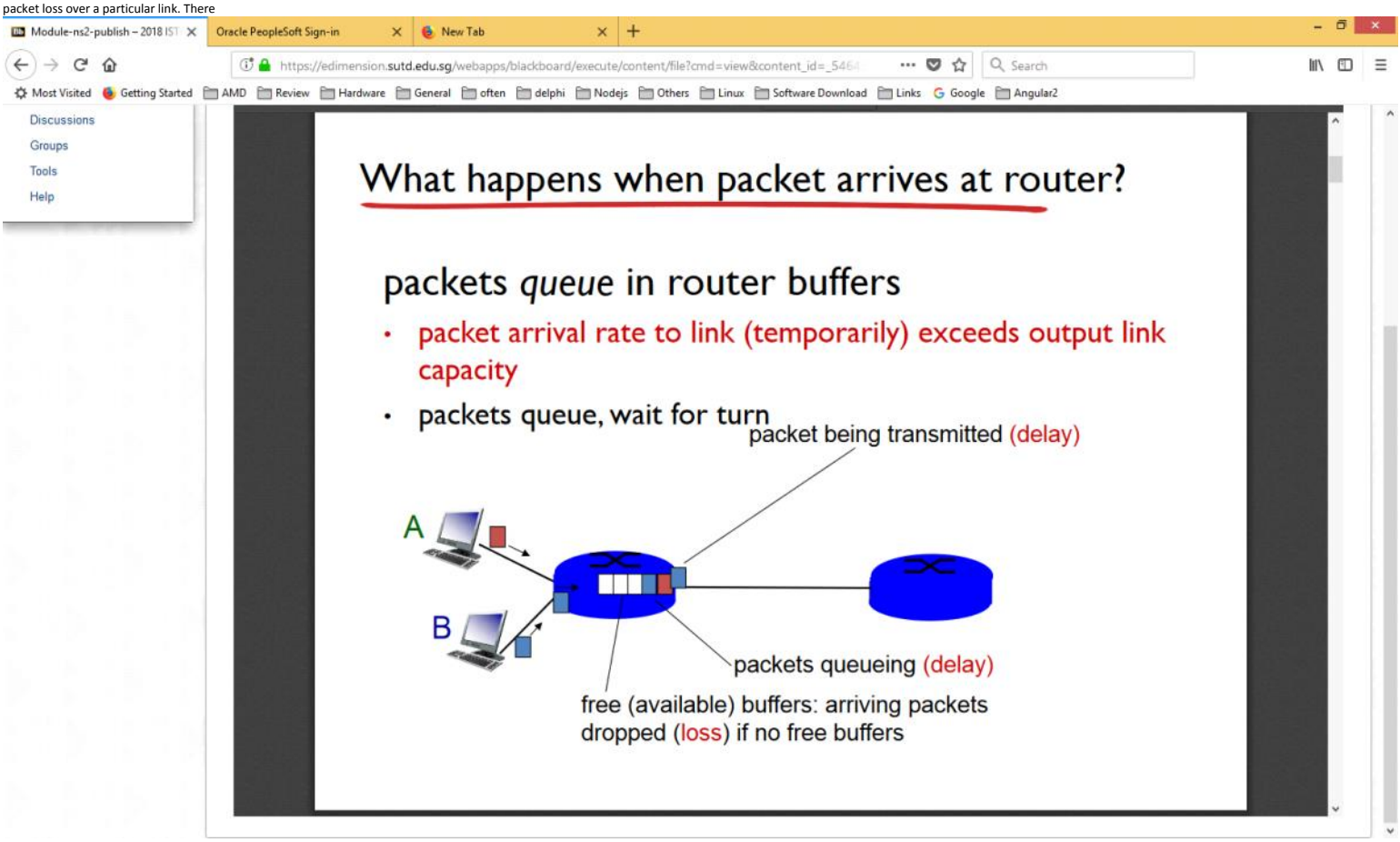
Monday - Performance

Monday, March 19, 2018 8:35 AM

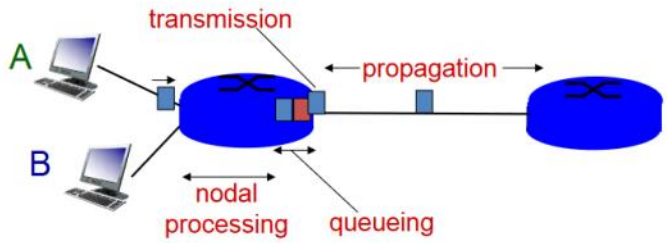
Processing of the incoming packets and then you have the queuing here. We just look at one output link. We have queue in here. And we have transmission and then you have propagation. Looking at one hop there are multiple sources of delay. This is quite deterministic, because it involves the inspection of the packet to check if there are any errors to figure out if there is an output port. This also gives the deterministic transmission. Y is the transmission rate deterministic. Transmission is pushing out the packets onto the link. This is determined by the specs of the transmitter you buy.

Queuing delay gives you the most loss. If this is an internal router, you can guess how much data is going through that router from thousands of sources for the set of this queuing links. This is what leads to all your packet losses and so on.

Last time, you saw the hop delay plus delay in queuing plus delay in transmission and delay in propagation. And the end to end delay is summation of all of this over all of your ops. That's what your end to end delay would be. If you're asked to calculate end to end delay, you do it over all of your links. You just look at model of losses. When this buffer overflows you have a loss. Let's say you have a 10% packet loss over a particular link. There



Four sources of packet delay



Single "hop" nodal delay (i.e., delay from one node to *immediate* next one):

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: nodal processing

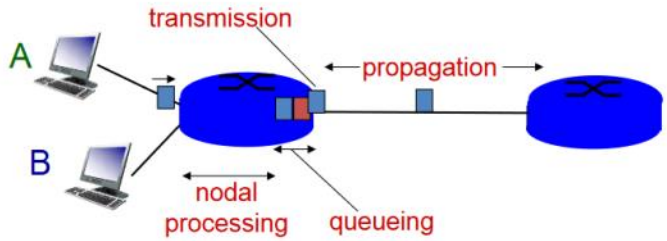
- check for bit errors (by checksum in packet header)
- determine output link (by destination IP address in packet header)
- typically < msec

d_{queue}: queueing delay

- waiting time for packet to get to front of the queue for the output link
- depends on congestion level of router (i.e., how much *other* users are also sending data)

Module-ns2-publish – 2018 IST | Oracle PeopleSoft Sign-in | New Tab
https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5464
Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2
Discussions
Groups
Tools
Help

Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay:

- L: packet length (bits)
- R: link *bandwidth (bps)*
- $d_{trans} = L/R$

d_{trans} and d_{prop} *very* different

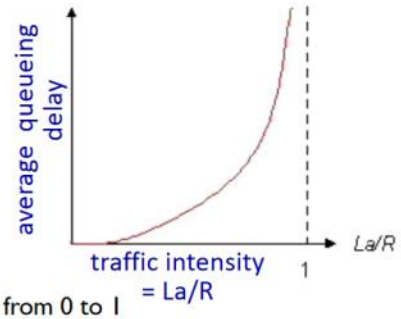
d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed in medium (~2×10^8 m/sec – 2/3 speed of light in vacuum)
- $d_{prop} = d/s$

1. Transmission delay: time to push whole packet (all the bits) from router to (beginning of) link – how quickly we can do this depends on the link technology, specifically its bndwidth (e.g., Ethernet has 10 Mbps bandwidth)
2. Propagation delay: time for packet to move from beginning to end of the link

Queueing delay (revisited)

- *R*: link bandwidth (bps)
- *L*: packet length (bits)
- a: average packet arrival rate



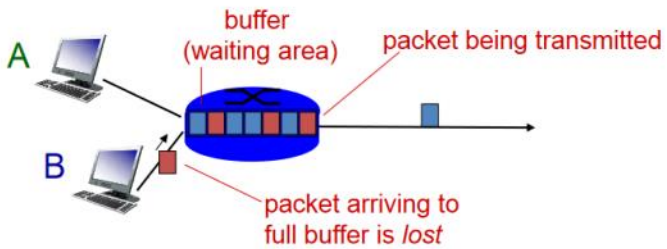
- *La/R* is also average link *utilization*, ranges from 0 to 1
- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!

Note that queueing delay is *convex* increasing function of utilization
This assumes random bursty packet arrivals (natural for internet); traffic arrivals in special cases may have different results
You should not be too greedy and try to use every bit of the link capacity (e.g., 0.95 utilization) – why?

Module-ns2-publish – 2018 IST | Oracle PeopleSoft Sign-in | New Tab
https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5464
Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2
Discussions
Groups
Tools
Help

Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all

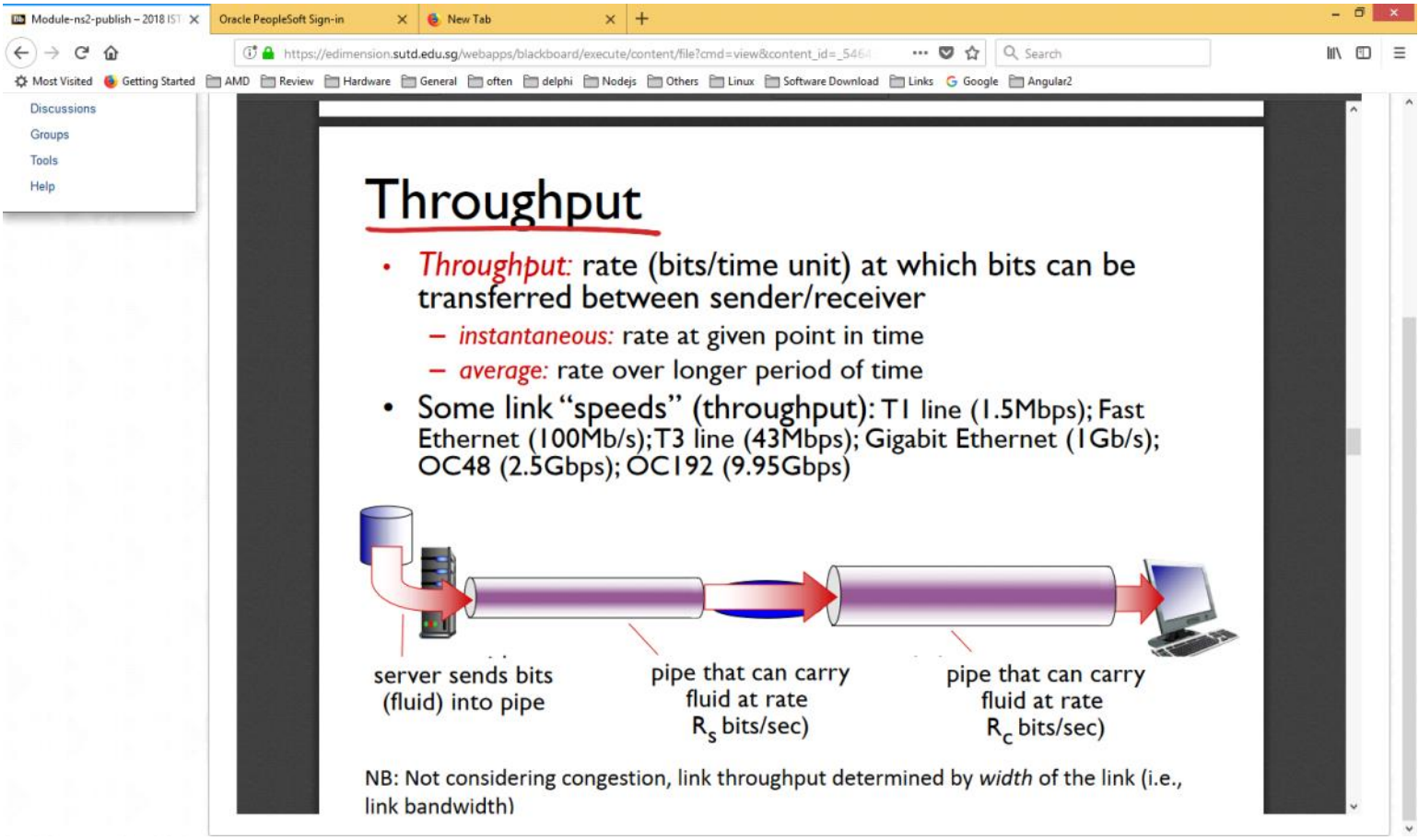


- Loss is *random* – each link has average *loss rate/probability*
- How do per-link loss rates compose into end-to-end loss rate?
- Impact of loss on applications: Email? Banking? Images? Video?

Our simple model of packet loss is that when the packet arrives and the buffer is full, that leads to packet loss. Loss is random, so we model it probabilistically. As I was saying, how do per link loss rates compose to end-to-end loss rates?
The question is, what would the end to end loss be?

What is the impact of loss on applications. If I lose packets, is it a serious thing? Or banking. I'm in the middle of a transmission and I lose a few packets. How about if I have a video and I happen to lose a few packets is that serious? Video can sustain losses: the term is temporal redundancy. Temporal and spatial redundancy. This is exploited a lot in mpeg standards. What it basically says if a bit is missing in frame A, it is most likely the same bit in A + 1 or frame A-1. You can extrapolate from this bit or the corresponding bit over there. It is temporal redundancy. You can recover the bit if it is clear blue sky. You can extrapolate from here to here. You can exploit that feature and recover information.

Loss on applications: certain applications like electronic mail or banking applications, anything that is security enabled can't sustain losses. However video and audio as well can sustain losses.



You have a server and a client. The server is pushing bits into the router, and router is pushing bits into the client. This link is Rs and the link capacity to the client is Rc. That is determined by the transmission rate of these outgoing links. These are actually average rates over long periods of time. In such a scenario, what would be the limiting throughput. In this link, you have a maximum throughput of Rs per seconds.

Whichever is the minimum is the link throughput. The client is consuming everything that the consumer is pushed, so Rs would be lower. If the server can blast on the other hand, the client cannot consume and you have the packets blowing up over at the Rs, and that's not so good. Some of the typical bits would be one and half bits per second, T1.
That is called the bottleneck link.

Module-ns2-publish – 2018 IST | Oracle PeopleSoft Sign-in | New Tab

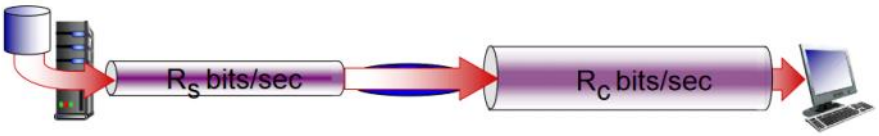
https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5464

Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2

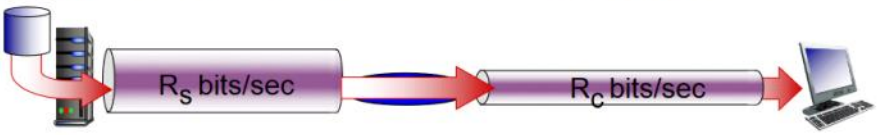
Discussions
Groups
Tools
Help

End-to-end throughput (how do per-hop throughputs compose)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

Module-ns2-publish – 2018 IST | Oracle PeopleSoft Sign-in | New Tab | How to Calculate Packet Loss

https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5464

Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2

Discussions
Groups
Tools
Help

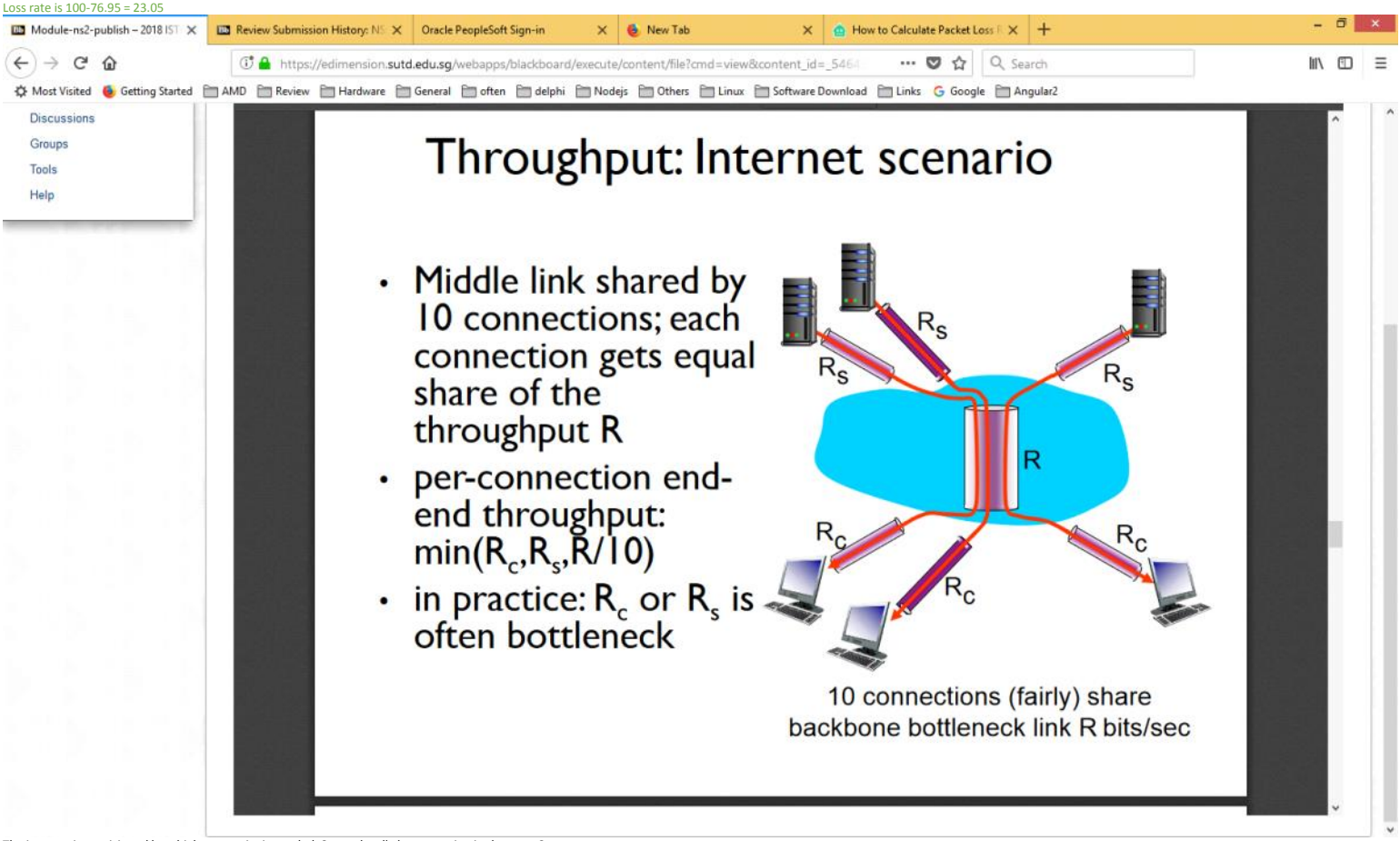
Activity 2.1

- An end-to-end network path *P* consists of three links 1, 2, 3.
- Link 1: delay 2ms, throughput 100Mb/s, loss rate: 5%; Link 2: 60ms, 1Gb/s, 10%; Link 3: 5ms, 10Mb/s, 10%
- What are *P*'s delay, throughput, and loss rate?
  - Assume that losses are independent

NB: Strictly speaking, 1k=1024, 1M=1024^2, etc. **But** in your calculations, use **1k=1000, 1M=10^6**, etc, whenever doing so simplifies your life.

Delay of 2 ms + 60 ms + 5 ms = 67ms
Throughput = 10 Mb/s
Loss rate = .05*100 + 1000*0.1 + 10*0.1 = 106
The throughput is characterised by the limiting throughput and the minimum link.
There are five percent loss getting lost here, 10 percent here and 10 percent here. Is it additive?

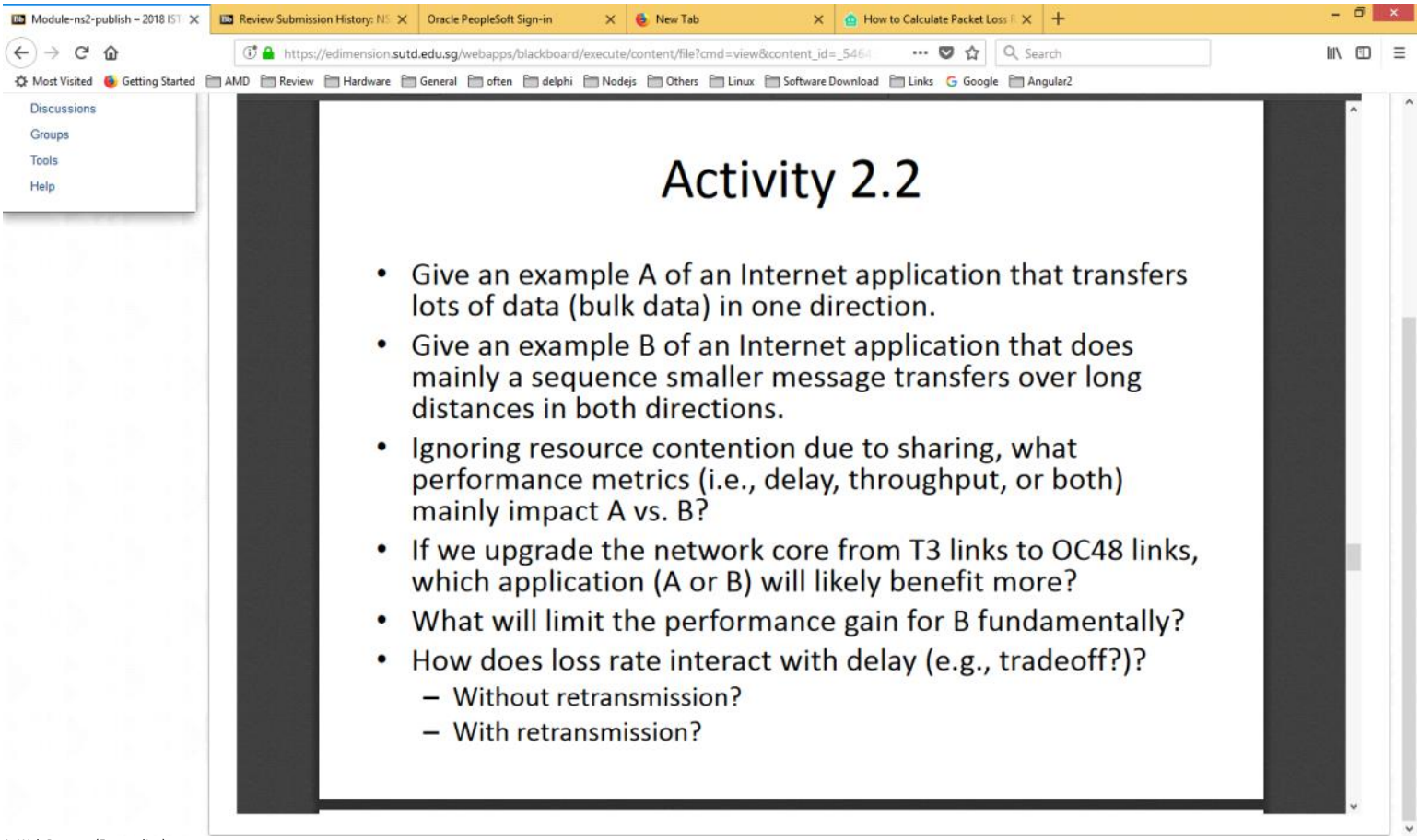
If five percent of the packet is lost here, then 95% get here.
10 % are also lost here. So 0.9*95 = 85.5
85.5*.9 = 76.95
Loss rate is 100-76.95 = 23.05



The internet is provisioned by a higher capacity is needed. So you hardly have queuing in the core. So we are not trying to model the big internet but looking at the microcosm of the internet and find out what is the limiting bandwidth in this case. You can look at the packet flows that are coming in. The throughput of each packet flow is more or less the same. In other words, the transmission rate of each of these incoming links is the same. These are the Rs and for the clients the transmission rate is the same. And we have 10 servers and 10 clients. If R is much larger than Rs and Rc, it comes to the case of minimum of Rs and Rc. If R is much larger, then we go back to our original order.

For instance, Rs is 2 Mb/s and Rc is 5 Mb/s, then we choose Rs.

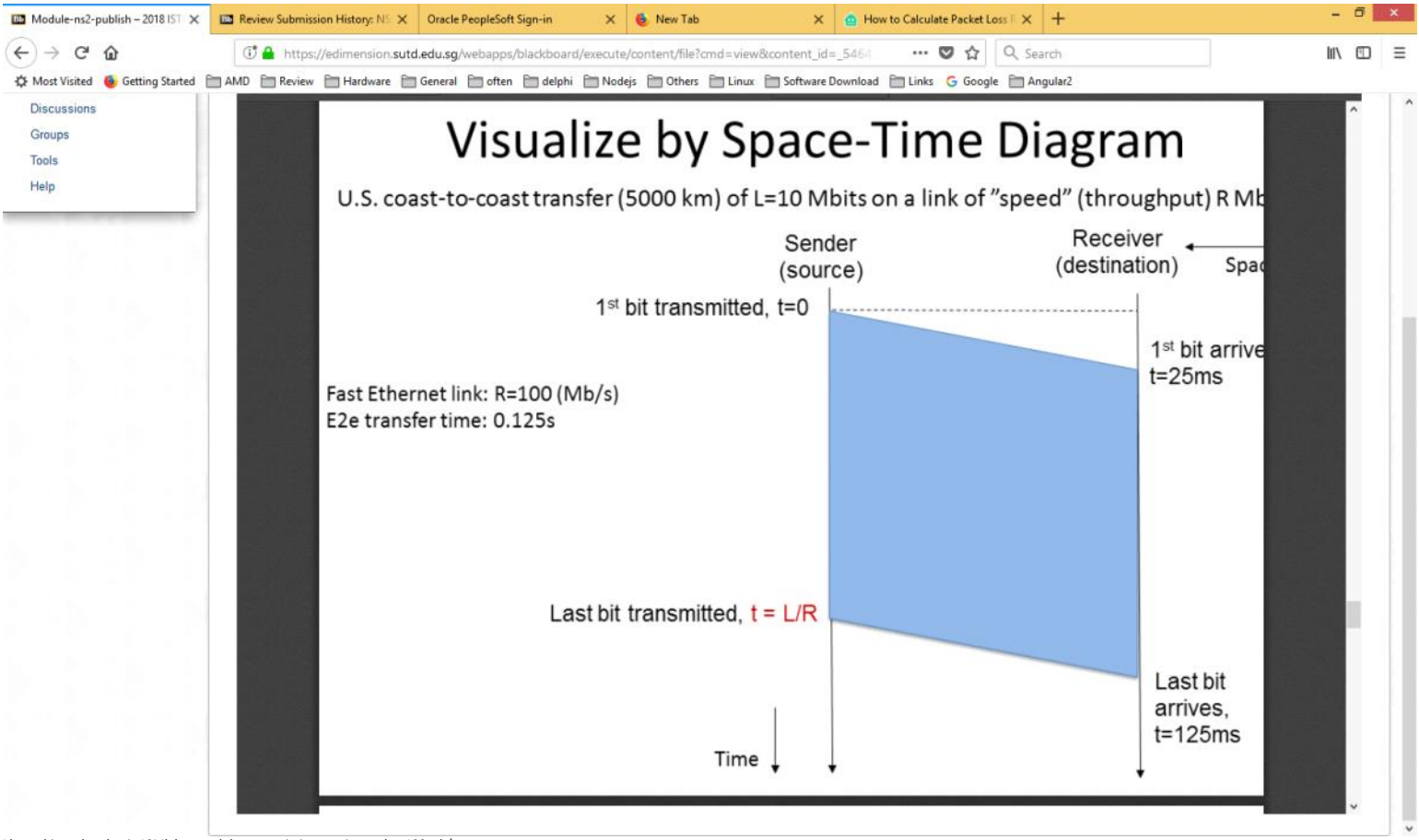
Each of those packet flows is going to the same bottle neck link. Because it is shared, the bandwidth is now divided. And again, to make a very simple model is to divide the bandwidth. There could be a scenario with video servers and client servers on the other end. Continuous packet flows is the main thing.



A: Web Browser (For reading)
B: Forum
A has video streaming and also file transfer. Download of large files is bulk data in one direction. Mpeg. MP3 music download. All of these are bulk data in one direction
B: sequence of smaller message transfers over long distance
Voice over IP or Skype. Forums wouldn't give packet flows. You will get multicast traffic or bursty traffic. You get jitter, and you can't sustain too much of a loss in voice. Voice over IP is packetised voice.
A has longer delay
What impacts A over B? What components impact A and B?
Voice over IP (delay impacts voice most.). The delay impacts Mp3 downloads. As long as your file comes, you don't want your files to be missing. Loss is what affects your downloads. Throughput is important. You would need througput in voice, but not that important. Voice is not very throughput intensive, but video has a lot of information. Delay is important for voice. In skype, you also have data. You are discussing a spreadsheet. You need to have no losses. So it is a combination of these things.

How does loss rate interact with delay? With transmission and without retransmission? If you lose a packet, you retransmit? How does TCP gives you reliable transmission of your data? It is reliability, UDP has losses, TCP doesn't.

You have a space time diagram where time proceeds downwards and packet goes from left to right.



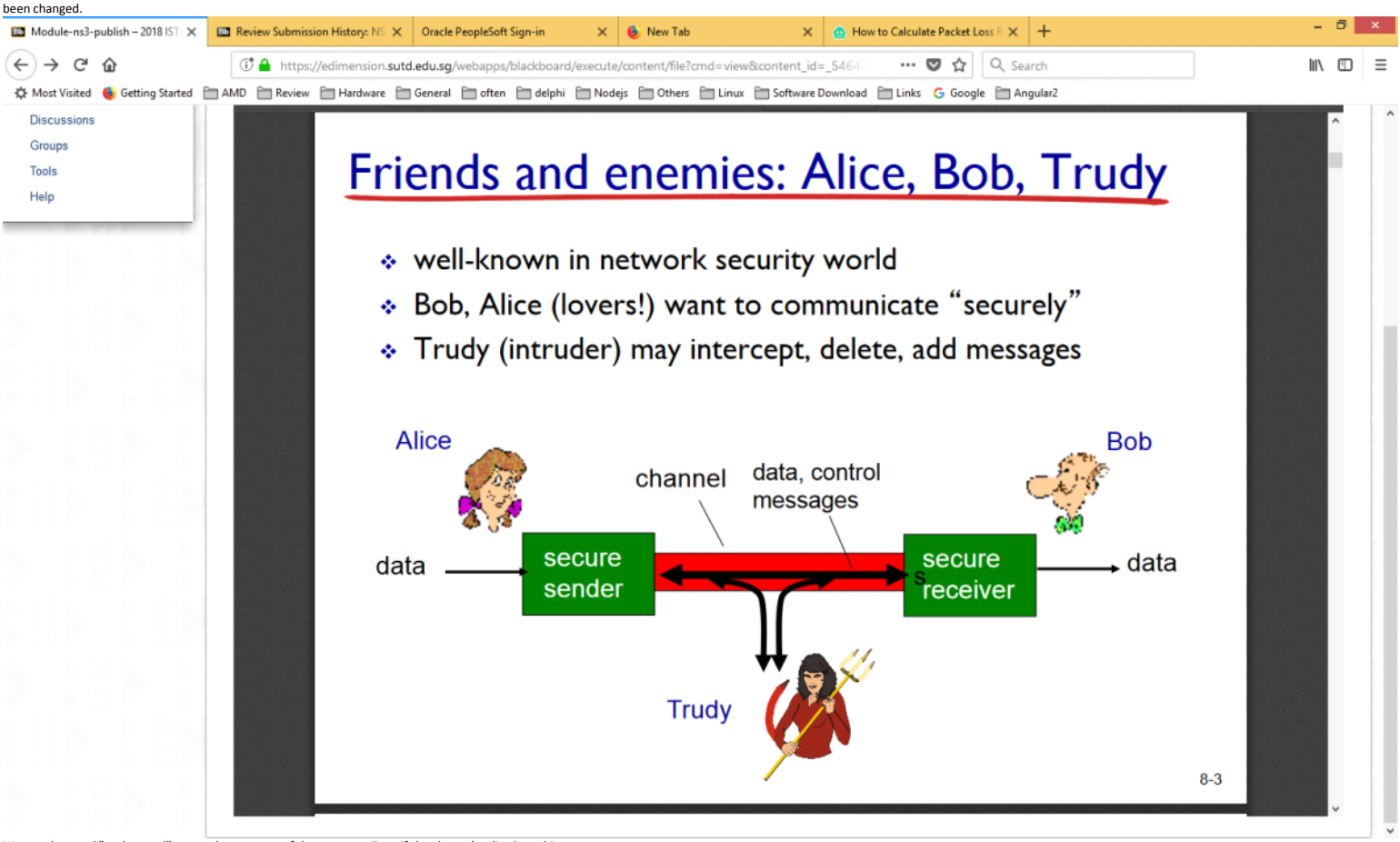
I have this packet that is 10Mb long and the transmission rate is equal to 100 mb/s
The delay for transmission is 0.1 second.
So you get delay plus file transmission = 125ms
How can I reduce the delay? One is the transmission time. The other one is the time taken for propagation? How would I improve my transmission delay? Faster connection, faster transmission.
Network transmission rate r.
So if R is smaller. So I get 1000Mb/s instead of 100. But no matter how hard I try, the limiting factor is the distance. The Propagation delay cannot be improved upon because that is limited by the speed of light. Or the time of propagation.

Monday - Security

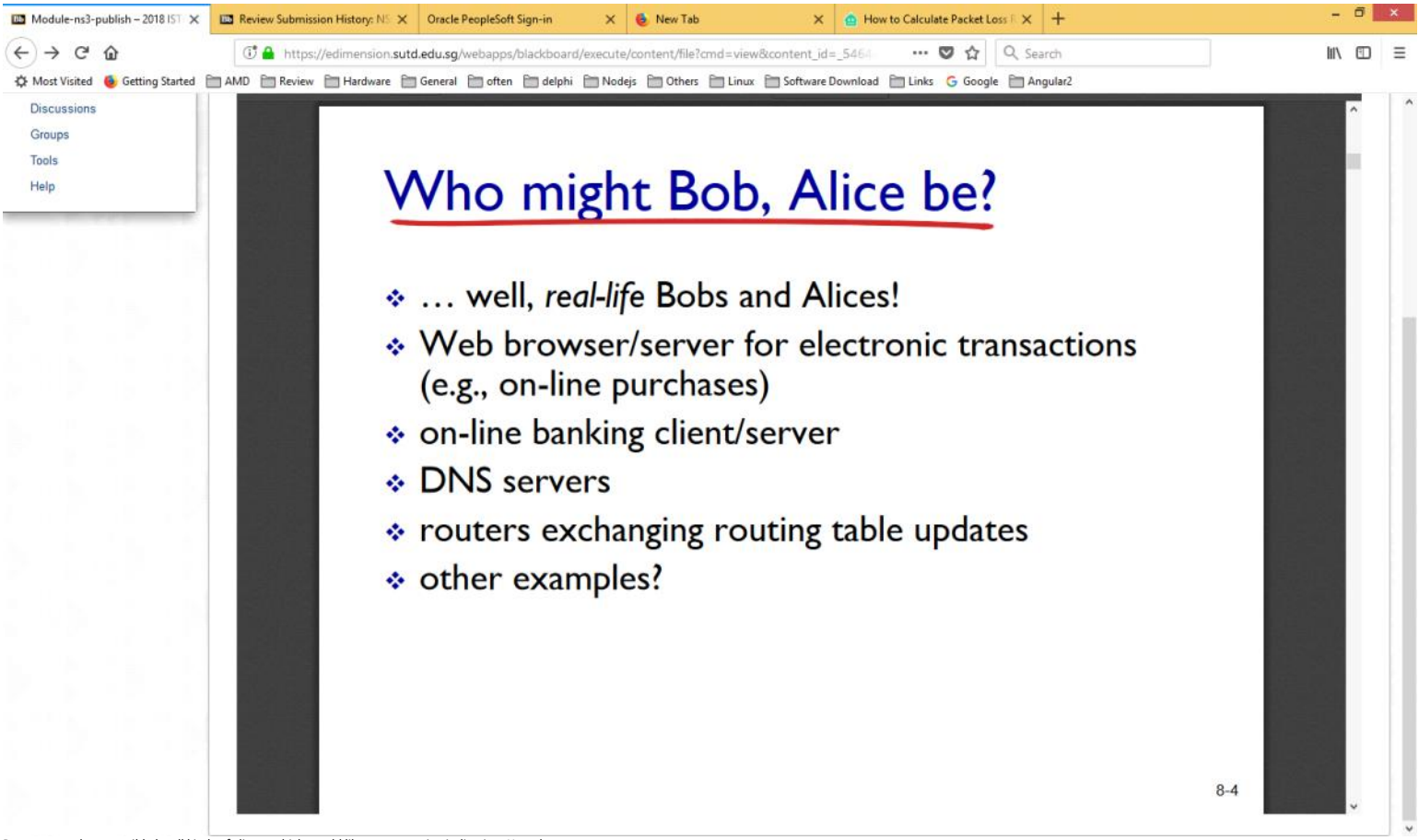
Monday, March 19, 2018 9:31 AM

By putting a receiver next to transmitter, I can capture all the packets? By the very fact that I am putting my packets out there, I am vulnerable. A person who doesn't get a good motive can analyse and get a lot of information about the packets that are going through. This can happen on wire networks as well. All you need is a link to a network. Ethernet is a shared media traditionally. It is a bus model. Mostly, the internet is implemented in a switch architecture. Traditionally, internet was a bus architecture. It is easy to tap into a bus.

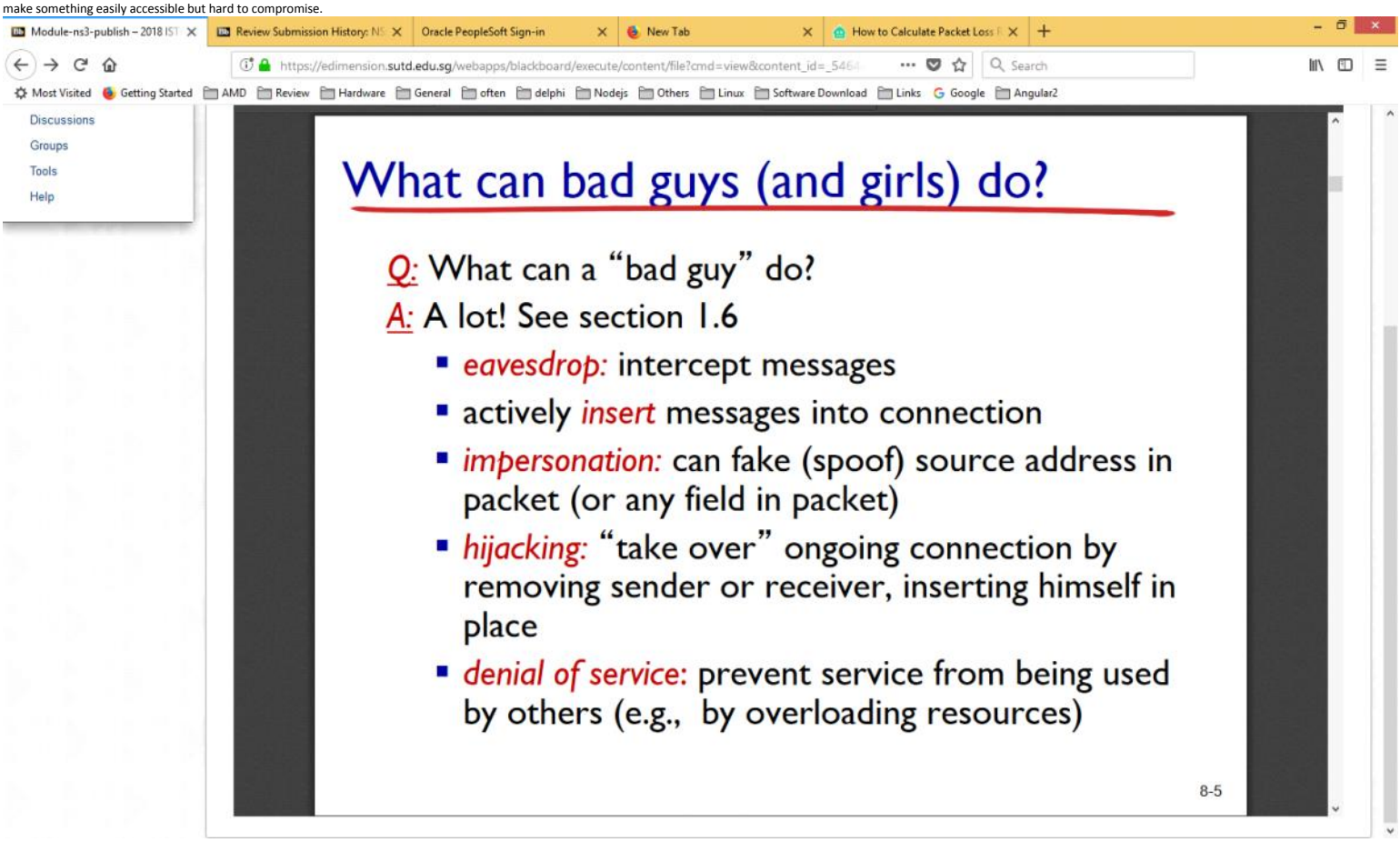
We need to ensure authentication: any entity at any end of the communication link is who they claim to be.
Message integrity: my message that has gone over this channel has not been tampered with. Has not been changed.



We use the word "understand" to see the contents of the message. Even if they haven't edited anything, this is already a compromise.



Routers must be accessible by all kinds of clients which would like to get routing indication. How do you make something easily accessible but hard to compromise.



Wednesday

Wednesday, March 21, 2018 11:32 AM

Confidentiality is to ensure that when the sender sends something, the intended receiver should understand the message contents. Authentication: sender, receiver wants to confirm identity of each other.
Message integrity: sender, receiver wants to ensure message not altered without detection
Access and availability.

Our attacks are going through the channel, not through the end points. You use an encryption key, intruder captures the cipher text and try to catch it. A decryption key is applied to get the plain text
KsubA KA(m) cipher encrypted with key Ka
m = Kb(Ka(m)) where m is the plain text message
The substitution cipher: A In a monoalphabetic cipher, the idea is to take a random permutation of all alphabets of the cipher. So in this case, a maps to m b maps to n and so on. So 26! = 4E26 Space. There are many guesses that one might have. If you know who send the message, you know what the likely letters are, so you can reduce the set space. If we know it is English, and we find that in cipher text, c appears most frequently, that's e. That's a statistical approach. One can break this cipher easily.

If you have a polyalphabetic cipher. Where we have two sequences. We have two Caesar's ciphers. So a maps to t. There are many ways we can try to improve upon the cipher schemes. Our symmetric key ciphers are various keys on that. You take a block of your plain text and equate it to a block of the same size. For a 3 bit encoding, what would the size of the key be. That's two to the three. Size of key for k bit would be what? How many tables would I have to search to track this key. So I would have a set space of 8!.

What is done in practice is to break up the 64 bits to 8 bit patterns. With 8 bits it is not difficult to store. 2^8 sized table is not difficult to store. 256 is not difficult to score. You need something that appears to be random. If you didn't have the scrambling stage. This would be an 8 bit pattern. It is a deterministic scrambling so the receiving and sending side would have the same algorithm. You have given it an appearance of randomness.

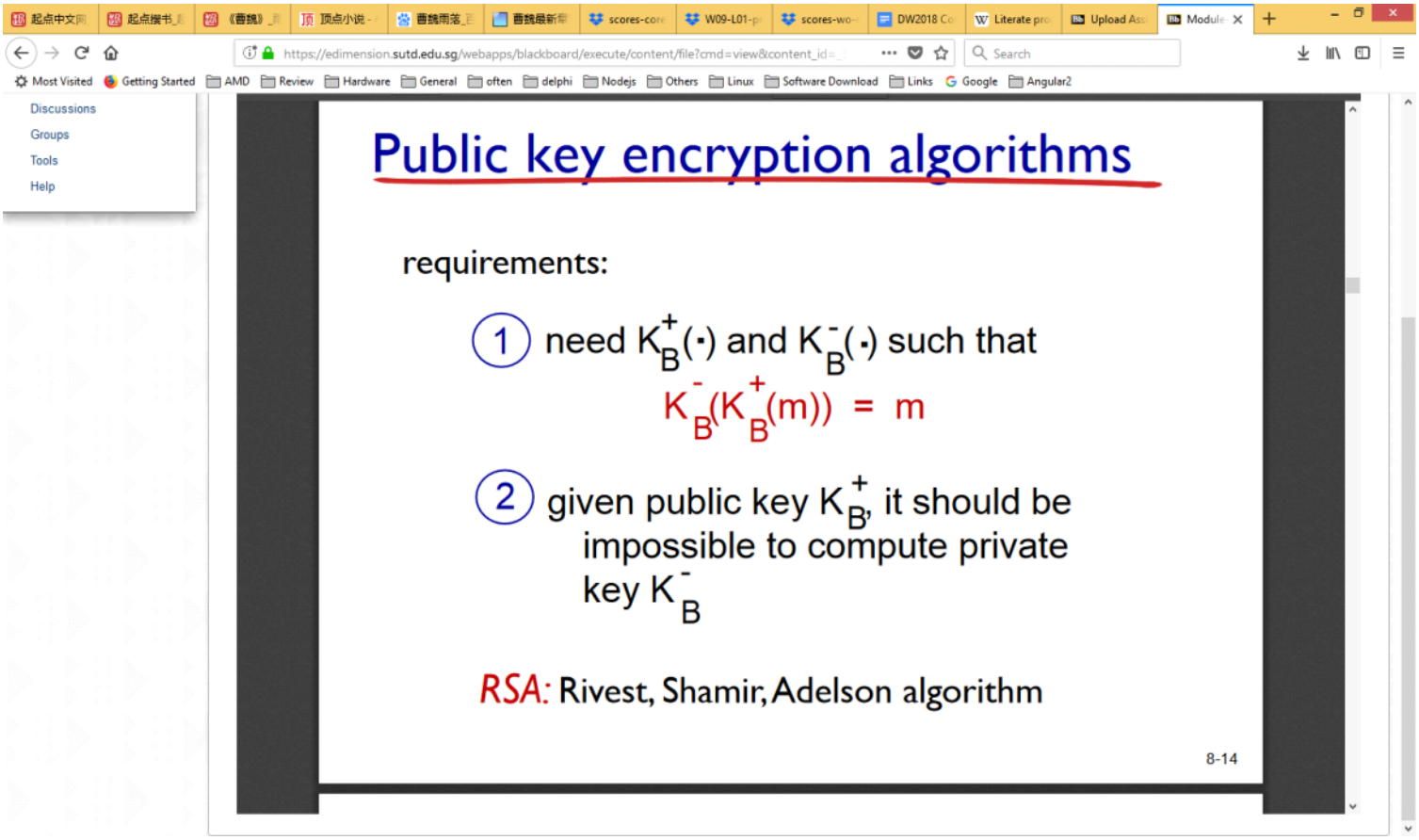
The looping has been unfolded. DES is the unfolding of that loop around 16 times. It yields you the 64 bit output which will become a cipher text. At each stage, you do some scrambling and (details not important). You have a permutation, 16 identical rounds of function application, distribute the bits from the source to the cipher text.

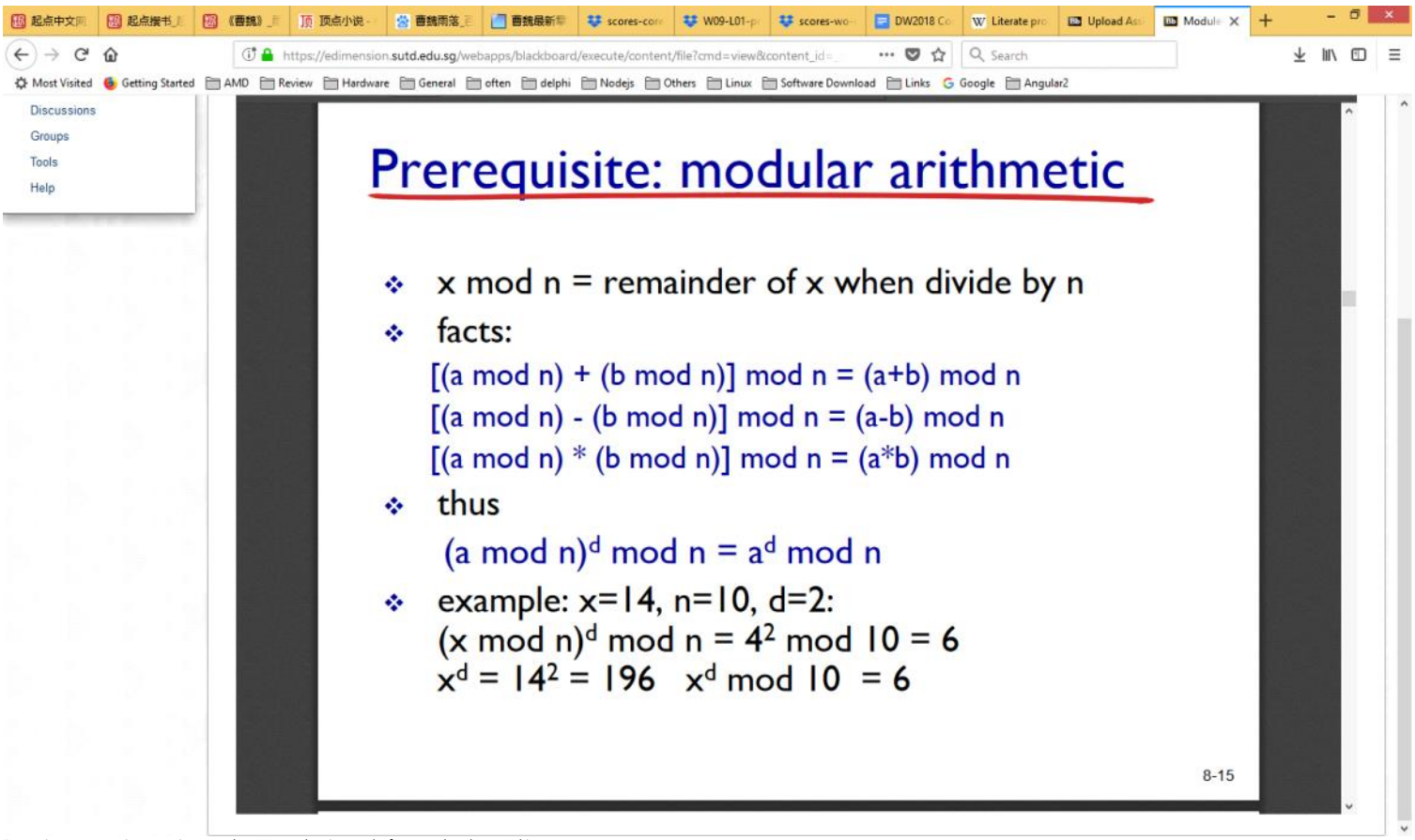
This was the standard that worked until 1993. When it was cracked in a few days by an array of computers. They came up with an improved version. 64 of that order, it is quite easy to crack within a few hours. The solution is to increase the size of the key. 56 + another 8 bits. It takes less than a day to decrypt (brute force) The only way to make DES more secure: 3DES: encrypt 3 times with 3 different keys.

AES: Symmetric key, processes data in 128 bit blocks.
Symmetric key is based on the approach that the sender and the receiver have the same key. Any substitution cipher. This key has to be shared. It requires the sender and the receiver to know in advance the shared secret key. In networking, it is rare to have the sender and receiver "meet", so how to agree on the key in the first place?

We come to a radically different approach: 1976: a public key. This idea of having a public key, where part of the key does not have to be shared to both sides. Only a portion of the key is known by both entities, and the whole world including the attacker. Nobody but the recipient knows the private key. The public key is known to everybody while the private key is known by the recipient. This is known as the asymmetric key crypto.

RSA makes use of the property proposed by Diffie-Hellman. Alice knows Bob's public key (Kb+) Bob's public key is used to encrypt the plain text. We get the cipher text, Kb+m. On the other side, Bob applies the decryption algorithm, using the decryption key of Kb-. And Kb- applied to Kb + applied to m recovers m. This is the basic idea of the public key.



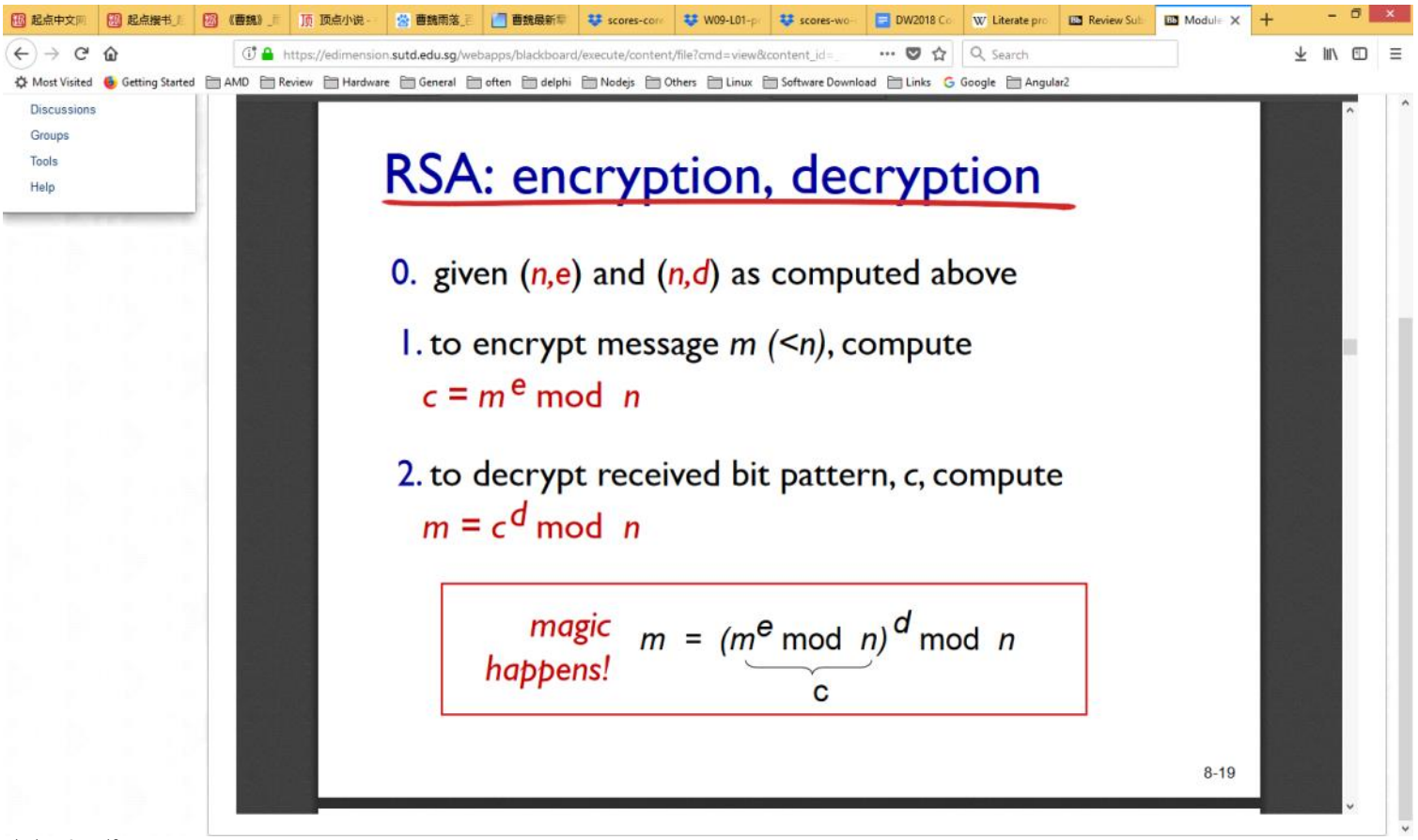


Encrypting a message is encrypting a number. We are drawing results from number theory. A bit pattern is an integer.
Some mathematical operations on an integer. For example, let's take this unique bit pattern. 1001001. when we encrypt this message m, we encrypt the corresponding number which gives ups the cipher text. So this brings us to the RSA algorithm.

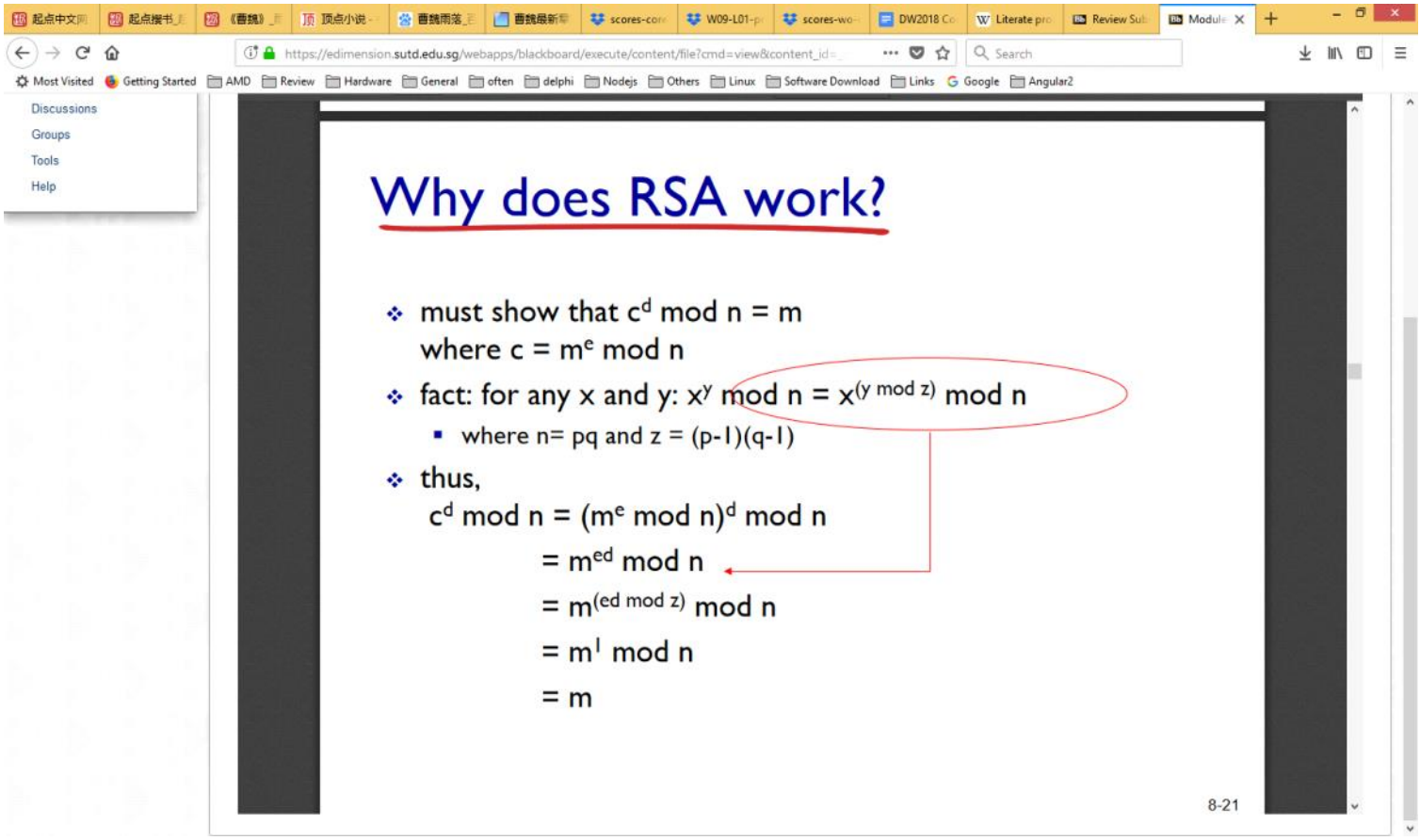
The first thing is: You need to pick two large prime numbers. Note the word large, it means 1024 bit

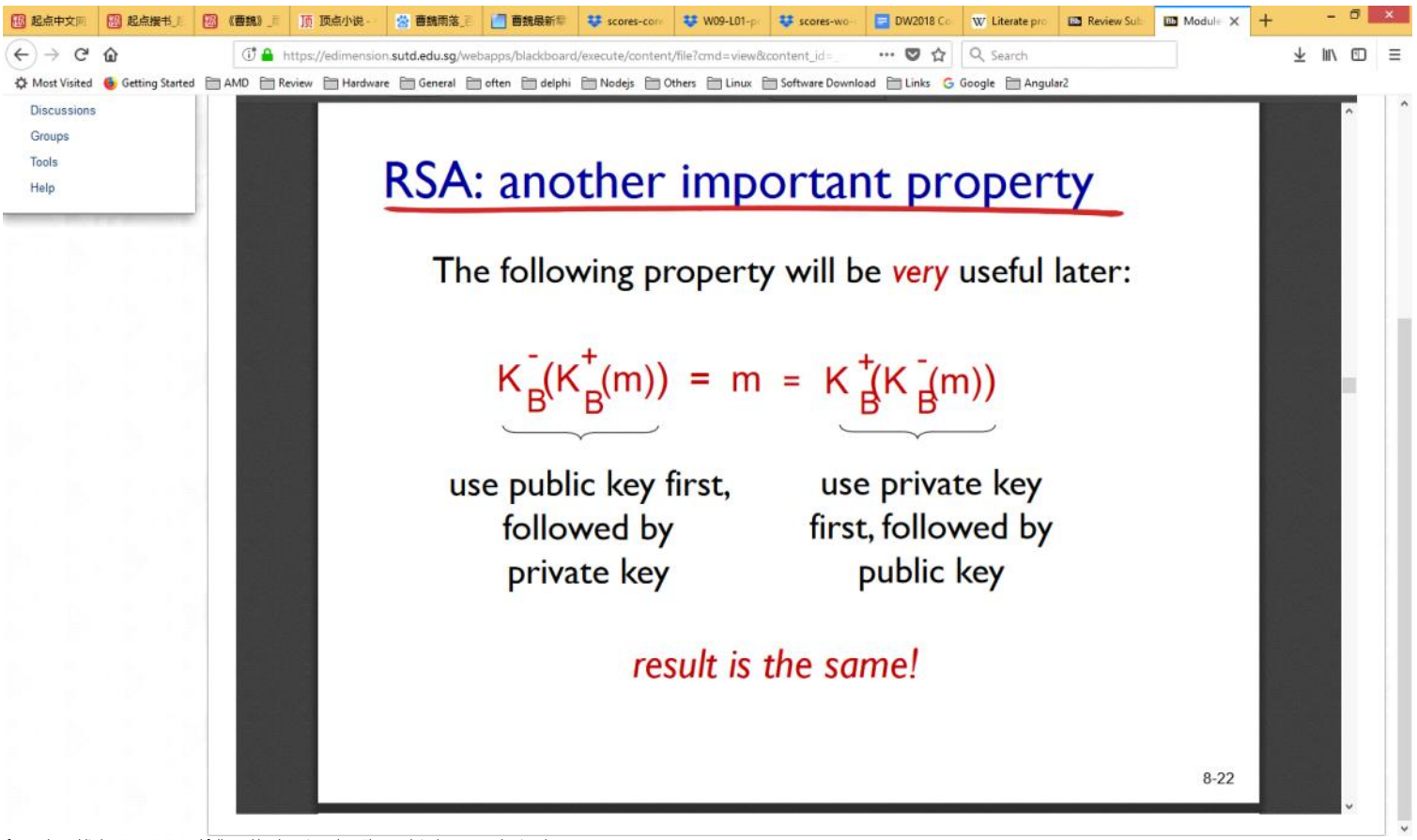
numbers. Compute n = pq, z = p-1, q-1
Choose e (with e< n) that has no common factors with z (e,z are relatively prime")
Choose d such that ed -1 is exactly divisible by z.
E is for the encryption key, d is for the decryption key.

N = 21. Z = 2*6=12 e = 5, d would be 5, so what's the point of having the encryption and decryption key be the same?
They are not large prime numbers, so they need to be on the order of 1024 bits.



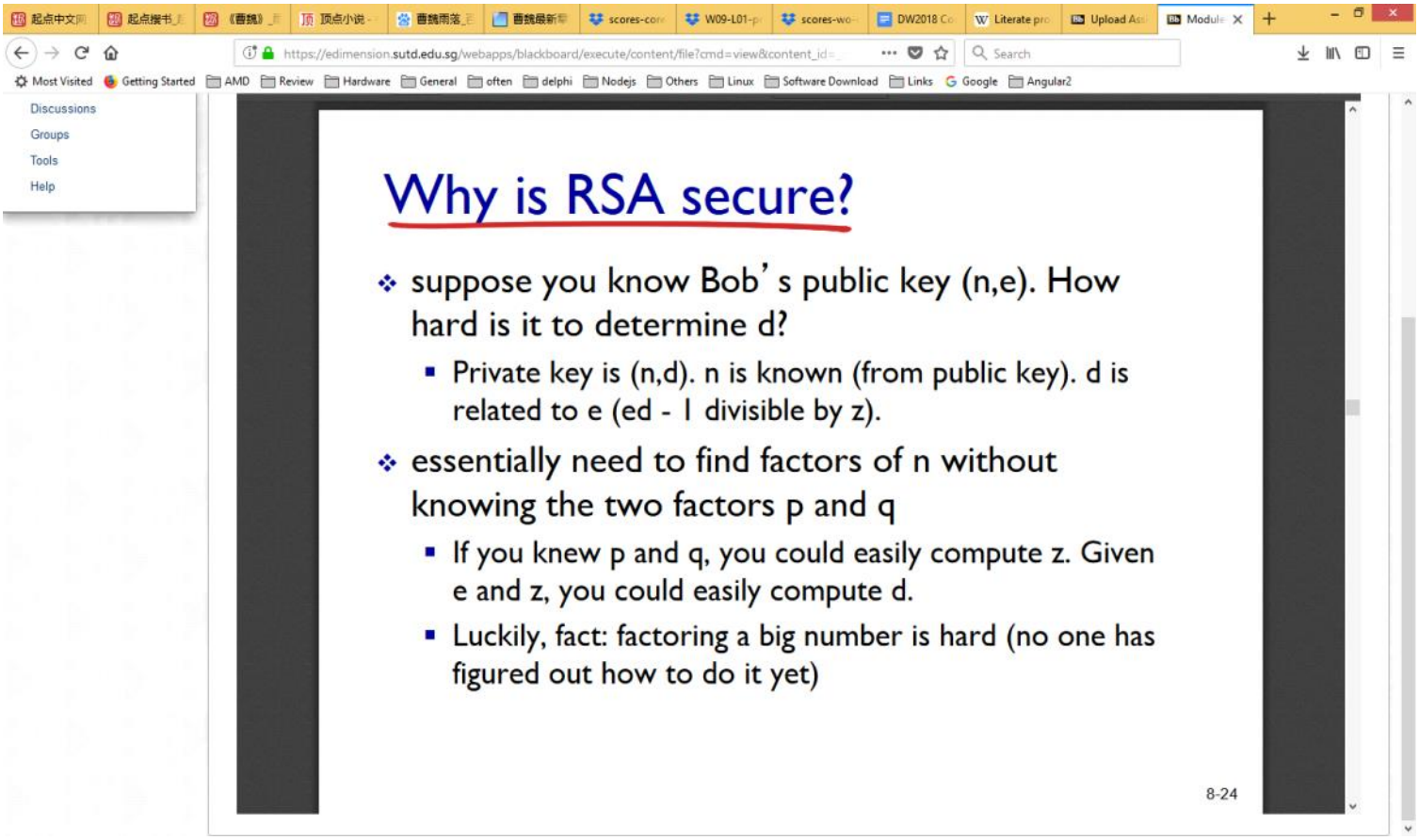
Why does RSA work?
Must show that c^d mod n = m where c = m^e mod n.
d = c^d mod n.
= (m^e mod n) ^ d mod n
Fact: for any x and y: x ^ y mod n = x ^(y mod z) * mod n
Where n = pq and z = (p-1)(q-1)



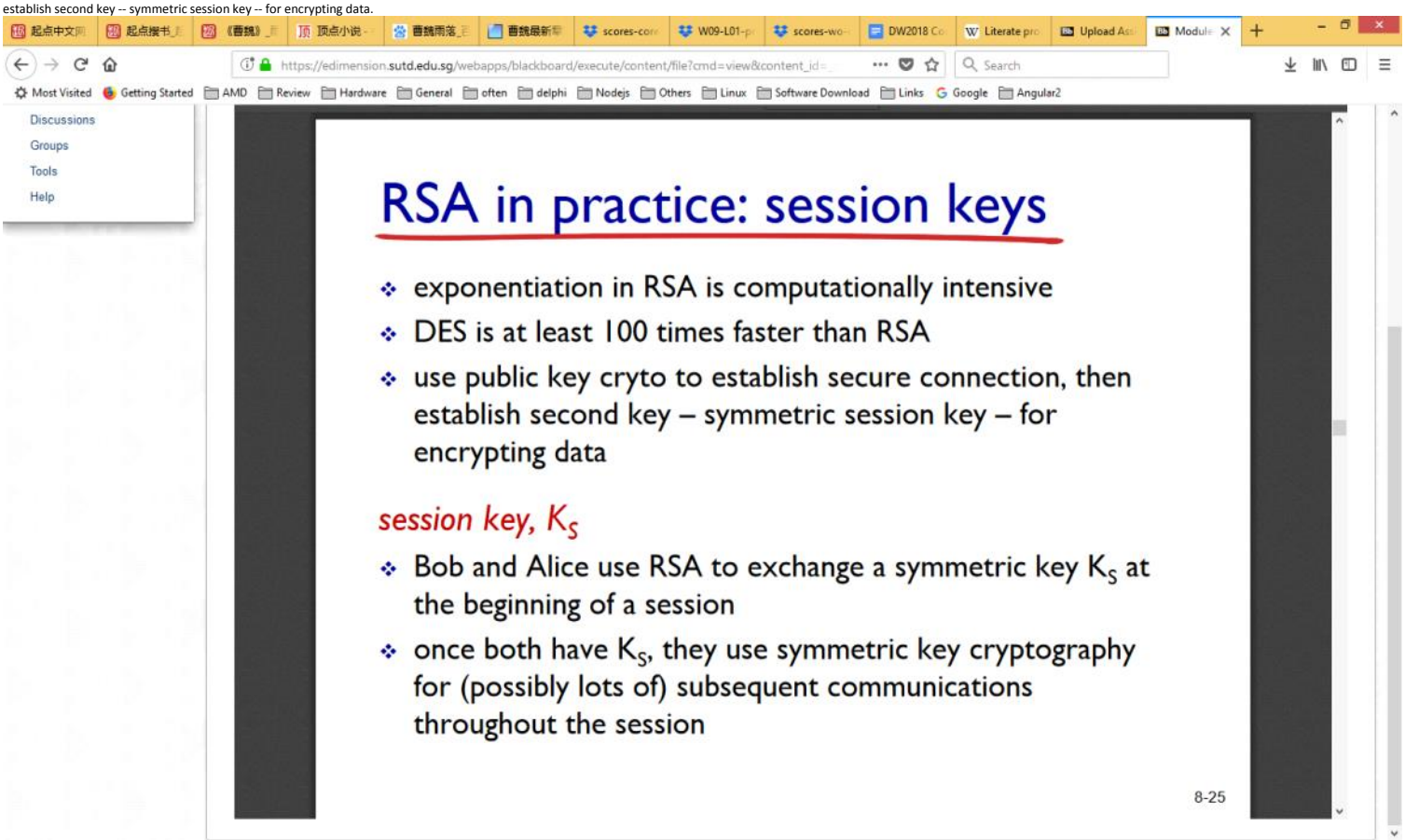


If I use the public key to encrypt and followed by the private key. The result is the same as having the private key first and followed by the public key. The idea of this key is that of a signature. That something like a signature, that authenticates me. There are other cases of people pretending to be me is often.

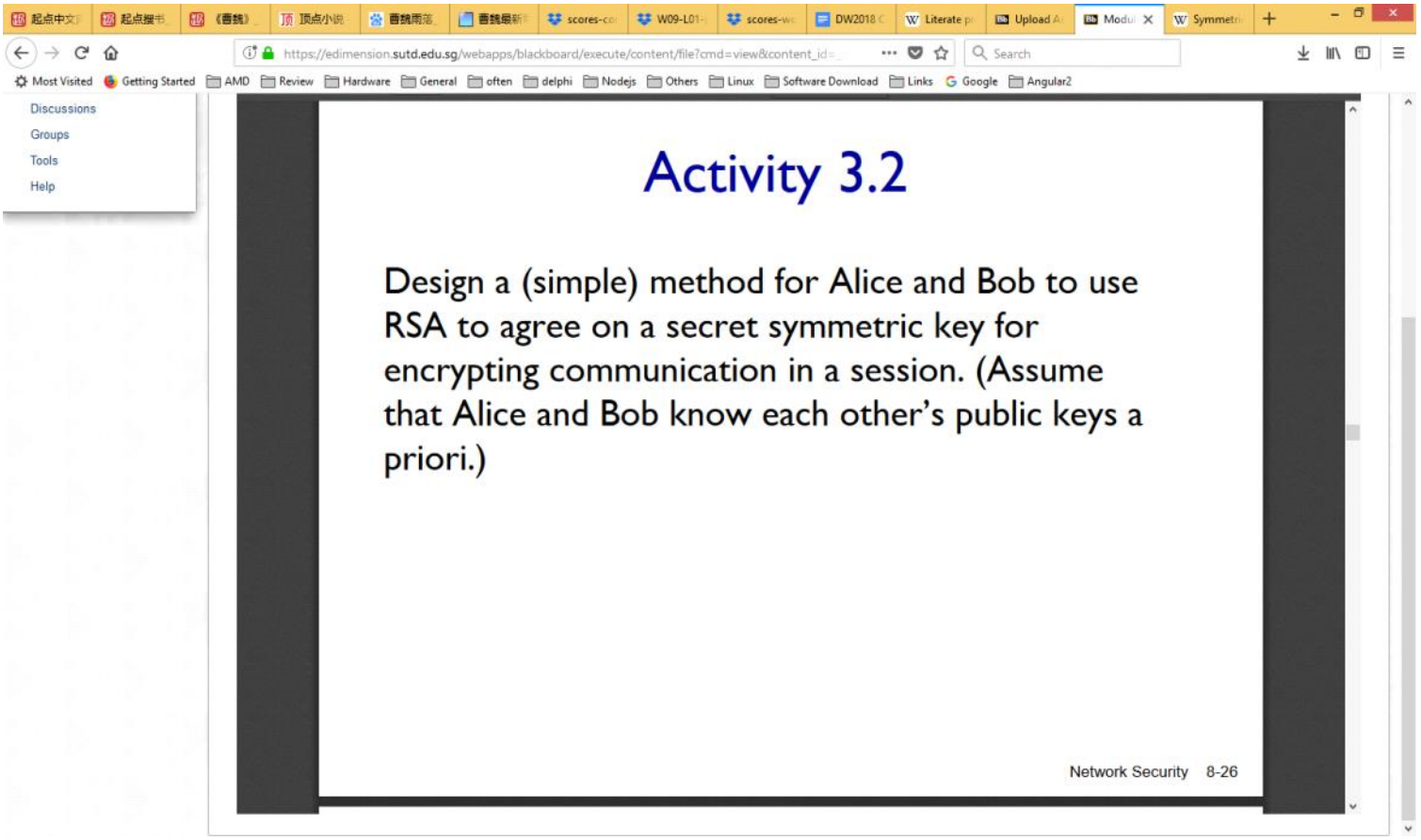
The basic idea is authenticating the identity of the sender. There is only one person who has that private key. I happen to have the public key of that authenticity.



RSA is intensive. Modulo, and all that is quite time consuming. If you have to do that with every part of the message, it would be slow. It would use a public key crypto to establish secure connection, then establish second key -- symmetric session key -- for encrypting data.



Like enigma, monitor the passing of the key of that day in very secret settings.



One of them needs to generate a secret key. A secret key is something like a large key used for encrypting (64 bits long, maybe 128 bits) and use something like openSSL which is widely available. You can generate your own DES or AES key. Alice is asking Bob, Bob is sending a DES key to Alice. So that can be out in the open. That secret key, Ks, which you got from open SSl has to be encoded. Bob would use Alice's public key to import that. Bob uses Alice's key to encode the symmetric key and sends it over to Alice. Alice would apply her own private key to recover the shared secret key. The weak point of the symmetric key is not possible to use.

We are getting the shared key communicated. We need authentication. Which is our next topic. Alice and Bob needs to agree on a symmetric key, and share the symmetric key for that session. So sessions are not that long, it doesn't hurt. Because we are using long keys, it would take too long to recover the shared key.

Monday - End Point Authentication

Monday, March 26, 2018 8:40 AM

So first thing is, we saw last week. The first protocol: Alice says "I am Alice". This can be compromised very easily because anyone can pretend to be Alice because the network. You do not see the other party, you cannot identify the other like in a b ank. So the next thing you try is the IP address.

The question came up whether the person behind a DHCP, configuration protocol where the protocol is handing out new IP addres ses. You get an IP address from a subnet, appended to the rest of the IP addresses and sent out. Whether you do this with a DHCP address or you do it with the rest of the addresses. The intruder could actually fake the IP address. The source code of Linux is publicly available. One can just root out the machine and manipulate the source code and do malicious things like ch anging the IP address. So it can be compromised. If you compromise the router, you have the ability to change the source and the add ress.

You can send a packet out with an address other than you. You can have a packet spoofing approach. How about another try? In this case, at this IP address, here's my pass word. I am Alice. You can intercept this message, the intruder can intercept. T here are certain protocols like Telnet, and when you login, the password is sent across. Anyone observing the server/telnet client or on the same as the telnet server would be able to see those packets. This is one favorite line of attack and a lot of pass words were compromised this way. This is a very popular way of gathering passwords, of stealing passwords. This would not work. Thi s is called an eavesdrop attack.

Why eavesdrop? Intruder is eavesdropping on everything that is going on and looking out through passwords. This is compromise d by a tape recorder. And mind that data to steal the passwords. The intruder comes along and spoof. And pretend to be Alice. This must be the correct person I am communicating with. The malicious router, the destination address. It is said to be the IP address of Alice. Whoever that's tapped in to do the spoofing would direct the packet to say to himself. This is an insec ure protocol.

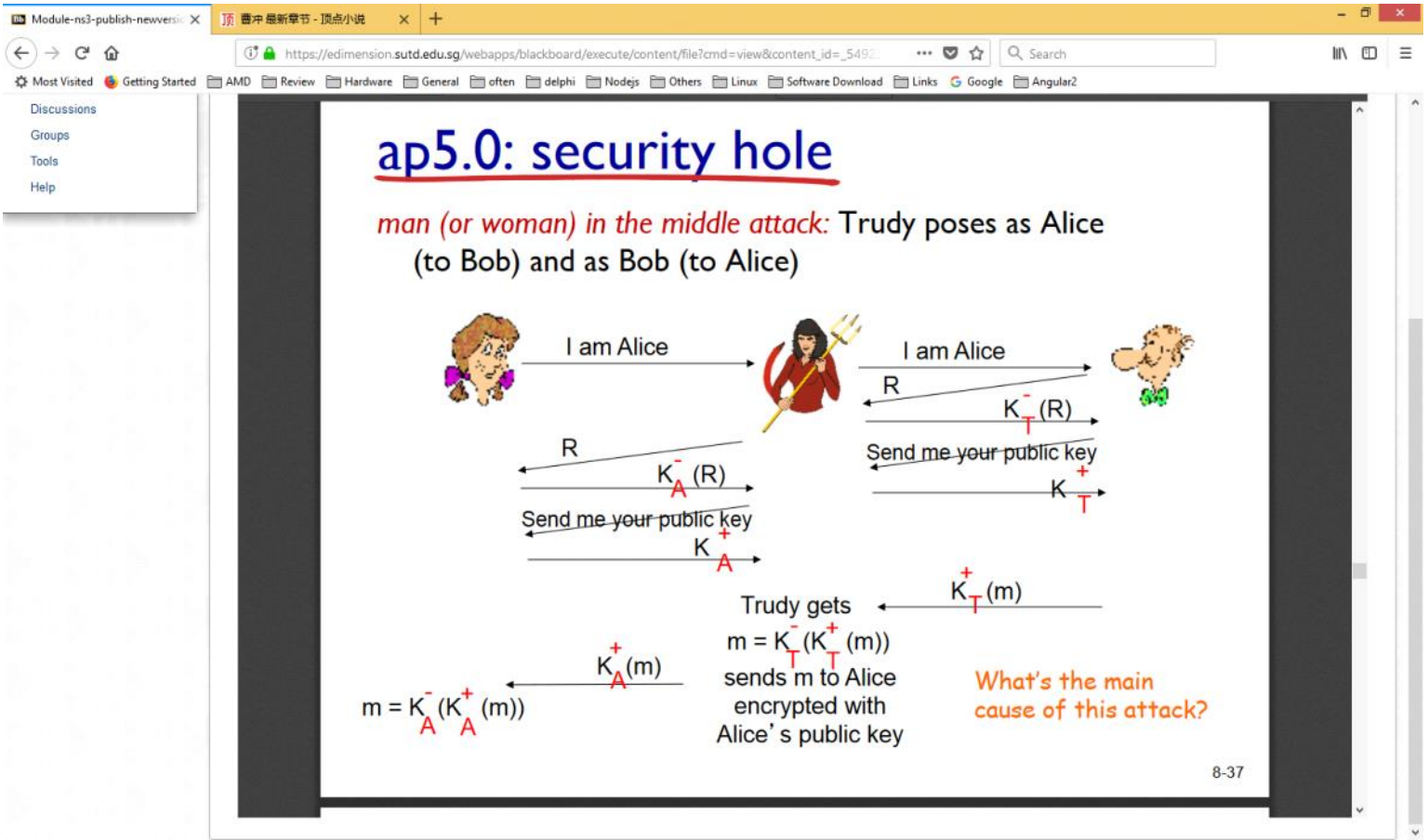
We can't avoid the fact that there would be someone trying to collect passwords. That's the same here, we try to encrypt the password. Let's say that the encrypted password uses something a key that is only known to Alice and Bob. Would this work?

You can just copy it, you don't need to interpret the password. It's not called an eavesdrop attack, it is now known as a pla yback because you do it at a later point in time. You still use eavesdropping technique to get the password. So you are tryin g to authenticate someone that is live. They were recording it and using something that you are attacking. Now we are trying to au thenticate a live party. So can you suggest a way to authenticate a live party. Whenever someone is trying to record and play back. So keep changing the key. But you are on the right track.

We call it a nonce. Nonce is a number ® used only once -in-a-lifetime. So when there is this person claiming to be Alice. R is a nonce, it is a randomly generated number that happens onl y once. Alice uses the symmetric key to encrypt R and send it to Bob. Bob can decrypt R and say that this is the nonce I have just sent and I am talking to Alice and not a prerecorded message.

There could be someone standing below, listened to by the third party. The intruder pretended to be Alice. So we need a way t o guarantee that Alice was the one who sent the message. If you use the public key, but only Alice would have Alice's private key. If Alice were to use a private key, it is only Alice that could have generated this encrypted message. So Alice sends a messa ge, says "I am Alice". Bob provides a nonce. Alice encrypts using a private key. Bob knows that it has to be Alice because no body else has the private key of Alice. Now Bob replies with a nonce. It need not be included. But once it gets here, When Alice encry pts the nonce with her own private key. Only the owner has the private key. The whole world has the public key. So what comes over here can only be encrypted by Alice and no one else.

In this version, Bob doesn't have to have Alice's public key. It can happen. You know everybody else's public key. If you are regularly communicating with the other party, you will not get regular public keys. Please send me your public key. If the in truder can actually capture these things and pose to be Alice.
Think of the intruder coming between everything. Intruder has the view of all the packet key calling out. When the nonce come s, the intruder then encrypts that with her own private key. The intruder pretends to be Alice. So now Bob thinks that this i s Alice. So send me your public key. It is apprehended here, what is applied here is the intruder's public key and not Alice's public key. But on the other side, the intruder carries on with this decision that says nonce. And says I a Bob. So the intru der pretends to be Bob to Alice and Alice to Bob. So then Alice would say that here is the encrypted version of Nonce and Alice thinks tha t this request is from Bob. And now the intruder gets the public key of Alice and the intruder can actually decrypt any messa ge that comes from Alice thereafter. So Alice sends messages to the intruder thinking this is Bob. This is called the man in the midd le attack.



The main cause of this attack is the man in the middle. He knows everything that happens between them. The problem is that Tr udy receives all messages as well. The intruder is totally transparent (that is to say, they think they are talking to anothe r always at the same time without knowing about the intruder)

Digital Signatures. It needs to be signed by an entity. So that the whole world would know that the person who assigned it. W hen we have the public key and the private key, that would be a powerful way to digitally sign everything. Whoever I sent it to would know that his document might have originated from me. A private key is a way to digitally sign the email. The problem i s that the encryption here is very expensive. And we don't need this document to be encrypted. Maybe it is something you can read, like routers.

Routers advertise their routing tables and the cost of each link. These are all routers and these routing tables are sent to each other every so often in their routing tables. You do not need to encrypt everything, but the sender must be validated. T he sender must have a digital signature so that everyone knows that this is the right person who has sent this routing signature. I can use my private key as a digital signature, but this is expensive if this is a large document.

We use a cryptographic technique where we use something like a handwritten signature that is obtained from the message but is not as large. The other benefit is that it is verifiable and non forgeable. The recipient can prove to someone that Bob, and no one else (including Alice), must have signed the document (non -repudiation)

Non repudiation means that A has proved that this message was sent to B and B cannot deny this message was sent by A. Can we use symmetric key for non-repudiation.

Use the public/symmetric key to sign a document

Instead of sign the document, we can sign the key itself. We just produce an encrypted form of the key itself.

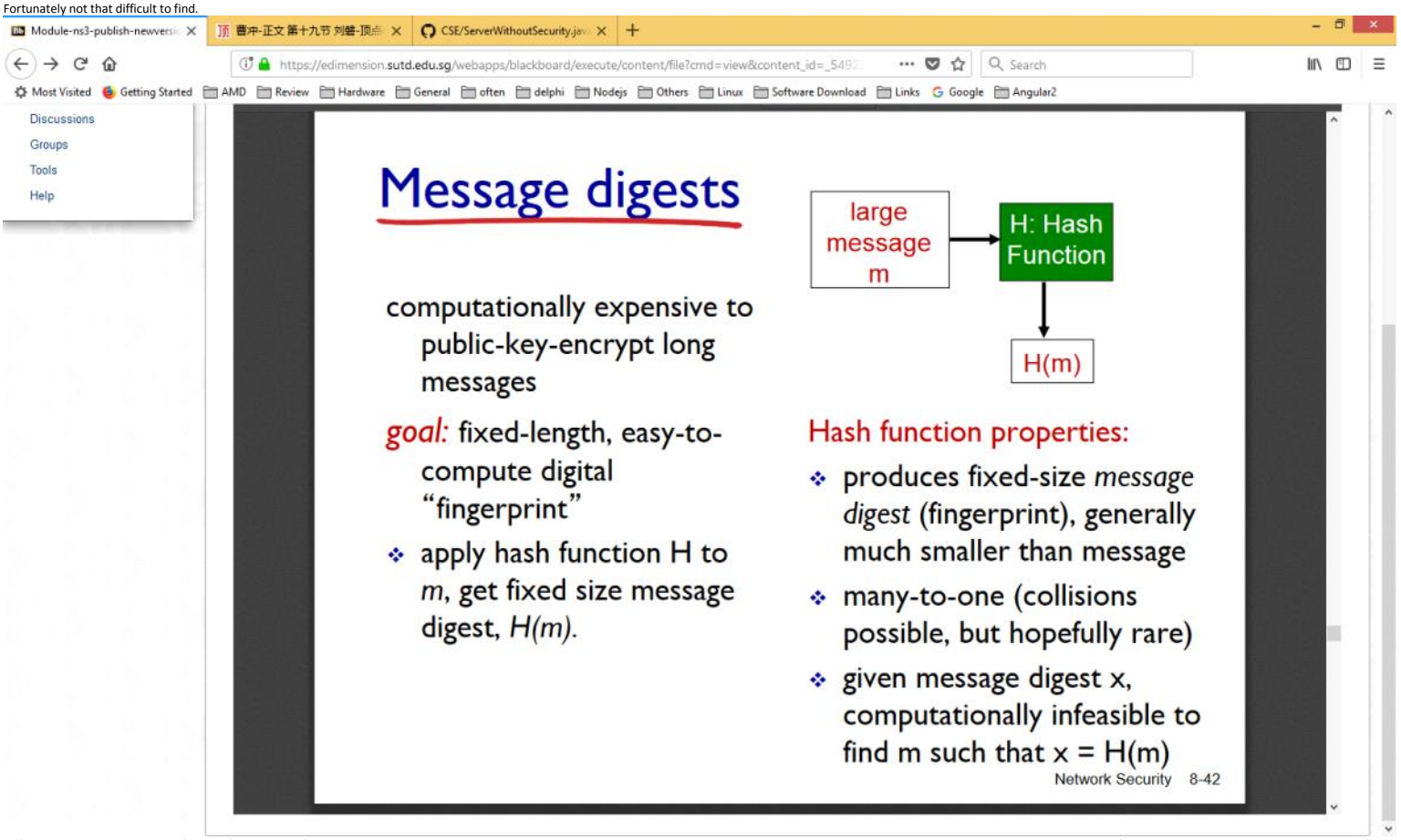
Can we use the symmetric key as a means of non-repudiation?

Public key is a good candidate but symmetric is not. Because Bob and Alice could claim they have sent that message.
Suppose Alice receives m with signature m, Kb-(m)
Alice verifies m signed by Bob by applying Bob's public key Kb+ to Kb -(m) and then checks that it is the original message m.
If they are equal, hwoever signed m must have used Bob's private key.
Alice verifies that Bob signed m and no one else signed m. Bob signed m and not m'. The integrity of the message has not been altered.

Public key solves message integrity problem because if something has been signed by public key, it preserves confidentiality and provides message integrity.

It takes a key and produces a value which is very unlikely to be repeated. Compilers use hash table all the time. If I have a long variable name in a Java table. They will hash the long variable name and get a unique value. In a sample table. Whenever there is a collision, you can deal with it some other way or have a linked list. That does not work for message idgest because you nee d to guarantee that there would not be a collision. The two messages should not give the same hash value. We need a function whereby if you give it two messages, it is highly unlikely to get the same value. You need that kind of hash function.

Fortunately not that difficult to find.



Let's say we use a common hash function (A check sum) . We use a check sum to get a hash value. To see whether it is a good choice or not. Where I say a hundred dollars and 99 cents and sending it across from Bob and we will use a check sum. Each of these is one byte so 32 bytes at a whole time. Translate to ascii codes. We get a sum of this 32 bit number. Let's say we hav e a another source message. Where 9 and 1 are flipped. When we use the sum, it yields exactly the same check sum. A check sum is obviously not a very good idea for a hash function.

Will birthdays work better as message digest?

The process of one entity proving its identity to another entity over a computer network.
"Live party" involved in a communication that is actually happening.

Wednesday

Wednesday, March 28, 2018 11:38 AM

To understand the problem a little bit more. Trudy is playing pizza prank on Bob. And Alice is a pizza store. This is something that Bob never wrote. This is something that Trudy wrote pretending to be Bob. Trudy sends it to the pizza store her public key, but say it's Bob's public key.

What's the difference? There is no secret message being compromised but it is something being quite undesirable. There being delivered to people who never really ordered them. Fake news on the internet.

This brings us to this notion of certification authority. It may be some entity like MDA or some equivalent entity in other countries. Or it could be Verisign. That acts on behalf of businesses and authenticates businesses in the united states. Every person or router will register his public key with the certificate authority. The ISP will say: this is my router, my IP address. And then the domain name of that router will need to be registered in the certification party. That party will have to provide certain proofs, with the ministry of commerce. Then the officials from that certification authority will visit. Only when they grant all those checks will they give it to this entity here.

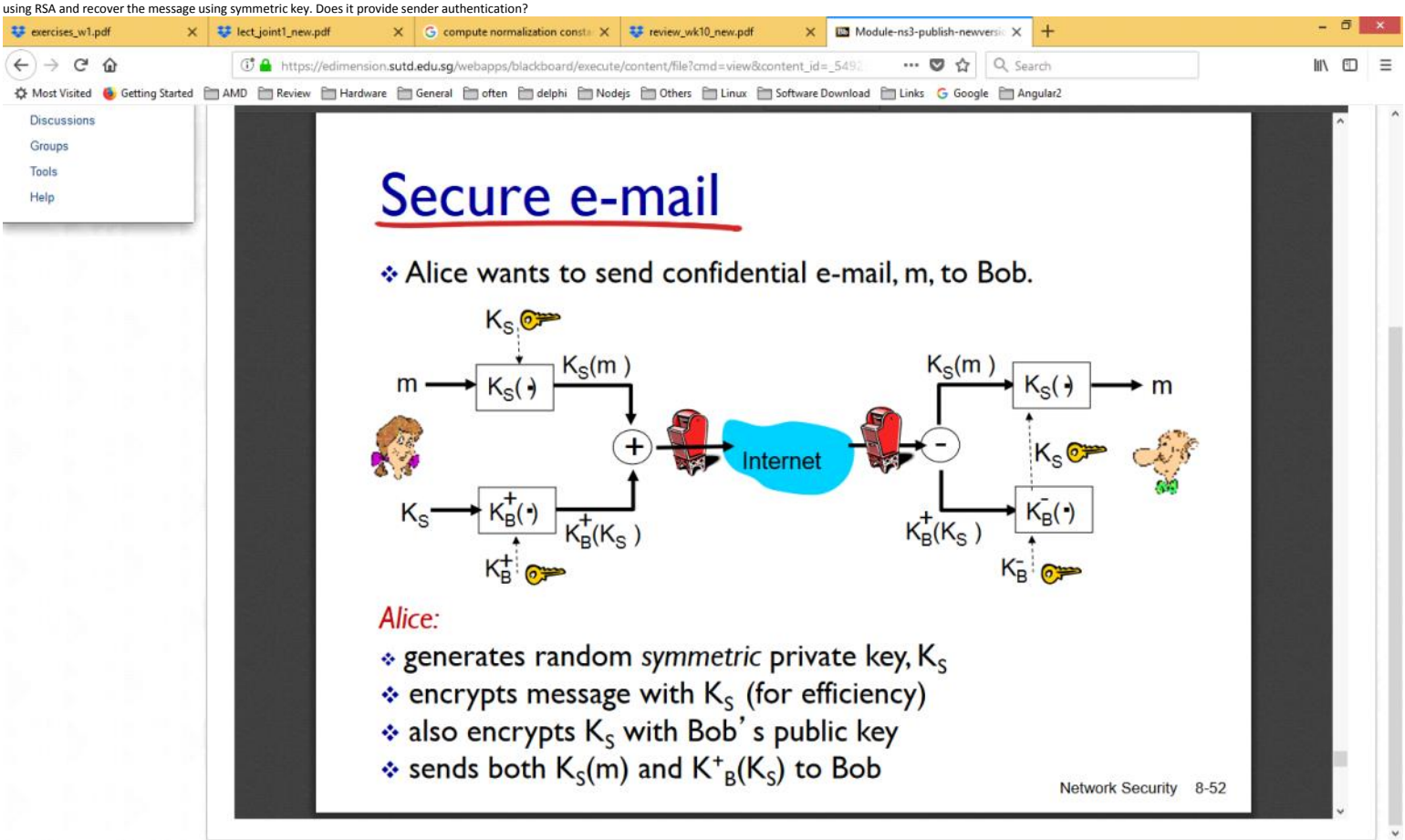
Our email address is quite an acceptable email address that is because we are leveraging on sutd.edu.sg is registered with the certification authority. With the IDA. That this domain is being registered. We have the public key which is digitally signed by whom? By Bob? The public key of Bob is digitally signed by the publication authority. They are the only ones who could produce that signature. What you show here is an electronic document. It is the certificate of Bob's public key signed by the certification authority. This indeed is Bob's public key. Somebody else produced a fake public key. Saying this is Bob's public key, Alice's public key. So now, by going through a certification authority, we can say that this is definitely this private key. Nobody questions certification authority.

They are the ones who certify that this is Bob or Alice.
When Alice wants Bob's public key. Are we trusting Bob? Or Alice? No, the ultimate trust is being placed on the certification authority. Nobody can fake to be IDA. Their public key is wellknown. We apply CA's public key to Bob's certificate, get Bob's public key. And with the certificate that we get from Bob or elsewhere.

Something across in the email, and someone sends me this certificate claiming to be this certificate of somebody. What happens if the browser that is installed.

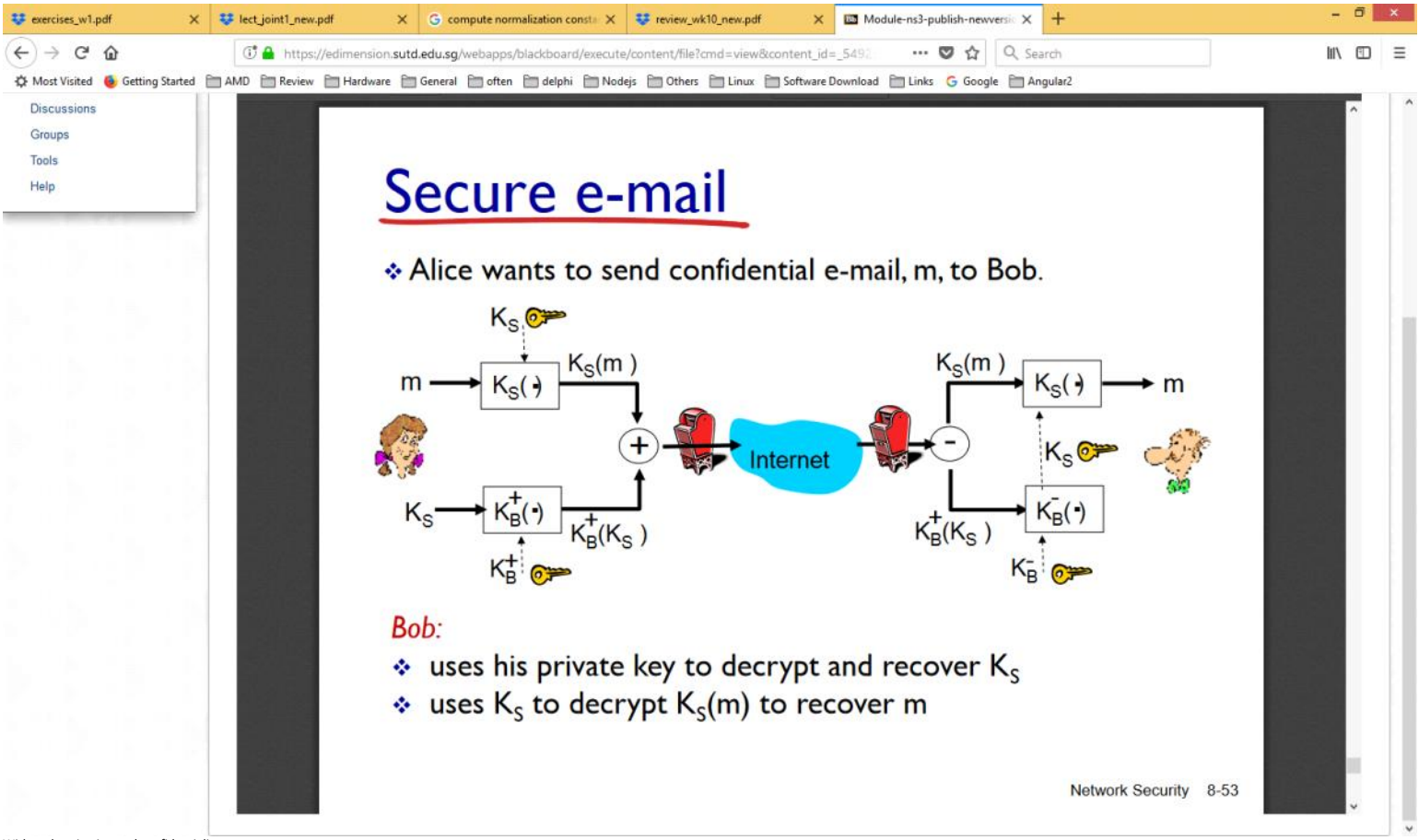
There is this property. There is sender verification. There is message integrity. Lastly, we have the receiver authentication.
The first thing is confidentiality. How can we have confidentiality? With symmetric key. Symmetric key is chosen for efficiency. We have signed the message with an efficient key by Alice, opens SSL, encrypts her message and takes Bob's public key to encrypt.

Also encrypts with Bob's public key. Sends both KS and this Kb_Ks to Bob. At the other end, there is a deconcatentation operation. Bob recovers the message, applies the message and recovers this message using RSA and recover the message using symmetric key. Does it provide sender authentication?



But we don't know that this is Alice on the other end. How would we. Now Alice wants to provide sender authentication message integrity. This is out in the open, a publisher. Knowing that this is not fake news but it is publicly readable. You need to produce a hash.

Get a hash of the message and then use Alice's private key to sign the hash. You know that Alice could have signed this and puts it out on the internet. First of all, the hash is recovered using Alice's public key and the message is taken with the same hash function. And then a comparison is made between the hash value here and here. They are the same. They know that the sender is authenticated. This is a sender email with authentication but no confidentiality.



With authentication and confidentiality.
Confidentiality is obtained through the symmetric key. The message is concatenated with the encrypted hash. What does this stage do?
This stage basically takes the hash. Except that we also encrypted using a secret key. This secret key is encrypted using the recipient's public key. This secret key is using the recipient's public key and sent over the internet. Alice uses three keys. The private key of herself to validate that it is her. Bob's public key is to generate a symmetric key to Bob. Symmetric key is submitted for confidentiality. The symmetric key is for confidentiality and efficiency. So that is efficiency. Then Alice's private key is used here to encrypt the hash so that Bob can use Alice's public key to get the hash value for message integrity.
Alice's private key is used to authenticate. The message integrity comes from Alice's private key because it is digitally signed.

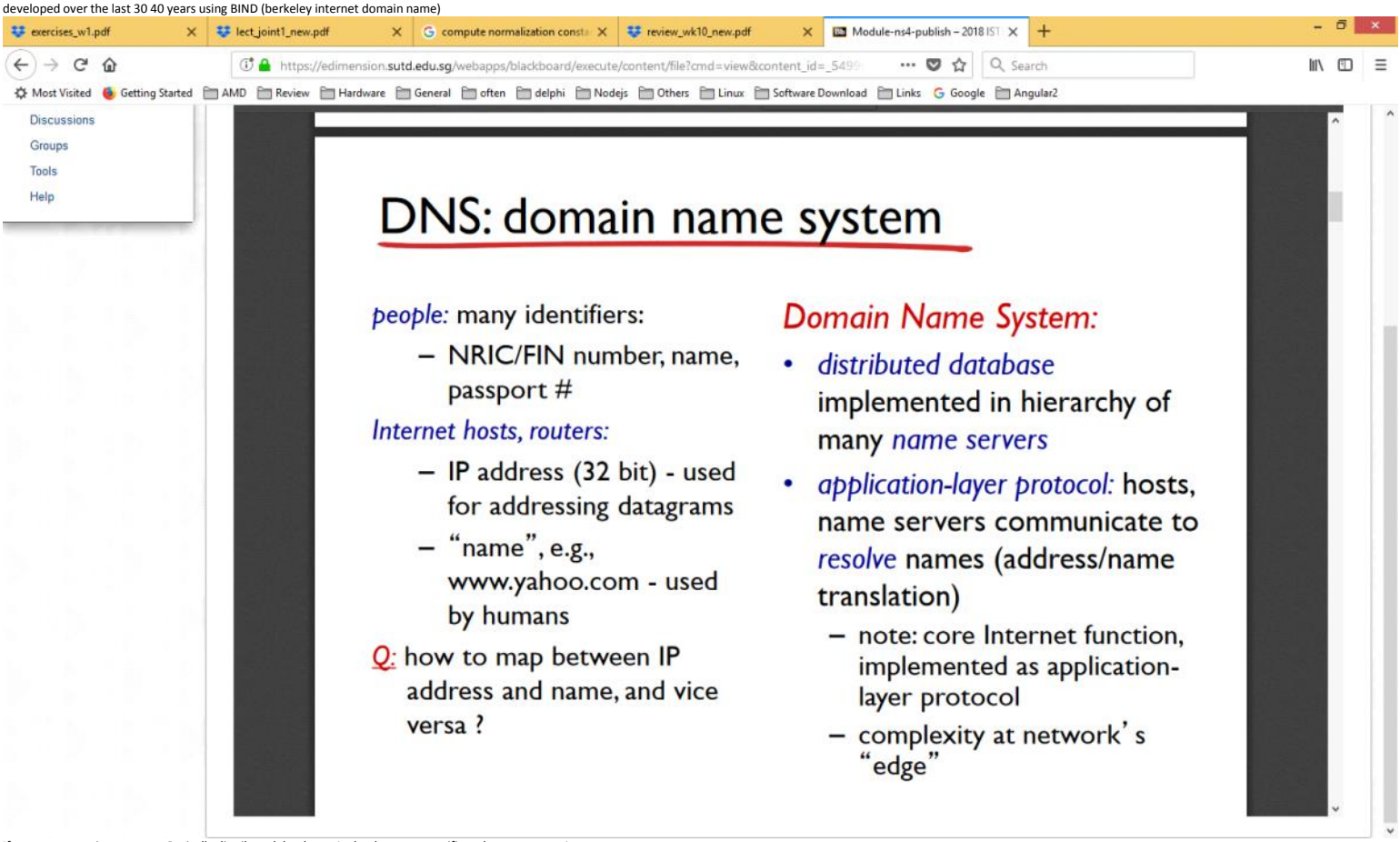
We have not done Receiver's authentication. Before the email takes place in the first place, Alice would have to know that this is Bob's public key. Through the CA. Through the email communication with one another. You go through gmail. Gmail is doing some of this for us.

Wed- DNS

Wednesday, March 28, 2018 12:08 PM

All of us has a name and a number associated with us. What you need is a mapping between a name like that and a valid IP address. A 32 bit number. Again, driving our intuition, the whole purpose of this is to actually be able, for a router to find out where this machine is and to route packets. As it happens, we go from more general to more specific, to left to right. You have a subnet mask which masks the last few bits of the IP address. And then be more specific and reach the machine on a subnet, you need the last few bits as well

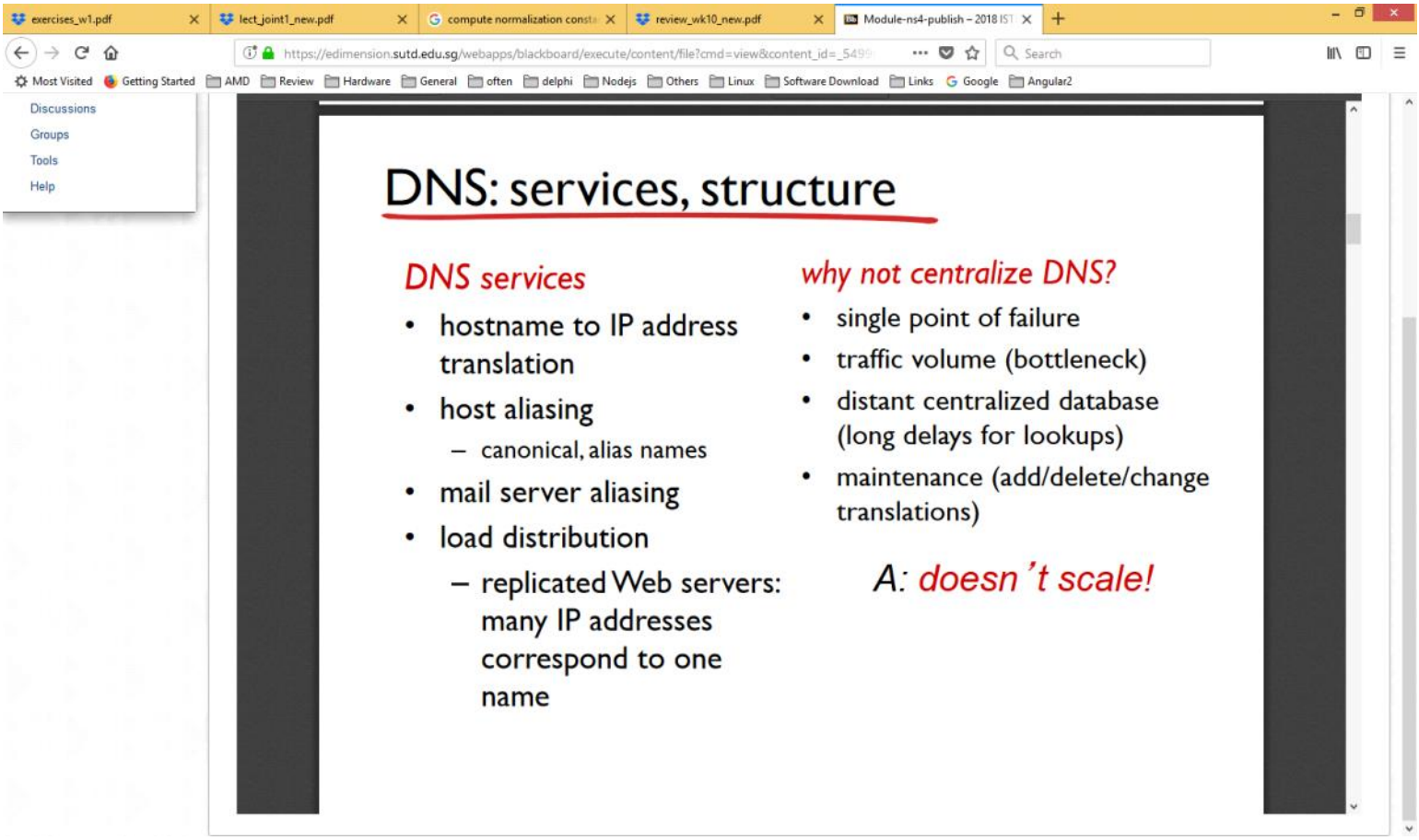
Similar things happen on your own address. I have an address in Clementi with a block and a number, we are going from more general (Singapore) to a more priority Address (570, #02-129) And we need a translation from name to IP address. This needs to be all over the world. This was evolutionary, it developed over the last 30 40 years using BIND (berkeley internet domain name)



If you were to write a system. Basically distributed database. And to be more specific, a data structure. A mapping from names to IP addresses. How would you write such a distributed data structure. As a set of name servers. We need to have a collection of named servers in cooperation with one another. If this named server was set right, that would be sort of a name approach.

We use names all the time, not IP addresses. Some of us might remember IP addresses. So in addition to that distributive data base, the set of notation could get back.

What do we mean by host aliasing? When we use something like google.com. We use google.com. But there's no way there's only one machine in the world that uses google.com. But we can just access it using google.com. There are these canonical names that are hidden from us. In the west coast of the united states, it could go from mail alias 1/2 .westcoast.google.com Or maybe amazon web server. All these aliasing is handled by DNS. This could also be on the level of mail servers. Our web server and mail server could have all these names. Last but not least is load distribution. There could be some heavily used sites on the internet.

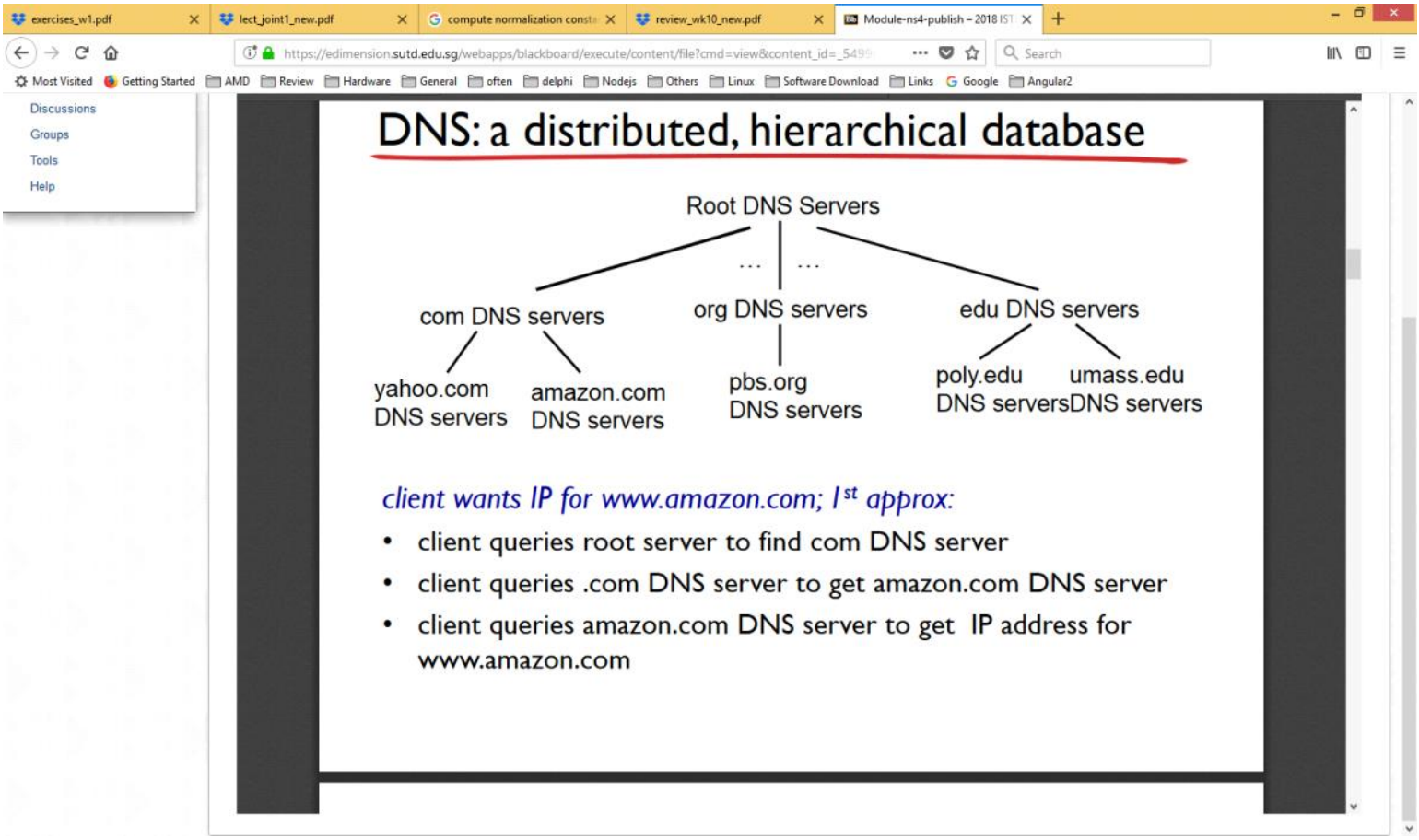


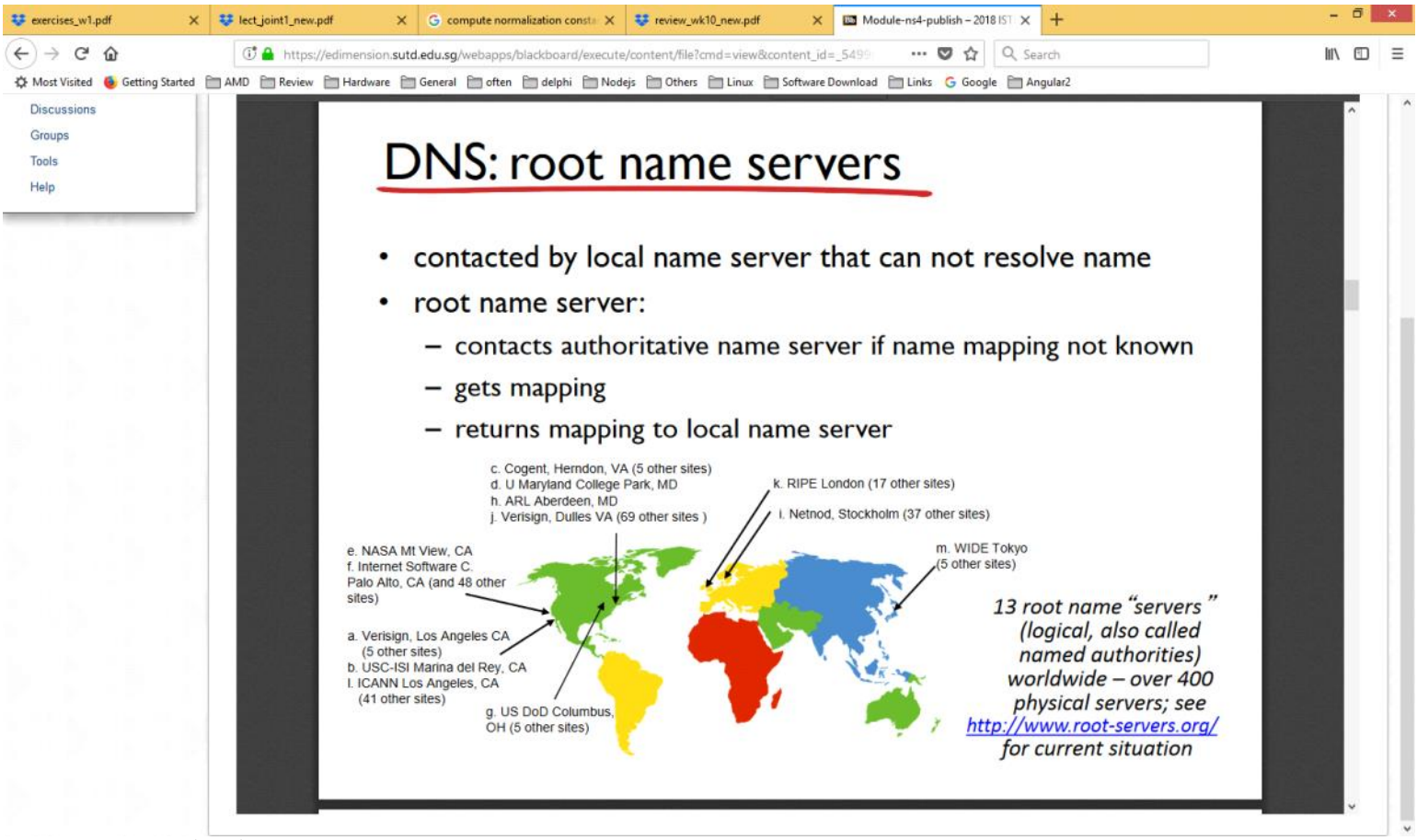
Like google and amazon. They would only corresponding to one name. How would that work? It needs to be some kind of a tree structure. A logical tree, not physical tree.

There are 13 root DNS servers, every unresolved domain name server requests will go to root. At the next level, we have what are known as tld /top level domain servers. As the name suggests, the top level domains are commonly known. .Net.Org as well as the country domain names like .au .sg for Singapore. So many others. At the next level we have the authoritative DNS servers. So let's say some client comes along. And this client could be a web browser with an embedded url and something other than the domain name, could be html resource.

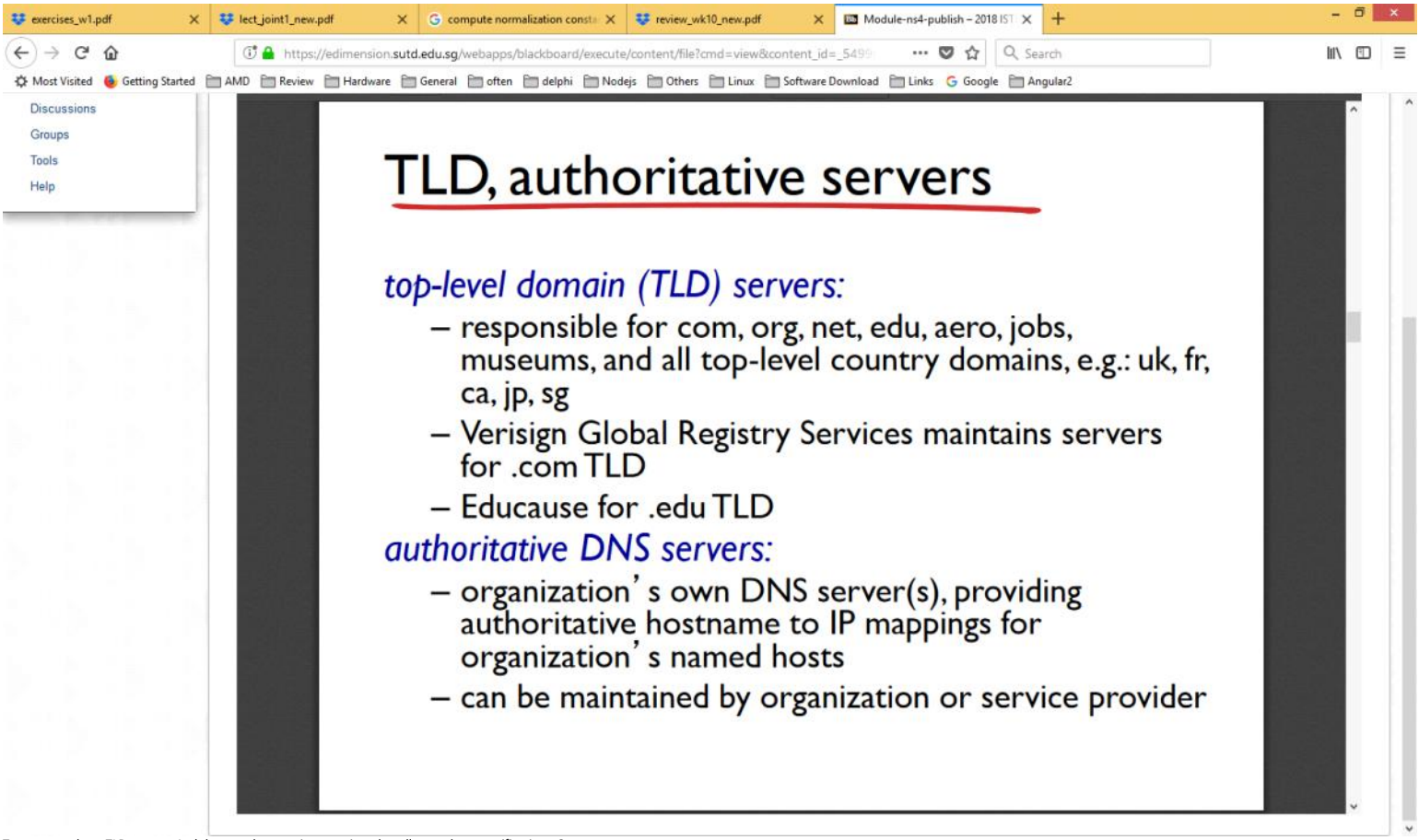
Solve and use for an IP address. And request that web page. This client wants an IP for www.amazon.com
The client would need to query th e root domain server. It is under .com. Looks for com DNS servers for dealing with all the dot com domain name servers. You are looking for Amazon, here is the authoritative domain name servers for amazon. So the second level would be to query the DNS server to get amazon.com DNS server.
Client queries amazon.com DNS server to get IP address for www. Amazon.com

It could be a number of hosts, and you could have load hosting and what nowt.





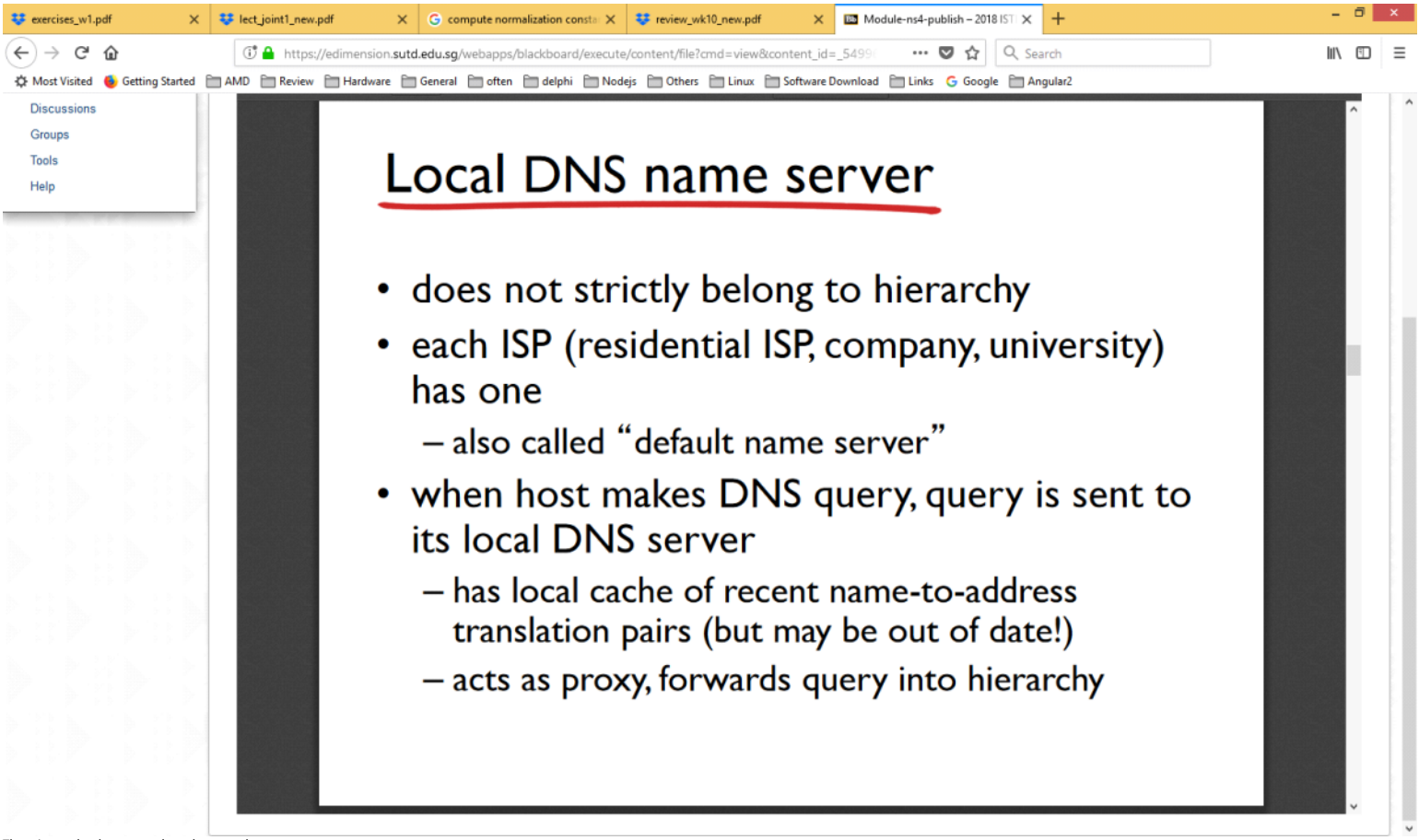
The particular site that responded (mit.edu) was more local. Somewhere in China. Which was responding to the traceroute request. Something similar would happen with the top level domain server. There could be something more local, but that is something else in the domain sattes.



To recap, we have TLD servers. And then you have registry services that allocate those certifications. So probably verisign that allocate to certificates and maintain the servers of dot com. Similarly, we have the company called edu host that hosts all the .edu IP addresses.

This brings us to the topic of local dns name server. Doesn't belong to hierarchy. We never saw it before. However, on every network, you do have local entity like domain name server. The ISP local domain name server will serve all our servers as well. To which all bearings is redirected. IF the local dns name server is already there in the data structure, it will not go to the authoritative name server and will respond to the request.

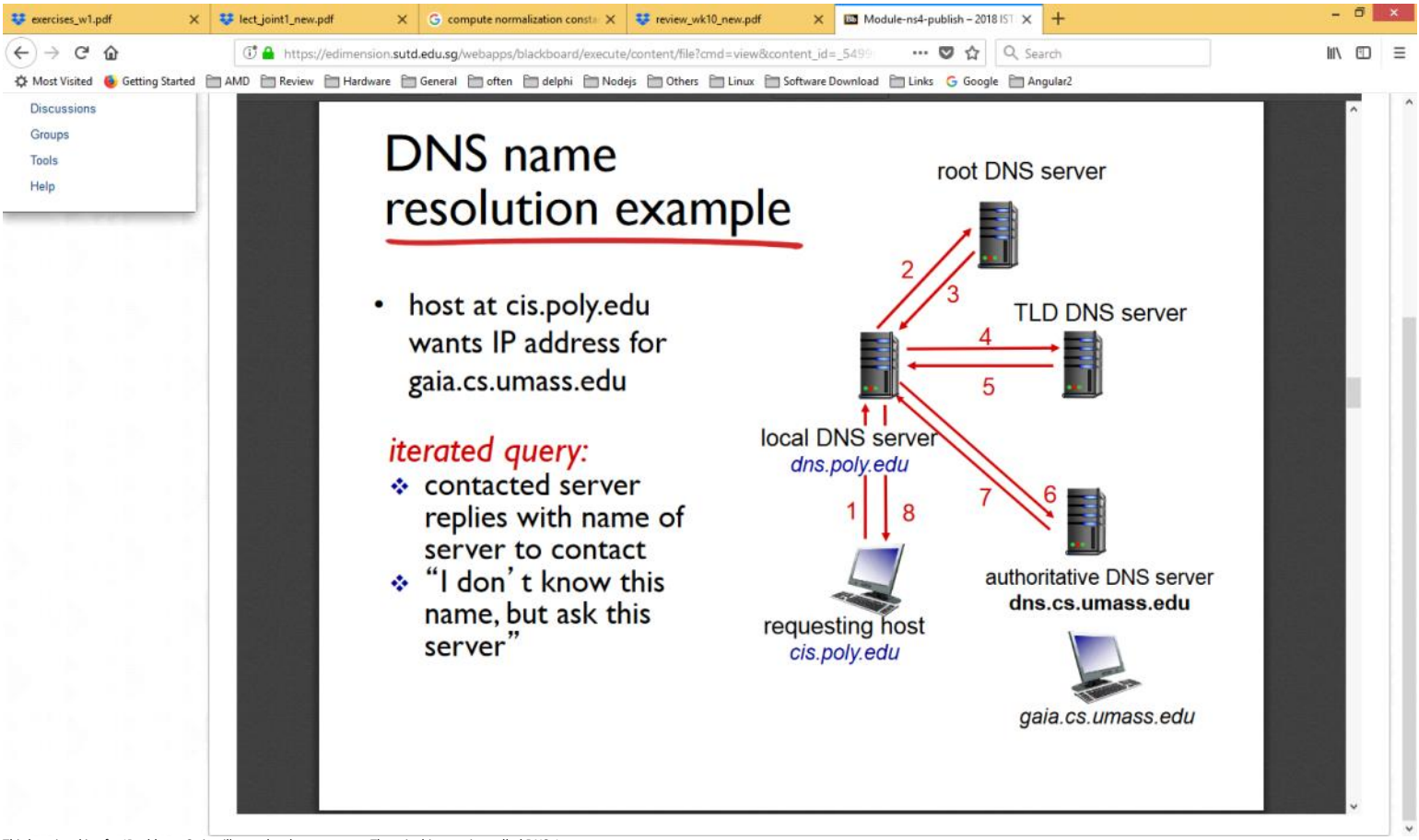
It will have its local cache of recent name to address translation pairs (kept for two days)



There is a cache that responds to the second request.

But we know that http pages changes frequently. But domain names and address pages changes as frequently as http web pages. I won't be that sure because we are operating with IP addresses that change when we come in to SUTD. We go through a DHCP and we get a new IP address. When I leave and go to another place, I will get another DHCP IP address. Domain Host Protocol. Use for a little while as long as I use the LAN I am logged into. I will be assigned for a little while. It is not true that the domain name and IP address is a long lived entity. There are proposals to actually resolve down to this level. In the past, any DHCP address was not resolved by DNS. It had its own applicability.

Nowadays, we see the case of wanting to resolve domain names to IP addresses even if its beyond the domain name server.

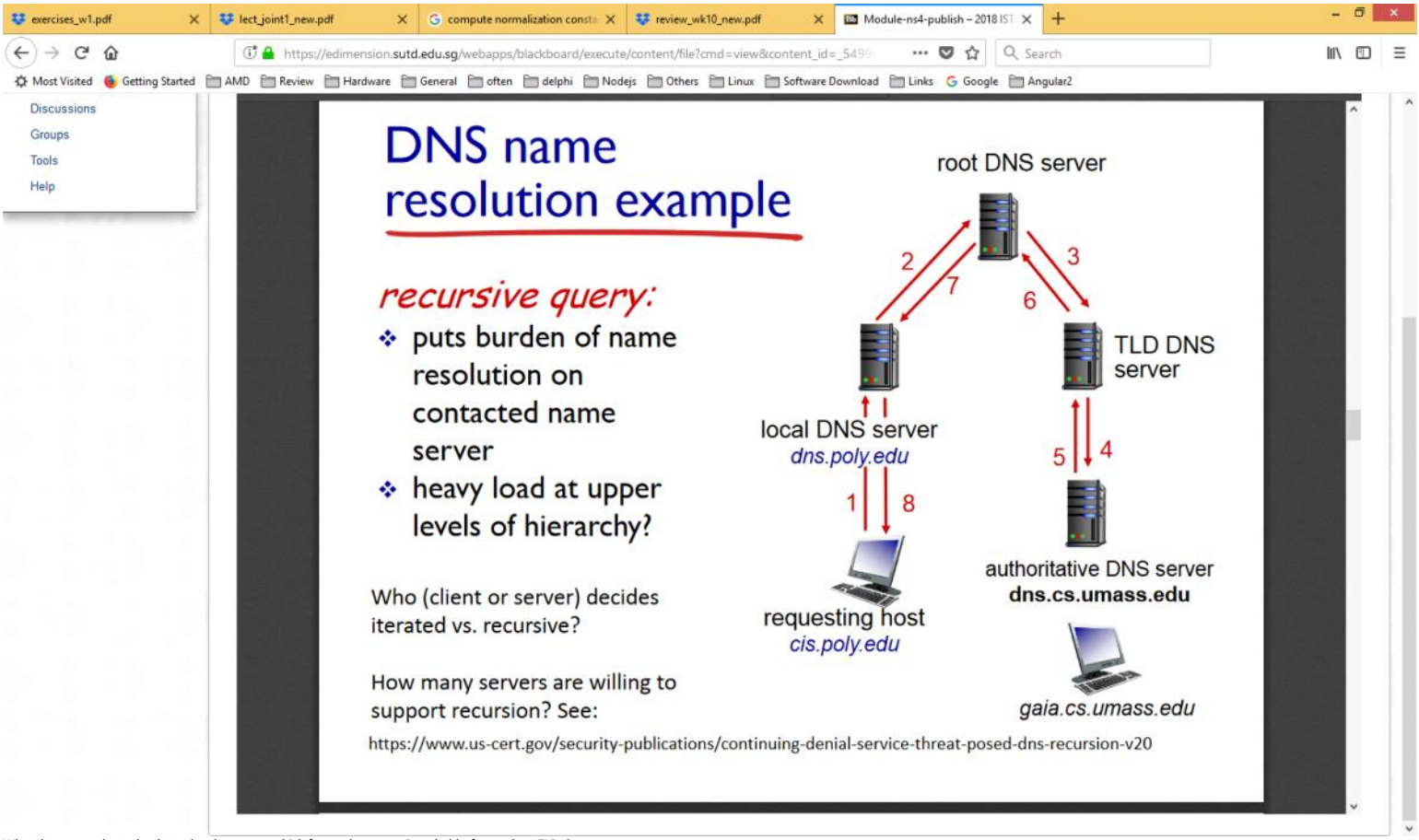


This host is asking for IP address. So it will go to local name server. There is this notation called DNS. It means that this DNS, although acting as a local domain name server. It is also the authoritative domain name server. In the same host, it is also the local domain name server. You can have large universities where the authoritative DNS and the local DNS is just a minor point. It need not be the same. It can have 20 domain name servers. You can have twenty or thirty students. And you can have several of them and some local dns. Servers. But this is a small university and it is the same domain name server. This request would be served in a following name.

This domain name does not exist in the cache and go to the root level. Say I don't have this IP address, but give you IP address of where you can look. Returns a top level domain name server address. And because it is the edu, it will be an edu TLD server. The next request is forwarded to the TLD to look up its IP addresses. So TLD doesn't have, will go for authoritative DNS server that may have. There is the IP address of Umass sent to local DNS server. So the local DNS looks for the authoritative DNS server and will have the IP address for that IP address. Because it is authoritative, it has the host you are looking for.

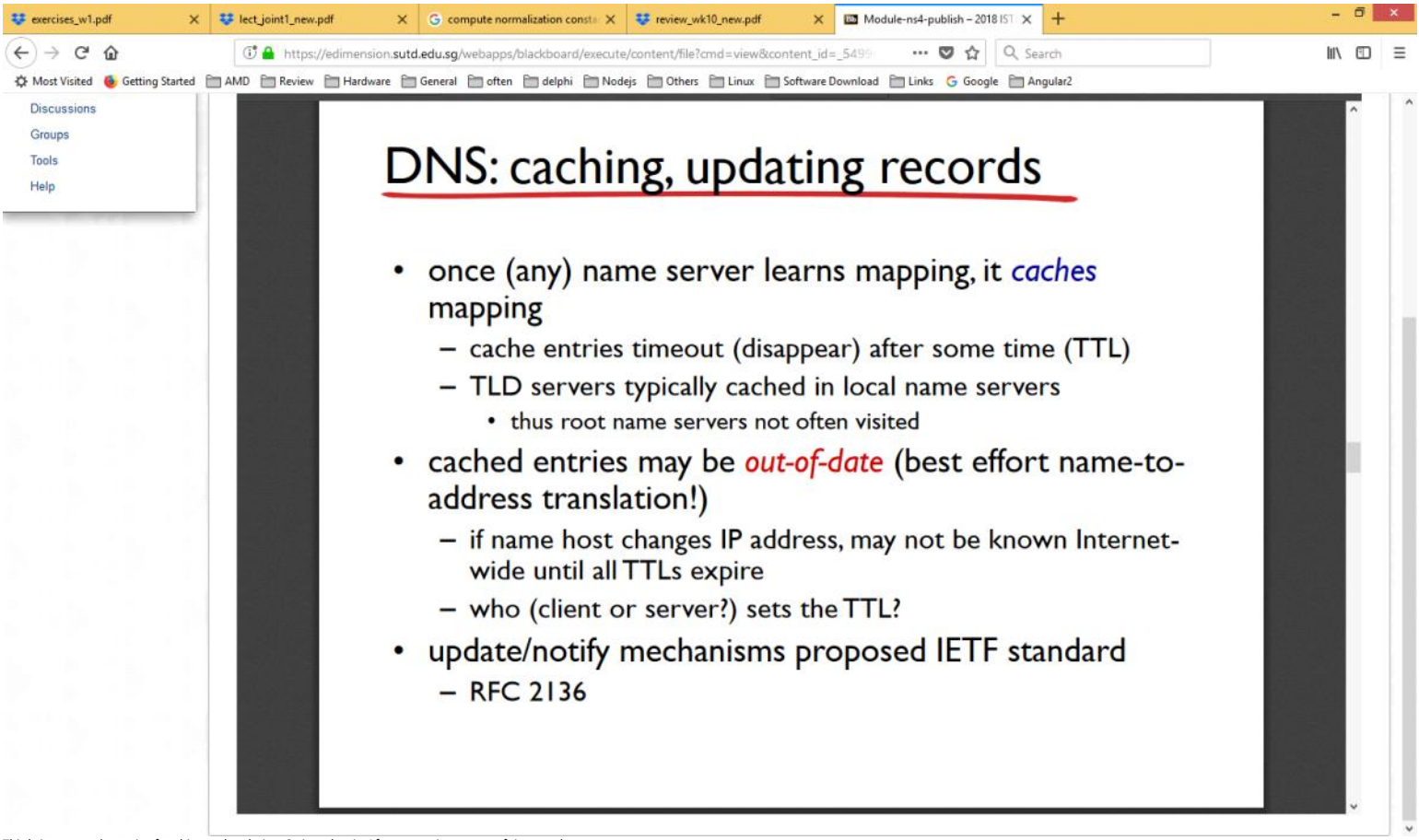
Just for one request, that was initiated with a mouse click on an IP address. Gives rise to all the mouse clicks on that IP internet. It is the iterated query, takes place through an iteration of a number of steps. Extending through the hierarchy of one domain name server. That is root, dot. TLD.

You could also have a recursive approach.



Whereby you can have the dns.poly.edu request which forwards to root. Root holds, forwards to TLD. So in recursion, you hold something on a stack and you resolve that function and location. Until this is resolved, the top level domain name server will be holding onto the request. The result would be forwarded in a chained fashion. What would be the pros and cons of changing this.

There would be heavy load on the upper levels. And they did a survey if those DNS servers whether the admin would be willing to do this. They are willing to support this heavy load as a service to the internet.



This brings us to the topic of caching and updating. So it caches it. After a certain amount of time, at the local level, it is about 2 days typically, that cache value is dropped. After some time, which we call time to live. At the top level servers, they typically cache it in local name servers. In their local name servers. Does the root name servers, it is not often repeated. The request may be resolved in the tld state itself. The cached entries maybe out of date. That is the reason why resolving the DHCP addresses. This may be out of date very quickly. So you have a best effort name to address translation.

Drawbacks: You have a few more bouncing around. It is an obsolete IP address. Your protocol would need to take that back into account. But host names don't change very often. But it is likely to change. If your company changes its name. If the name host changes IP addresses, may not be known until TTL expire. To figure out that this IP name changes. Who sets the TTL (time to live?)

Client or server? They are being reinstated because of this proposal that this DHCP IP addresses should be held by the domain name servers. This is already being done by Microsoft. They do it that is microsoft specific. May proliferate beyond Microsoft and go to the rest of the internet community.

Every database has a metadata. So it is a nem, a value, a type and a ttl. And it is the domain name. The type is four types. Ythere is a host name, a domain name, an alias aor a name of mail server.

Thursday PA

Thursday, March 29, 2018 1:32 PM

While the final encryption of the encryption action, do find your method and input to see it is Byte Array. So the input whatever. You want to encrypt is given as a byte stream. So the result would be also this. The encryption result. What the doFinal returns you is a Byte Array. So byte array in byte array out. Those bytes are also binary, even if the text file is not. It will become numbers. Not necessary you can get out a readable ciphre text.

We want to observe the result of your encryption; just to verify if it makes sense. The cipher text is no longer readable and we do want you to display it. The binary text is not printable. We want you to use base64 and encoding. Generate binary bytes to printable characters. We print out the base result of the base64 encoding. The output are the encodable string instead of the byte array. It is good that you learn about base64 encoding. We have a question to contrast it with the encoding. Just to get general things into printable ascii characters.

Just you can do it on screen. It is useful in emails in the old days. If you have a jpeg you want to send, you translate into base64 encoding and send the base64 picture encoded as your message body.

Decrypt is still exactly the same, you set it to decrypt mode. Versus the encrypt mode you just solved. We want you to apply to image file. The image file would be a common format that is called a bitmap. You know bitmap of course right?

It is a bitmap file. This is a common format. It is not mpeg. What's the big difference? Between a mpeg and a bitmap?
Jpeg is compressed. Bitmap just raw bits. Uncompressed. But makes it easier for your assignment. Every bit map uncompressed. Is made out of pixels for example in this case it is 4 by 5. What is in every pixel in a bitmap file? RGB.

Opaqueness is also there. Each one is basically one byte. It is just a raw pixel array. This is where the data type conversion is, every pixel is represented by 4 bytes. In the trial I drew on the board, it is represented as a 2D. It is an integer array but we want you to encrypt it. The basic details that you need to do is given an integer array, transform it into a byte array, and takes the output as a byte string and convert it back to a 2D image file.

Remember that DES is a block cipher. If I have a secret kind of a byte stream to encode. Long one. I want to encrypt this long byte string given that this is a block cipher. Chop up the byte stream into units of a block. Say that this is the first block and so on. Who never remembers what is the block size? 64 bits. So 8 bytes.

If byte stream is just longer, just chop it up into units of 64 bits and encode sequentially. If plain text is a multiple of 64 bits, pad it up with some extra bytes, artificially generated by you. Make it by 64 bits. Rather than just a simple idea, gives rise to more detailed questions.

How do you do the padding? Can just use 0 as a padding bytes. People have designed more sophisticated coding methods. If you want to know what this is about. If you ignore this this is fine. There is no educational value. Just one specific method so that the plain text is a multiple of 64 bits. The other detail is what you want to compare: there is a ecb mode, cbc mode for you to encrypt this long plain text block by block.

Identical input, the output will also be identical for ECB. The same 64 bits would be transformed, but as

long as input is the same, output is the same. Not satisfactory in practice. We have Cyber Block Chaining(CBC). The idea is simple

You have this big string, chop it up into individual blocks. You encode the first block and there would be a cipher text. You use the encoding result as input into the encryption for the cipher block. You have a 64 bit plain text, you encrypt it to get an encryption result. You use that result as input to encode the second block. And then you use the result of the second block to help you encrypt the third block. This is called CBC mode.

Set it to PKCS5 encoding, just add it as another attribute for your cipher object.

Now this shows the conversion from the 2D integer array to the byte array. It can be used as input into the cipher function. Just copy this is the source of the copy. This is the designation of the copy. This is the offset of the source, this is the destination source. 4 bytes at a time because 4 bytes respond to each pixel. We are doing the conversion one pixel ata time. Concatenate it all to get a byte array.

This is the other way of the conversion. The reason why is so that you see the point we want you to see. What is the way we require you to do? Given a 2D image shown here. You will get columns of pixels. Each one is basically one column of pixels. Now we want you to encrypt this image column by column, from left to right. Treat the first column as the plain text. Using both ECB and CBC mode. After that you are doing as one column.

Put that column into your encrypted image. You expect it to be something like noise. Do the same thing and column by column you transform the plain text image to the encrypted image. We want you to display the image and see what you get. What is the scope of the chaining?

Does it apply within a column? If you use cbc to encrypt this column. You use cbc to encrypt this column. Is it applied to this column? No, because you are using cbc. It is block chaining. So a column as you can imagine. Block size is just 8 bytes. So if you have I don't know there would be many blocks already. You are finding the chaining. Use the result to inform the result of the second block. No. Because we specifically ask you not to. Do the encryption column by column. We limit your scope. Within column you do the chaining but across columns we give you different patterns. So just look at example code and then write it yourself.

So the question we will ask you why you see a certain image. Some image in spite of certain encryption, you get the idea of what the original image is. Not really the original image, but you get an idea what it is. ECB, is accurate, but CBC does not solve the problem. We want you to relate it to how you should do the encryption.

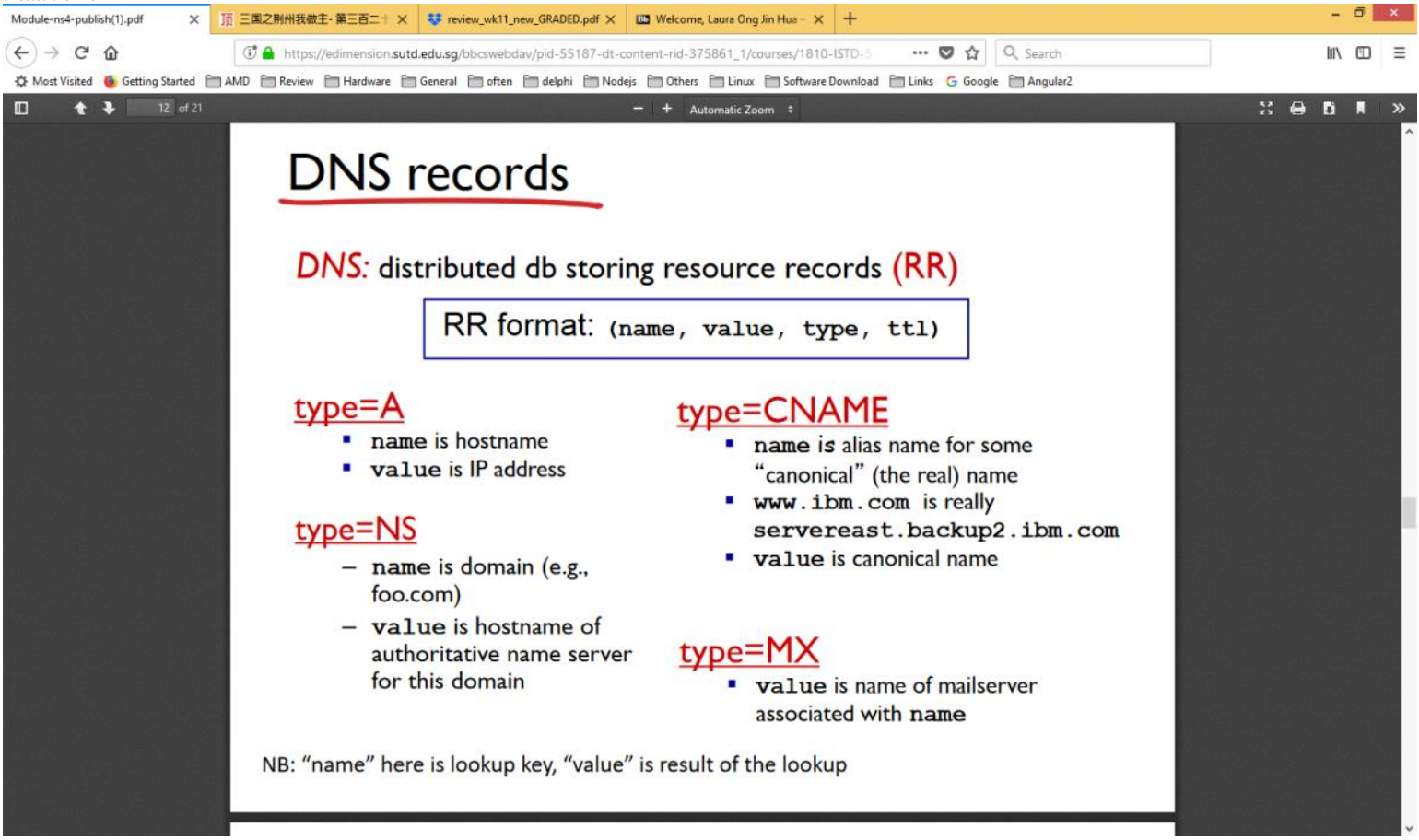
We ask you to do MD5, but if you are interested in SHA and other schemes, Just input it differently. So byte array in and out. We call this the digest. For sign and message you have to do RSA. It is a common t size that we use in message. RSA is a public key encryption system, it has a keypair. We use key pair. Given the keypair we can extract the two components. The way you use is the same as before. Set the padding, use ecb mode. And we do the encrypt.

And of course you can decrypt the result to make sure that you get what you expect to get. Same thing, you show arrangement, due one week from now and provide a document. Get some conceptual questions rather htan just output of your code.

Monday : Internet Naming and Addressing

Monday, April 2, 2018 8:34 AM

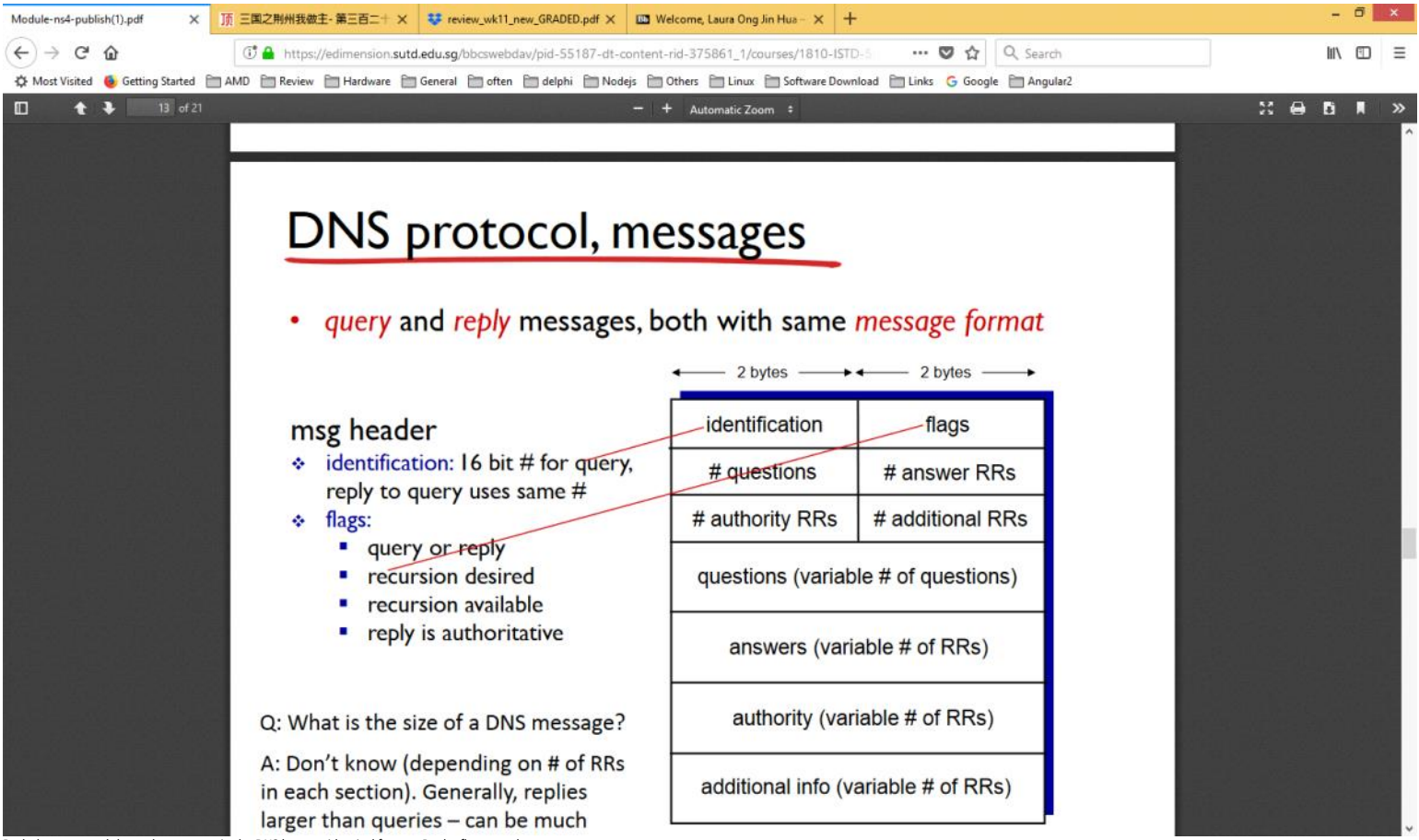
It is a hierarchy of servers (DNS). And even though it is at the application layer, it's not like the software. We use this all the time, but we don't use DNS. Even though we just talk about DNS, we don't use on a day to day basis. It is via the other software such as FTP, telnet, which make references to the DNS by means of APIs. There is an API that sends out requests and receives responses. There are certain tools, especially those who do network administration. This would be the topic for your lab exercise this week. How to use those tools to carry the domain name system and get the information. We talk about potential attacks on the domain name system. Some of them have been quite severe, and how they affected the DNS.



Let's move on to...Has these four fields. This is a record that resides in the name server. The name is actually: a host name, could be a domain name. For example, csail .mit .edu. This csail is a host name while mit.edu is the domain name. That means you are looking for an address/record. If mit.edu, then you are looking for a name server for that particular domain. Cname is a canonical name. This means in a name server (name server holds collection of records) some of those records of type a or type s. And then some of them could be cname. This means that this record is an alias name for some canonical name which is a real name. for example, www.ibm.com is an alias name for servereast.backup2.ibm.com. We don't remember that name. Which is why we have this canonical name. Even SUTD would have a canonical name. You can imagine a university the size of SUTD has a number of servers. Some could be mail, web servers and each of them would have a host name. If you want to query the name server at SUTD or IBM, that's all you need to do. Somewhere in the domain name server, there would be a record saying this is the canonical name and the query would be redirected to the canonical name.

When you query the local DNS server, that would go to the root server, and would attend the address of a TLD DNS server, and you would send a query and you would get a response from that host with the address of the authoritative domain name server. Finally, the address of the host you are looking for. At umass.edu.

The way it works is whenever network administrator creates a new name, they must register. That's the means we enter new names into the new addresses. You have to go through a registry.
We have these four types of records, they can have a time to live. How and when can a record like this have a time to live. You have quite a few entries in the resource records here. All of those would have time to live fees. After a while, you want to clear it out. There is not much interest; they will just clear out the cache. The network admin would set half a day or even shorter and set that time; and after that time it would be purged from the cache.



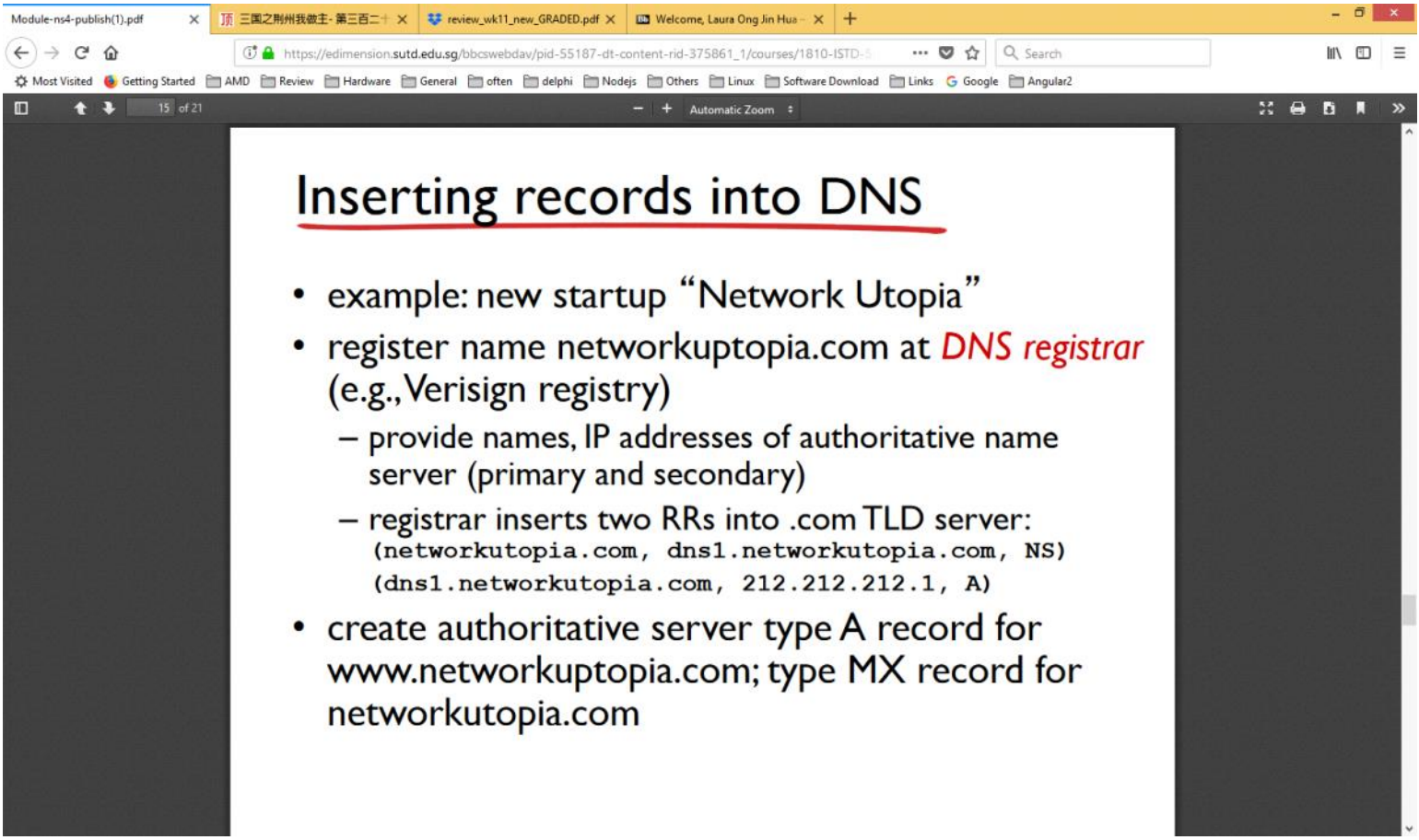
Both the query and the reply messages in the DNS have an identical format. By the flags over here, whether it is a query or a response. There are a number of flags. The identification here is a 2 by 16 bit identifier that allows the host to match the request with the response. For example, whenever you click on a URL in a HTTP browser, it would resolve that link. It would resolve some names that are inside. At any given time, one mouse click would yield about five or ten more requests. Or the requests will go out in parallel and will come back. With the incoming response.

That's the reason why we need an identifier field. Then we have flags. There are several flag bits over her. Query has zero replies. Recursion available. The last lecture we looked at how the requests can be served iteratively or recursively. Quite a number of requests are created by upper layers (resources of upper layers can be depleted when serving requests recursively). When both of these fields (the one who is sending a query desires recursion, server supports recursion, query can be supported in a recursive fashion). The last can be whether the reply is authoritative.

If for example we have a reply. Say first query came along and and it came from the authoritative DNS server. Say I am at this poly here and get a response (which is authoritative at local DNS server). Say someone else makes a request goes from another host, it would have resolved the name again. It would get a response from the local DNS server. Because it is cached. The authoritative bit would be set to negative. That's the reason for the authoritative bit.

Queries: queries come in all of these areas. Number of answer Resource Records. In response, I could have five responses. There are two additional sections. You don't see them that often You get an additional number of authoritative. Domain name servers. Perhaps you want to do load balancing. As you said, they are replicated. They are giving you all the replicated name servers. They are giving you additional information. There are more canonical names that you didn't ask for. You didn't see the first five fields in your DNS message.

What is the size of the DNS message? Can we say? The query can be quite small but the response message could be huge. They can craft the response in a way that is so huge and can come from botnets and attackers and can completely overwhelm either the server or the client. These are the questions, number of questions or answers. Number of records for authoritative servers. You get more than one authoritative servers for the same domain. Additional helpful information about aliases and stuff like that.



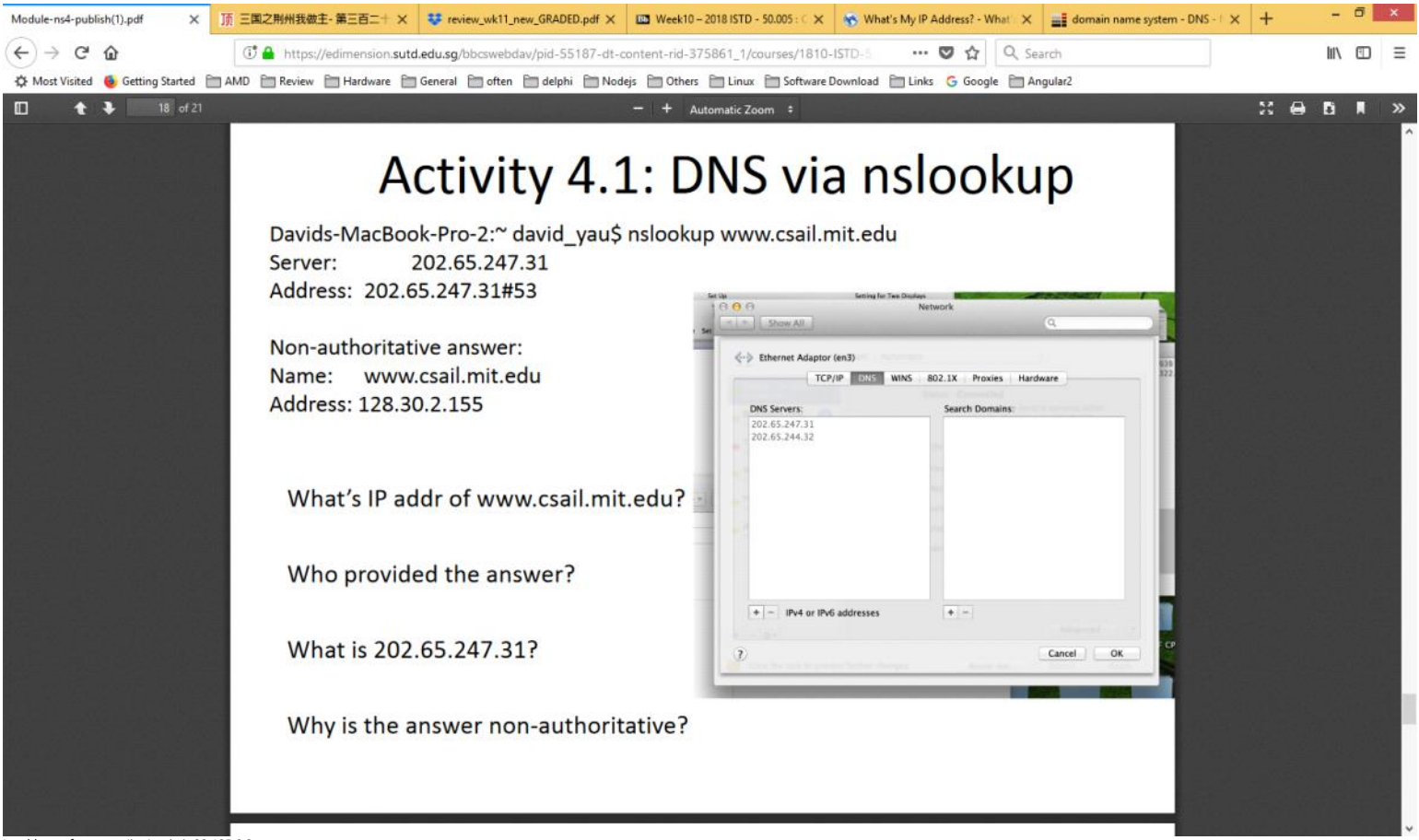
Structure of IP address has to be similar to phone. Because it has to be routed on the physical layer with real routers. Just like the telephone switching network; the landline phone needs to change because the landline phone is dependent on a real set of numbers. Same here, when we move from dover to here in Changi, it would be routed through a different set of routers, so IP address would change. How to change for all the hosts in SUTD. It has to go through a registry. A registration process. You need to apply to a DNS registar.

Used to be one through 1999. The data get so big that you divide to several companies that act as registars and pay a small fee to register domain name and your hosts. You have to apply to registar closest to you and they will register your IP address in the DNS system.

How many records will they register in the DNS server. One for the domain, and of course for the host it would be separate. You have to have an address and a record that maps domain name server to the IP address. You need to have the name server records.

The name of the host that acts as my local name server. But this is not an address type resource network. It just resolves to another name. I would have an address type resource record of the A type so that I would get an IP address to which my queries can be redirected.

In this case, you need a type MX record for network utopia .com. Often you can have the same alias name that gets you an mx record or an ordinary NS record. Application needs the mail servers record. The application would look for an NS record. If a web server, the server looks for the domain network utopia, gets the name server, and then be redirected to the webserver. In general, every time a new company registers with the registar. They would need to get at least these three. Registrations of the name server, IP address and registration of names.



Ip address of www.csail.mit.edu is 23.185.0.3
Provided by fe3.edge.pantheon.io
The local DNS of David yau's macBook
Answer is non authoritative as it does not come from a name server that is considered authoritative for the domain which it is returning a record for. (In other words, it is a name server that gets their answer second hand, just relaying the information along from somewhere else. It's been cached somewhere else.

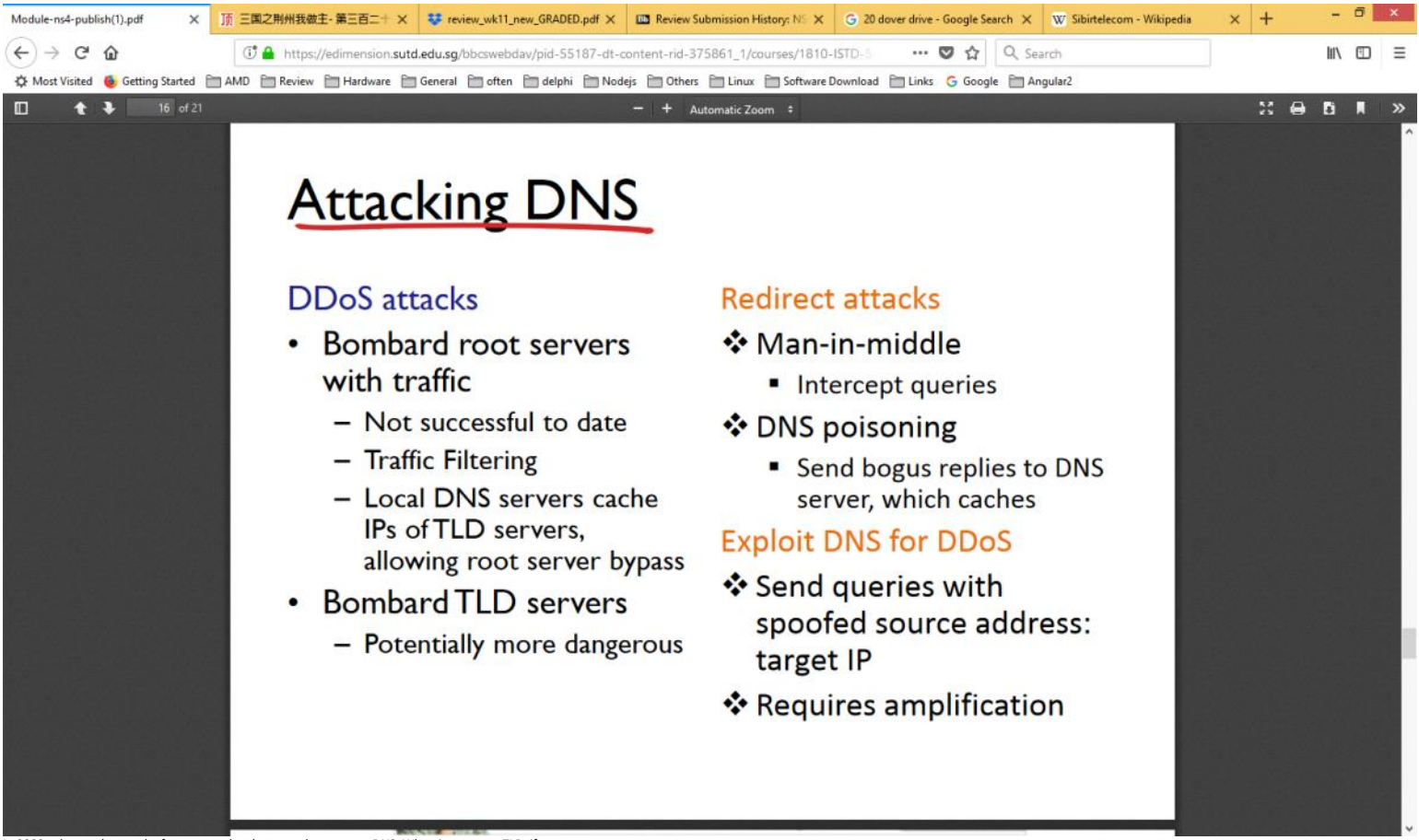
Email servers should use dmz-mailsec-scanner-(1-8).mit.edu
8 answers. Why useful to get? Redundancy for load balancing, for efficiency
You get the IP address of 18.7.68.35
United States Cambridge Massachusetts Institute Of Technology

We get these two sections here: other authoritative domain name servers. There are several of them
There is an additional section that points to an additional mail server that is not part of the answers.
That's the extra information

What is the DNS type of the extra records? Well, this one (authority section) are all NS records. While Additional section is an A type RR.
Local DNS server reports that it is at 18.9.25.14
Obviously, this means that you always address your local server by default. Do a dig on that domain name.

```
ongajong@ubuntu16: ~
ongajong@ubuntu16:~$ dig @asia2.akam.net dmz-mailsec-scanner-3.mit.edu

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @asia2.akam.net dmz-mailsec-scanner-3.mit.edu
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21644
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;dmz-mailsec-scanner-3.mit.edu. IN      A

;; ANSWER SECTION:
dmz-mailsec-scanner-3.mit.edu. 1800 IN  A       18.9.25.14

;; Query time: 232 msec
;; SERVER: 95.101.36.64#53(95.101.36.64)
;; WHEN: Mon Apr 02 09:41:43 SGT 2018
;; MSG SIZE  rcvd: 74

ongajong@ubuntu16:~$
```

The two answers agree.
Address if asia2.akam.net is at 95.191.36.64
Owner is a Russian Telcom: Russian Federaction Tomsk Ojsk Sibirtelecom
According to wikipedia, Sibirtelecom is a regional telecom and ISP in Siberia.



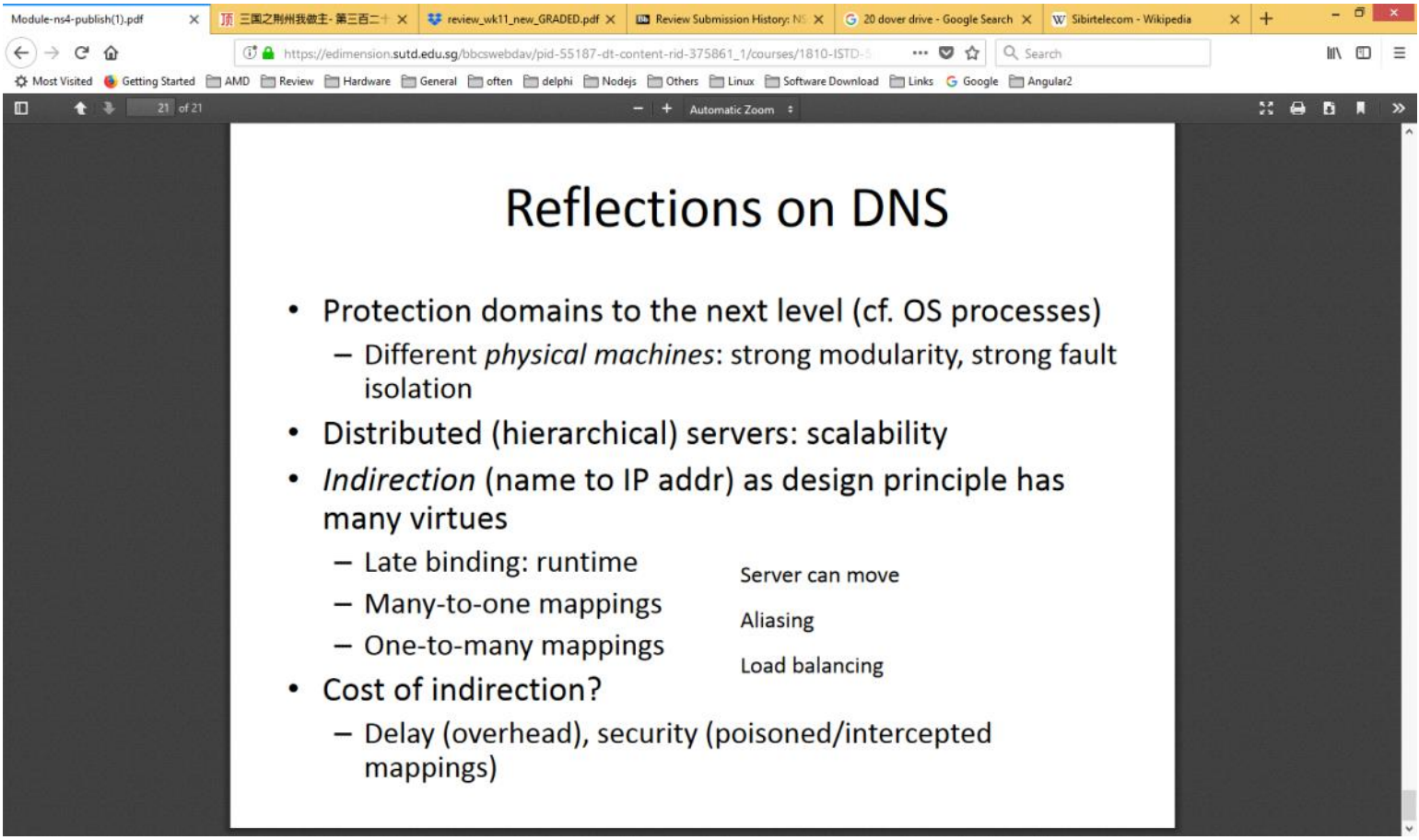
In 2002, a botnet (network of many attackers) sent packets to root DNS. When it comes to TLD, if you have a Ddos attack, with the same tactic. That is more difficult to detect with filters and so on. They were still able to curtail those attacks. The more insidious attacks are the ones directed through a man in the middle Like you saw, the man in the middle attack takes an IP packet and changes the source or destination. Didn't need to direct traffic to itself. It sends some bogus entries and did something that increase size of reply packets.

If what you expect is 1 response and amplify that to a hundred responses. It went from one mail server to another. Somewhere here, a poof entry has been injected. Even at this point, any spoofed entry which is now placed in the cache here will affect any future requests, and those requests will sit in the caches of domain name servers below that. That's what we mean by cache poisoning. Those are more difficult to get rid of because we don't known which entries are erroneous.

In the last one, it deploys the DNS for DDOS. With the spoofed queries source address. Say I am very unhappy with xyz.com. I have requested mail servers, I want all information to go with the spoofed IP address. It's getting from all over. For one request, there are many replies. This is exactly what this attack. It's a little bit harder because it tries a lot of tricks but can be targeted to specific target companies. The first three attacks are targeted at the upper layers and spoil the entire system. Whereas the last one is an attack for a particular host/company.

How can I try to authenticate someone trying to spoof my address. I have to advertise, the website has to be freely accessible. I can't have encryption/authentication. It is a fundamental problem to solve DDOS.

One click on the web browser we get a response. But there could be 8-9 web drivers. So it has to be scalable. Indirection (useful to machine architecture, memory mapping and use for searching domain name server on the internet.()


Module-ns4-publish(1).pdf
review_wk11_new_GRADED.pdf
Review Submission History: NS
20 dover drive - Google Search
Sibirtelecom - Wikipedia
https://edimension.sutd.edu.sg/bbcswebdav/pid-55187-dt-content-rid-375861_1/courses/1810-ISTD-5
Search
Most Visited
Getting Started
AMD
Review
Hardware
General
often
delphi
Nodejs
Others
Linux
Software Download
Links
Google
Angular2
21 of 21
Automatic Zoom
Reflections on DNS
Protection domains to the next level (cf. OS processes)
Different physical machines: strong modularity, strong fault isolation
Distributed (hierarchical) servers: scalability
Indirection (name to IP addr) as design principle has many virtues
Late binding: runtime
Many-to-one mappings
One-to-many mappings
Server can move
Aliasing
Load balancing
Cost of indirection?
Delay (overhead), security (poisoned/intercepted mappings)

Answers to Midterm

Wednesday, April 4, 2018 1:48 PM

When a user program tries to access unallocated memory, OS kernel will get control

True

Disabling interrupts should be privileged instruction in a multi-user OS True.

A computer system begins execution in user mode immediately after it boots up False

When a CPU cache miss occurs in a computer system, the OS kernel will get control to handle the miss False

A microkernel (Mach) provides better protection between OS subsystem traditional kernel architecture True

Context switching between two threads in the same process shas been about the same overhead as that between two separate processes False

Without kernel support, when a user level thread in process P blocks inside a system call P's thread scheduler will generally switch execution to another unable thread (False

An unsafe state according to Banker's algorithm means that the system will become deadlocked at some point int its execution. False

The Banker's algorithm has time complexity O(n^2 x m where n is the number of processes and m is the number of resource type True

In Unix, a process can update its file descriptor table without using system calls False

Ubuntu Linux allows cycles in the file directory structure through hard links False

In Ubuntu Linux, it is possible for symbolic link to become a dangling pointer (link refers to file that no longer exists) True

If result of a computation depends on the relative speeds of two concurrent processes executing it.

Race condition

A re-entrant binary lock can be acquired again before first process released, provided that the process acquiring it again is the process

Holding the Lock

Printf statement in the following code wl execute only if the exec lp fails

Q's value is 1 because I = 1,

Yes because I will implement i++ in the for loop but termination will go through one more time. The answer is yes and then 1.

If you take this logic, then you apply this logic, the first child process will fork one less than parent, the second less than first and then you get a tree.

P1 -P2 -P5 -P8

Public synchronized void insert(Work item)

If (count – Buffer_Size()

tryP

waitP[

By using a synchronized method

If you have multiple producers and consumers, they will be waiting for different logical reasons. A consumner waiting for a full slot. Although you have two logical conditions, you will be overloading that condition variable

Both producers and consumers can be blocked at wait() at the same time. Some processes may be waked up for the wrong reason. So the solution is after you wake up, don't assume condition you are waiting for is not true. If you use while it will be okay.

Sempahore mutex = new Sempahore(1)

Sempahore full = new Sempahore(0)

Semaphore empty = new Semaphore(N)

Full.acquire()

Mutex.acquire()

Item = buffer[out[

Out = out+1 % N

Mutex.release()

Empty.release();

Return item

I runs and enters CS (still inside it)

J starts to run and preempts i: gets stuck at inside while loop trying to enter CS

J wants to enter but it keeps busy waiting forever since it has higher CPU. [accept alternative solutions and give partial credit where suitable. Still accept answer if they think that I has higher CPU priority instead]

b) The non zero process assumption is not true. The static priorirty CPU scheduling cause I to starve and get no progress. (key assumption that you used in your proof does not hold for the above system.)

No.[1pt] P4, p2, p1,p3 is a possible termination sequence of all the processes [2pts]

Nothing – the system may or may not become deadlocked in the future

What can you infer about future deadlocks. Future deadlocks may or may not appear in the system.

a) You”can”
b) You”ca
c) “a”few
d) Wor

Open(“whatever.txt”

Close(0);

Close(1);

Close(20);

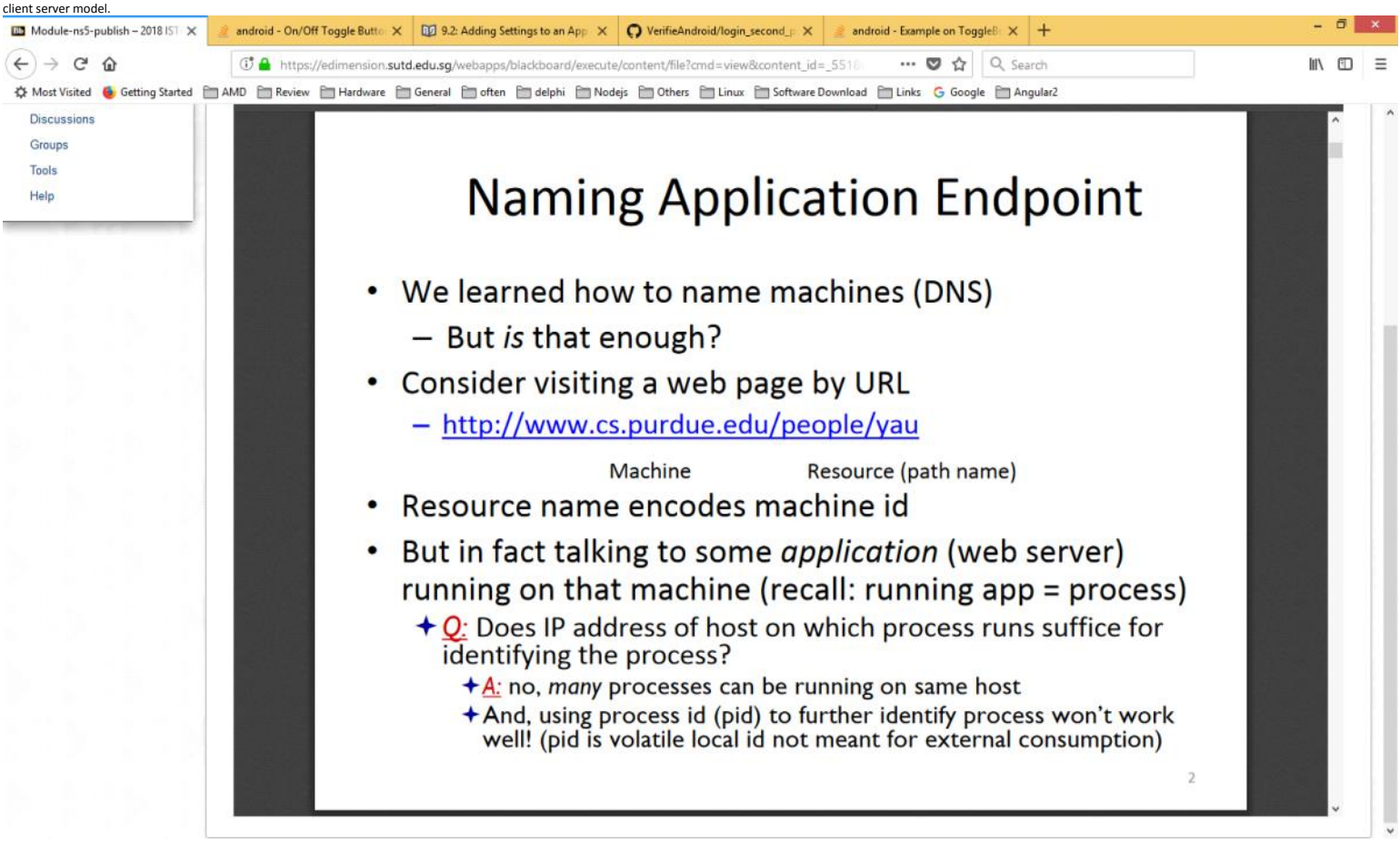
[the open must be before the close of 0 and close of 1 – deduct 1.5 pts for violating this condition; otherwise, these conditions can be reordered. Give partial credit if the sequence of operations results ina partially correct fd table for B, or contains extraneous operations that aren’t needed.

Thursday

Thursday, April 5, 2018 11:35 AM

Predominantly, the majority of applications out there are using this paradigm of clients and servers. The evolution of computer networking took place bottom up so we have to figure out how to do the connections, how to make it such that communication is possible from one end to another. The paradigm that was used was like a call. When you make a call, you had a handset. You picked up the handset. You had a signalling network. Only when the network was used could you start caling. It was built into the packet switch networks which we have been studying all along. Came this notion of client server and the web. We still have the same model of connection being established. Especially when you want to have reliability. So we will see we have something called TCP sockets. Transmission Control Protocol. It came from the days where systems were to go over transmission lines. You have to control the signal to have data communication. Then we have this very important abstraction of sockets. You talked about socket programming abstraction. At the application layer you can establish the communication end point to start communication from one end to another.

We will talk about that later on. You have sockets, ports. Sockets are communication end points. It consists of two things: a process ID and a port. And then we will talk about how the TCP layer does multiplexing and demultiplexing. We will cover the sockets, the port numbers multiplexing, go to the client server model, and if we have time, we will get to web servers and how web is an application of the client server model.



To briefly recap, we looked at the domain name system. Domain built at the application layer. It is used by application protocols at the application layer. FTP, whatever it is, supplies a name instead of an IP address, and you need to use the services of the domain name system.

This URL consists of two portions. One is a domain name, cs.purdue.edu. That is the name of that machine. It is www as well because that is the web server machine but could also mean an alias at that domain. And the rest of it here is a resource that is available on the web. It is two parts. A machine and a resource. The machine consists of domain name and a machine name. For example, gaia.domain name. You can actually look at your own PC and run your host name. Every machine would have a host name. End to End, it's more than just the machine name. Within that machine, you can have more then one application that are running. For example, on my laptop, I may have a web browser and a mailer. Mail client. So I can have a number of applications that are in the same machine. Does IP address of host on which process run suffice on identifying the process that I want to communicate with.

Let us stick to process 0 and 1. If I supply the process ID, would that be enough? They would need to identify server that is running on my host. Somebody wants to communicate to SUTD web server to communicate about various things like this. They need to know a well known port. As long as I address the web server at 80, then the connection request can be guided to the web server at SUTD. We need to advertise a well known port. Any reason why that process may not be a good idea. Processes are not always running. Processes can crash for some reason, but doesn't mean the communication should just disappear. You may be in the middle of a banking transaction. Maybe even a HTTP. If a process has disappeared, that doesn't mean the whole communication should disappear. Processes are phenomenal, you need something more stable. That is why we need a port.

You can have many processes on the same post. So adding the unique id to further identify process won't work because PID is volatile, and it is a local concept. When I reboot my machine, my web client, will get a different process ID. A socket is unique, a pair that has an IP address and a port number. Uniquely determines that socket. Another socket at the other end, you have uniquely identified a connection. A started out by saying that this came from a model of a telephone receiver. That kicks in a signaling network. A signaling network task is to introduce a virtual circuit from one end. When you have a TCP, a connection to set up from one ned to antother. You cannot have unreliabilty.

With UDP, you don't need virtual circuit. If packet makes it to the other side, that's fine but that's unreliable. When you want reliability, you need to have this connection. It is through a pair of sockets. What is a socket? It is a tuple, IP address, port number. It is a quadruple. A socket on one end and another, and that would be my connection. It is just built on top of this abstraction of socket.

The letters have to have address on them, and the door is a post box into which they pass all their letters, and the door is through the transport layer, controlled by OS kernel space. The delivery on the other hand decides how to take the incoming letters, and channel them to their respective, to the dorm addresses. That would be a good way of thinking about it.

You can think of a process as analogous to files.. You think of everything with one form of abstraction. Devices look like files, directories look like files and sockets are kind of file descriptors. So one way of uniformly doing things. It has an IP address and port number. We know the well known port number for web server which is 80, and mail server is 25. To send an HTTp message to this host at gaia.cs.umass.edu, you need to put the IP address as such and such and have port 80, and build a packet into this network. The machine there, at this host, this is esteemed to the web server that is running at that particular web number.

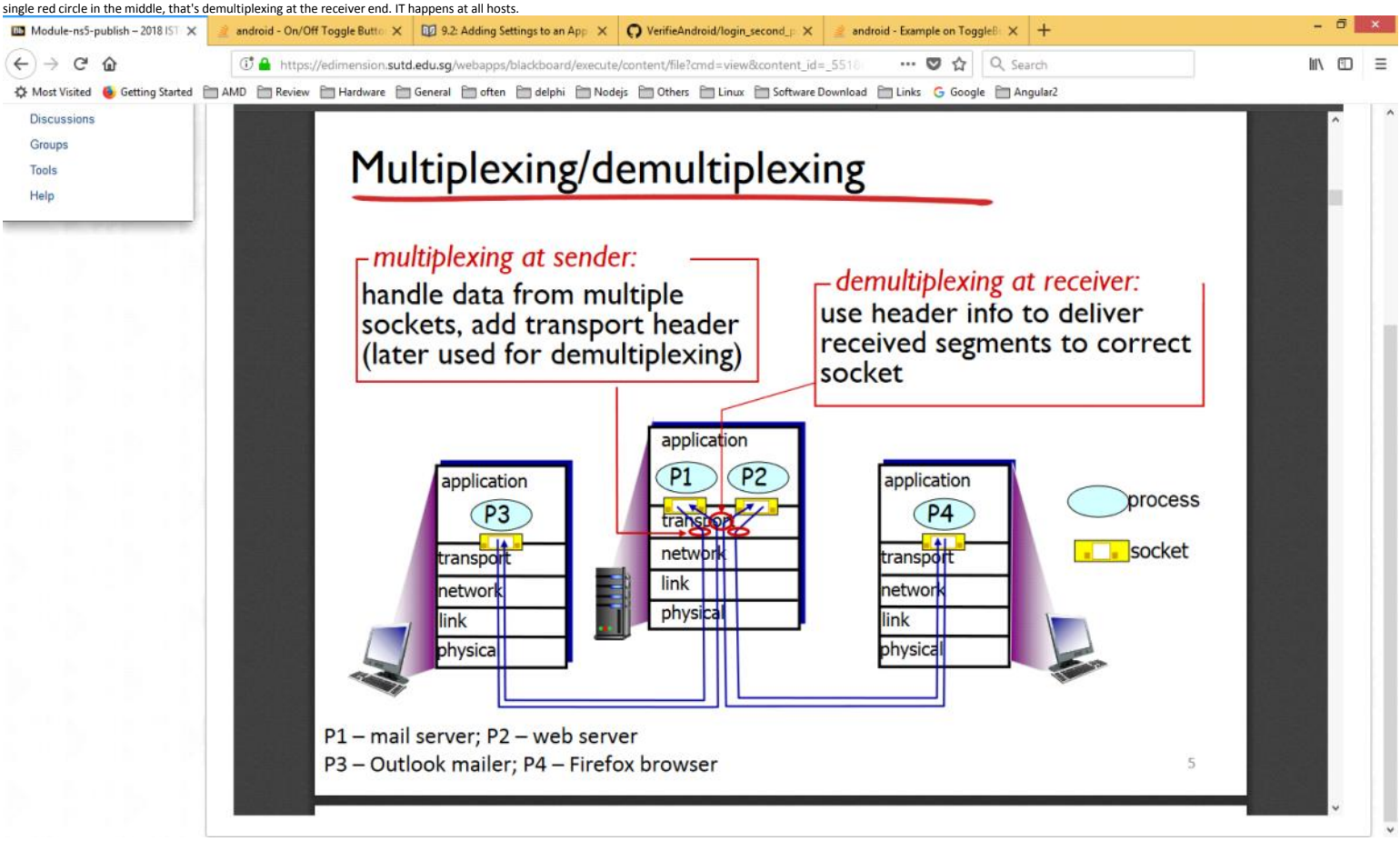
Alright, this brings us to this notion of multiplexing and demultiplexing. What are we saying here? When we have end points like this. This end point is on one laptop, and this laptop is different, and they are accessing this same server here which is a tower CPU. And your first application here could be a mail application where the server is here running the mail server on a process called P1. And P2 happens to be a web server on the same machine. P3 is the mail client, running on this host here, your laptop. The firefox browser is running on another laptop somewhere else. And you have 4 sockets, that are the end points at the communication set up.

What we mean by multiplexing, is that when have multiple doors going to the applications at this layer. The communication going beyond the door is going to the same protocol stack. Why do we need to maintain a difference when it is going through the same protocol stack and the same transmission/transceiver. The physical layer, it would go through the same wireless or ethernet transceiver. It is going to the same hardware device. They will be multiplexed, because they would go together. People leaving the dorms, heading into the same bus. They might be going to the airport, but in order to get out of the dorm, you need to get into the same bus. That can happen at the end point, even if the end points are clients. We are trying to show the multiplexing here. Can you tell me which flows can be multiplexing and which flows can be demultiplexed. Let us just restrict our attention to the server here. We are showing a flow by an arrow that goes out and then in. An arrow represents a flow. A sequence of TCP packets.

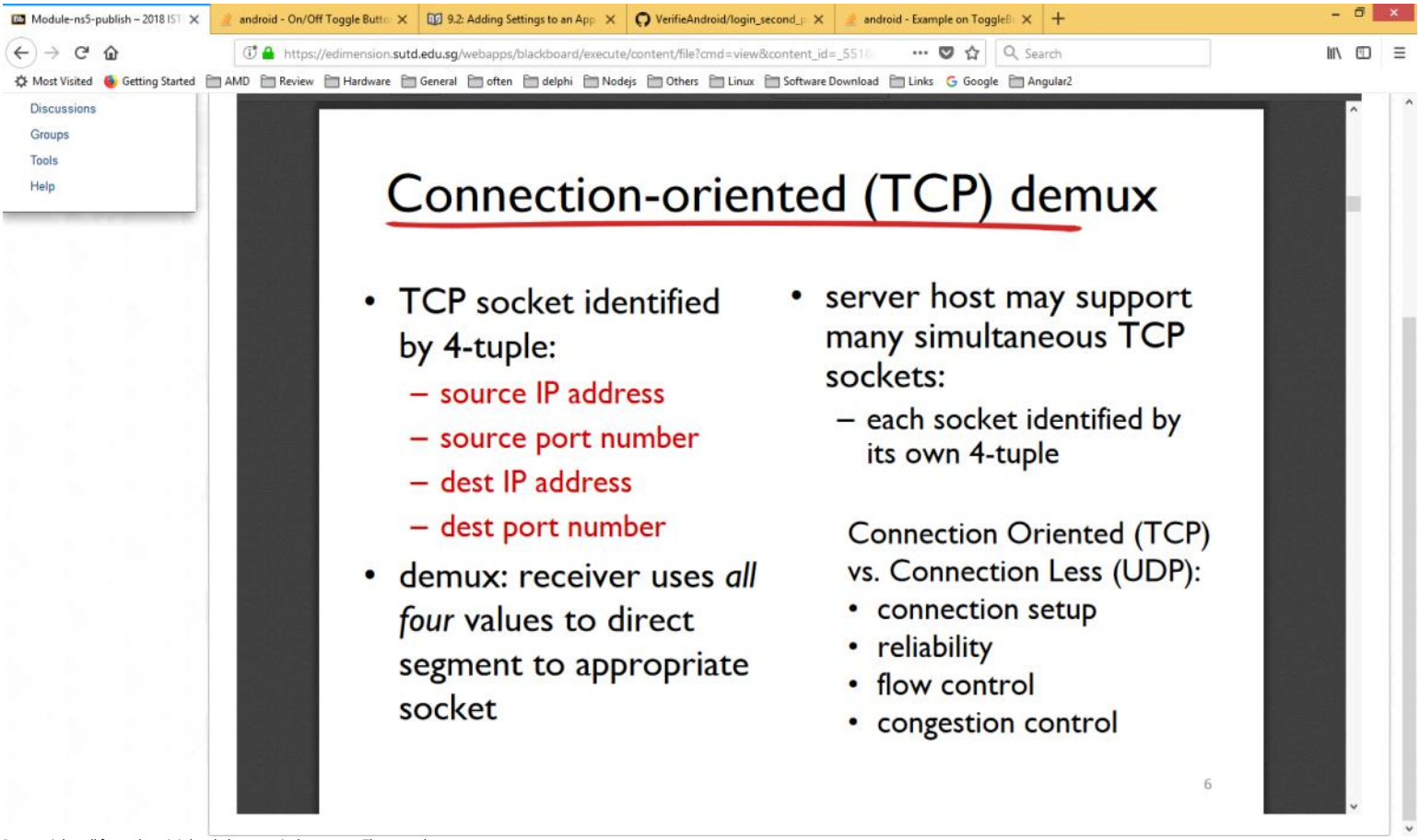
We are talking about TCP now because we are going via connection oriented traffic. There are two incoming flows from P3 and P4. where are we multiplexing and where are we demultiplexing.

Can you see there is a switch here between P1 and P2. That is demultiplexing. They are headed towards different processes. How would we know which processes they are going through? Through the port. We know the socket number here, and by looking at the different port numbers, we know they are heading to different processes. But they come through the same channel and get demultiplexed. The ones that are going are multiplexing, as they have to go to the transceiver, and one goes east and one goes west. It happens on every router, every switch and every host in the network.

You can see the two red circles are actually multiplexed at the center end because they are sent out. The single red circle in the middle, that's demultiplexing at the receiver end. IT happens at all hosts.



On an operating system which runs on a machine, you have many processes running over TCP and run over different type of web applications. One can be some kind of a new Skype. So you can have various applications running here but they are sharing the same transport layer, network layer. You don't have to maintain individually.



Because it has all four values, it is headed to a particular process. The server host may support many simultaneous TCP sockets. How does it do that? There is a web browser and the web browser is running a HTP service. Every client when it tries to request a URL or HTTP page, the client doesn't know anything other than a port number. Server needs to continue to listen on that. You can have many simultaneous TCP sockets on the server. For that web service application, you need to create a new socket for every incoming request. So what it says here is simultaneous TCP sockets can be for different applications. Could be skype, file transfer. Each socket identified by its own 4-tuple. What's the difference between TCP and UDP. TCP is transmission, it's unique feature is that it is reliable. What we mean by is that in the first lecture, we talked about the model of the packet as it goes. You can have packet loss here if a router is very busy and buffers are full. This is a router and the outgoing buffer somewhere here can be filled up. The packet is dropped. If the packet is dropped, now we have this end to end communication and the packet is dropped along the way.

The TCP would remember that through the use of counts and packets IDs. The packet IDs that are sent and acknowledgement that are received. Each packet would be acknowledged. Looking at acknowledgements received, then it guesses which packets have not been received/lost. You can retransmit a packet that the server doesn't see, but you are sure that each packet sent would have been received. That is how reliability afforded in TCP. The way it is achieved is through the use of packet IDs, identifiers.

What is the drawback of this model of communication. I have this big chunk of data, which is very important. Can I deliver the data with the missing byte here? I can't deliver it. This big chunk of data would release until the packet has been received. That would take time. See if computers have already come. It may develop out of sequence. You have to wait for a while that the packet hasn't come. I have to notify that the packet has not been sent, and not been received, I will retransmit. Only when chunk is received and acknowledged can I release to the upper layer. The problem associated with having a reliable connection is Delayed. The UDP transmits a packet and falls behind. All memory of that packets is not found in the lower layers.

UDP packets can be lost, and you have to deal with it yourself. There is also this phase of connection set up. There is no connection with UDP only for TCP. Connection is a port on either side, lets packet through then terminates.

Flow control has to with a slow receiver uttering a request to boss to send off. At theis powerful server at the other hand. The mobile phone cannot happen it. Congestion flow is done at the routers where you don't want to put in too much into the network.. Maybe there is no problem at the end point. The network is heavily congested. So it is congested even further. You need to know the state of the network..

Let's take another look at the Connection-oriented demux example. It is not UDP, so we have source, disturbances and IP addresses. Over the connection, this pcket has a source IP address of this host. And the source port number is 80 and this is a web server. This machine's IP address is this port here for Process P3. We have three incoming packets and one outgoing packets here.

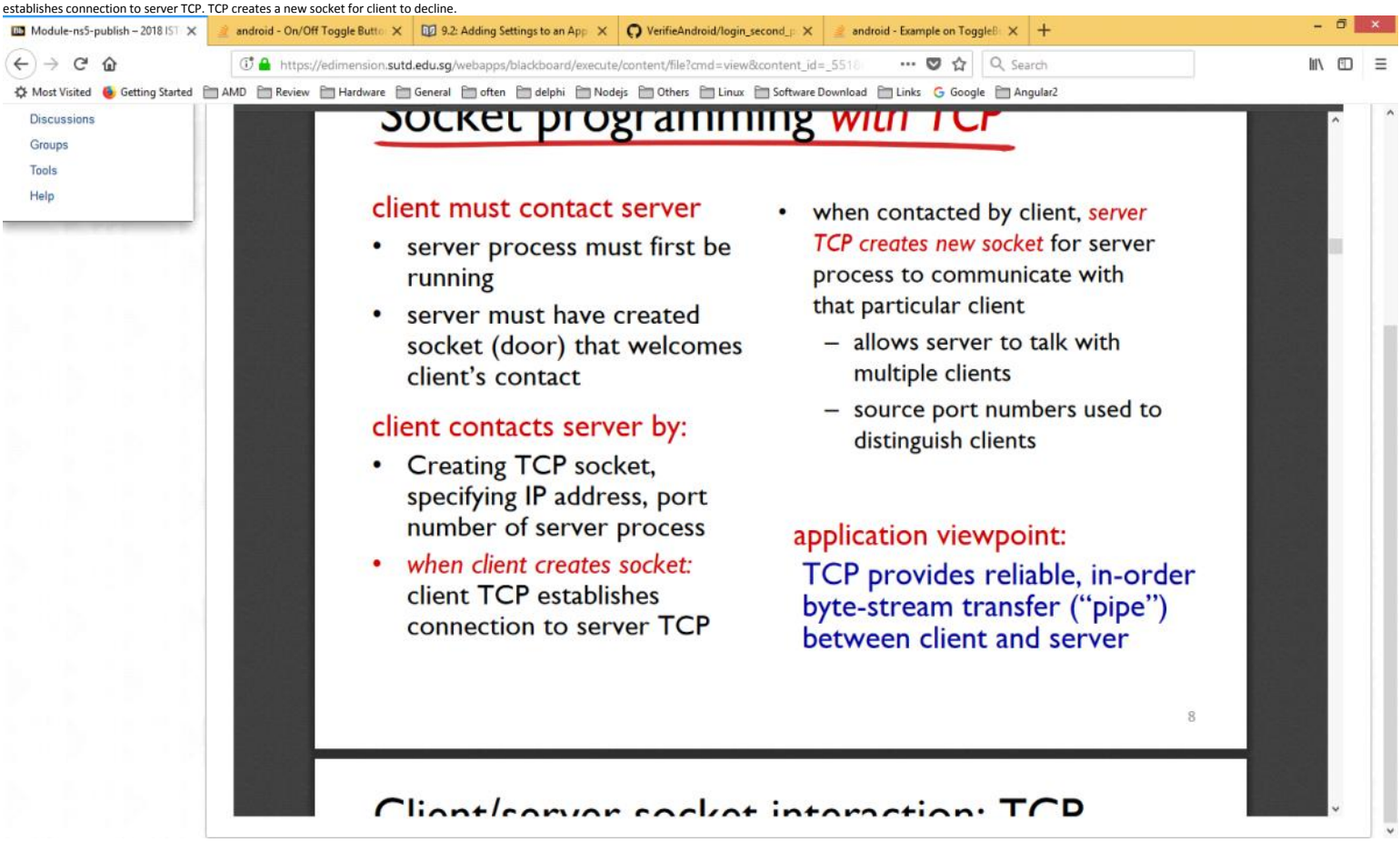
There are three packets that are destined to the same IP address. Destination port 80 are multiplexed to different sockets. They all have the same 80 destination port and they are addressed to the same IP address. But they are demultiplexed to 3 different processes.

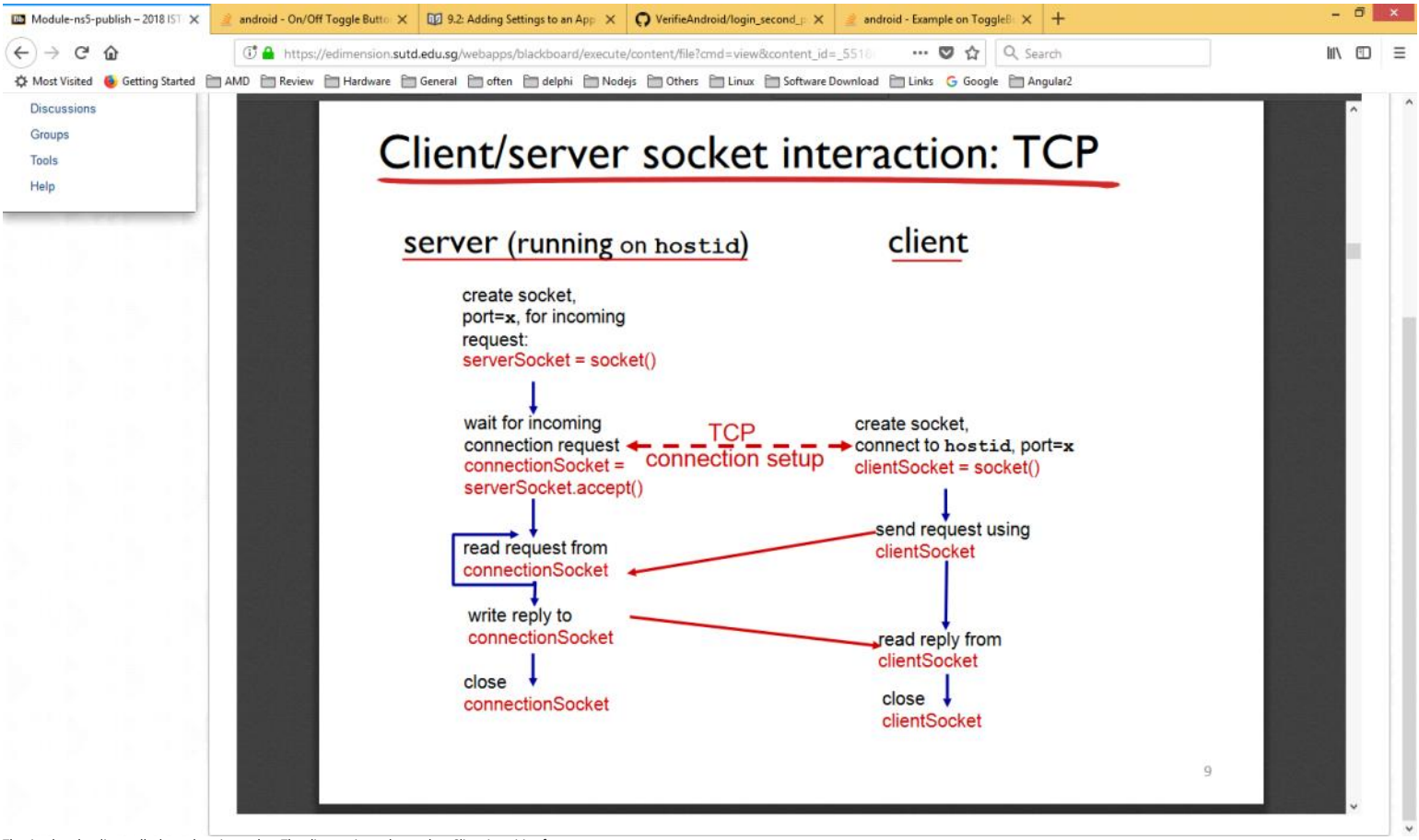
Remember the 4 tuple. The 4 tuples have to match. The destination port, IP address has to be matching. Any matching is a different flow, therefore it comes from different IP addresses, so they are characterising different packet flows or connections, and therefore should go to different sockets. Let's go the example of socket programming with TCP using client server model. We use the client server model because you see it in multiplexing.

First of all the client should contact the server. If the server doesn't run, then the client must be looking for something empty. So server process must first be running. Does the server know the client's port

number. Any IP address and the server doesn't know anything about the client. The server has an open door. It is also calling a welcoming port. 25 is an electronic port for email. The contacting SMTP client could be from any IP address. That is called a welcoming port.

The client contacts the server by creating a TCP socket. It is called a welcoming socket but the client has to create a socket and specifies its own IP address and port number of the server. Like they say Gaia, I want to access the web server at Gaia.umass.edu. So get the IP address, put prot number 80. And client establishes connection to server TCP. TCP creates a new socket for client to decline.



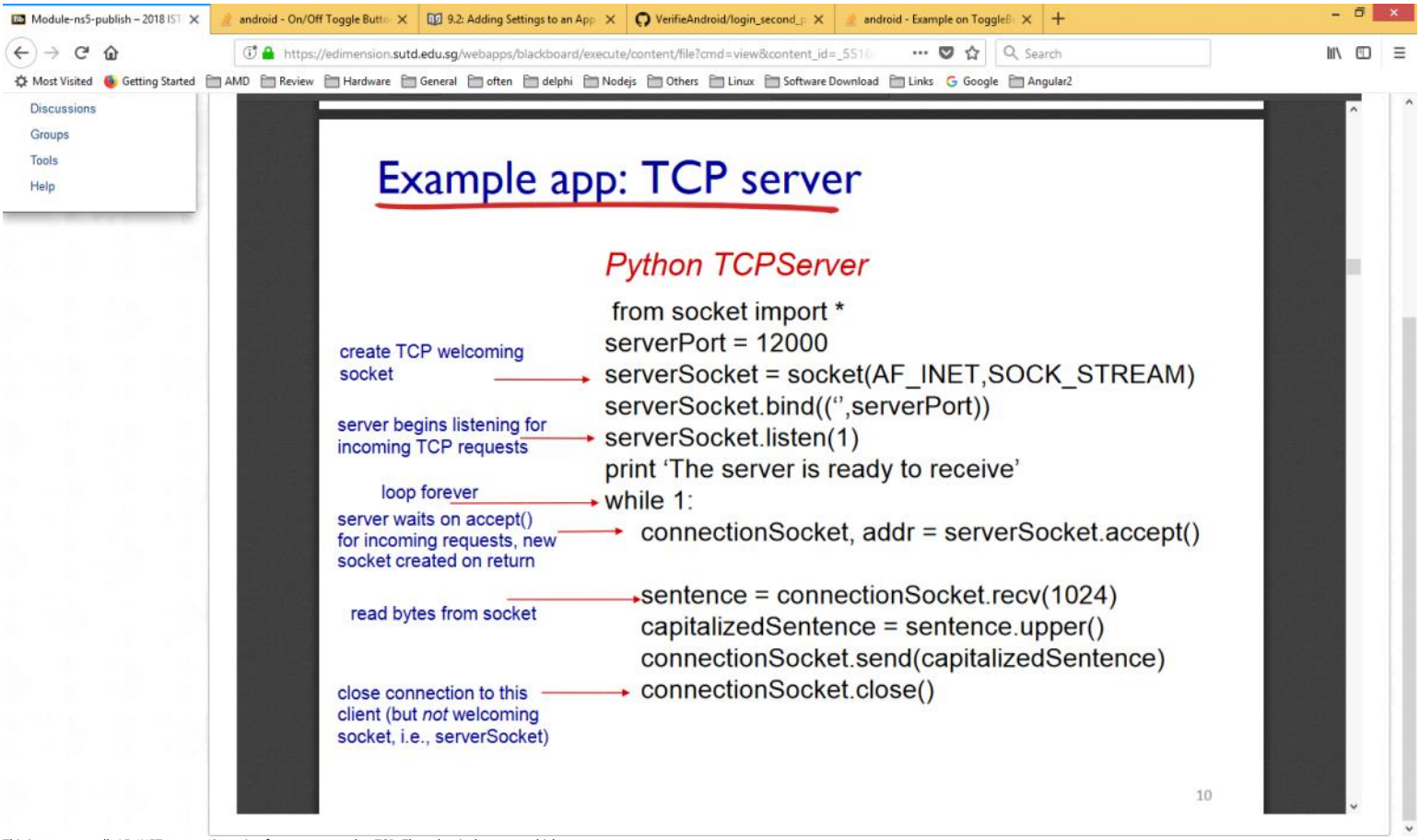


That is what the client calls the welcoming socket. The client waits at that socket. Client is waiting for people to make web server requests. It has to wait for the client to make the request. The operating system call is actually an accept. This is an object oriented programming, socket has a method called accept. Server will invoke that method to accept a client's request. This is called waiting on a welcoming socket.

On the other end, client creates a socket that it wants to connect. Server runs 24 hours, but client only on when it wants to send some mail. This will be done sporadically. Client would be operating through the same system call. It is a three way connection set up, and then a reverse acknowledgement. If you have a three way connection set up, you can see the packets using wireshark. You can see the connection going through in this set up. Data transfer cannot happen so the model is very similar to telephone network. You pick up a handset from source to destination and back. It is a switch network. You have signals going back and forth and only when the connection has been set up, I know that this is my source port, this is my destination IP address. At either end would the data transfer takes place. That's what happens next.

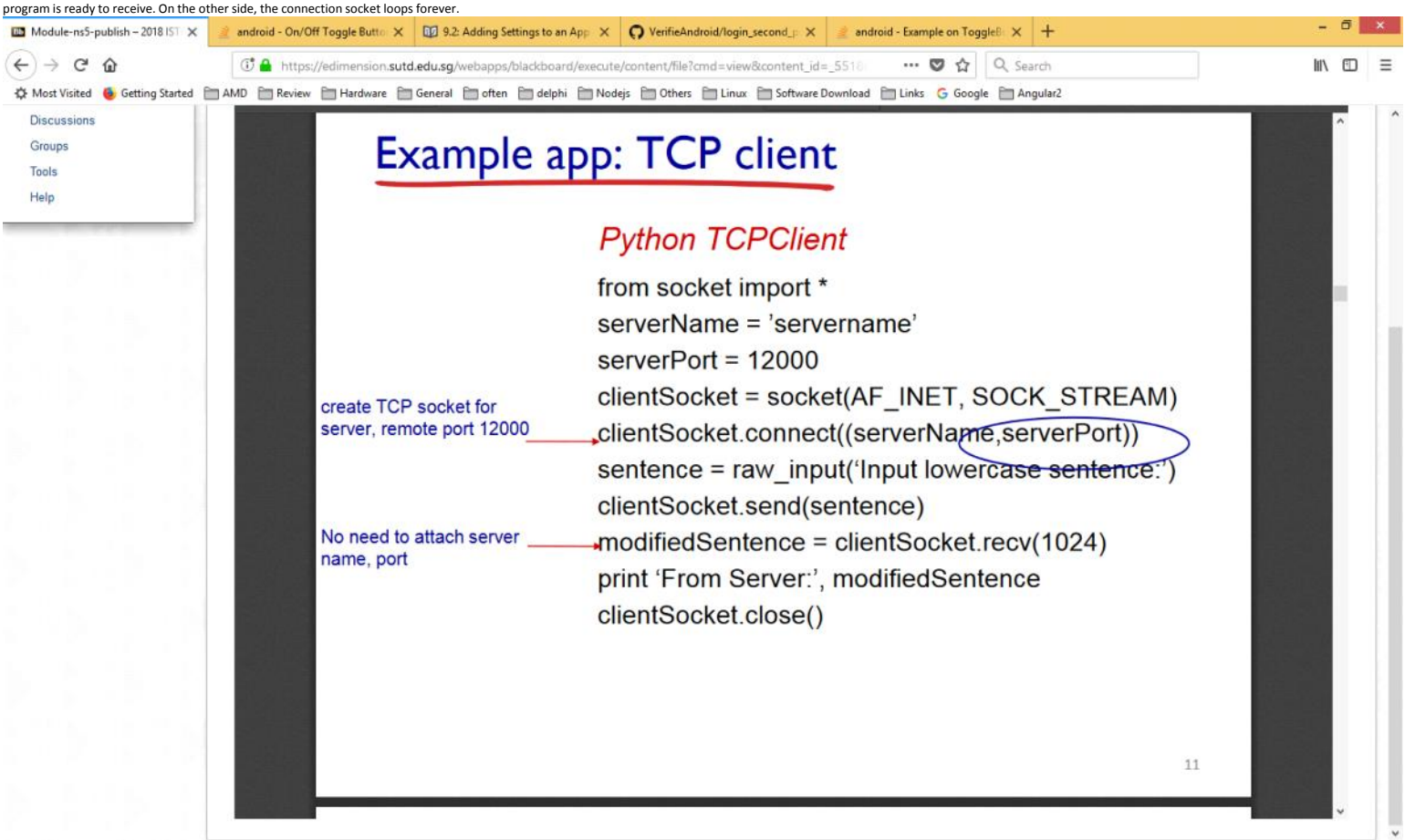
Client now sends a request and in the case of the server, the server is using a new socket. There is a subtle point here. The server is not using the same server socket. Why is that? It has to allow other clients to come in and connection. This server Socket, if used for the same connection, could it have been open for other clients? Web server has to have 100 of clients. Then you have the cloud and how this could be replicated. You can have many clients accessing one server and you need to use the same socket over and over and you need to create a new connection socket. And have this connection flow with this server. There is no reason for you to have one server and client. When you are done, the other end can close this communication. The way it happens, could be done at either end. It has to serve other requests.

This could be multithreaded. This server is multithreading.



This is a system call. AF_INET, means IP version four, stream socket TCP. The other is datagram which means UDP. This one is TCP. You have a binding phase where you bind to server port which you have created. So this is not that important.

The server does a listen. So the server begins listening for incoming TCP requests. At this point, the program is ready to receive. On the other side, the connection socket loops forever.



When client is ready to begin communicating, it has to supply a server name. The server name is the name of the host. Mapping from host to IP address which would be handled by the application layer. And a client creating a socket. Supplying the same information, server name, server port and once that finishes, we know that on the other side, the listening has accepted that socket. So the server is listening on that socket request.

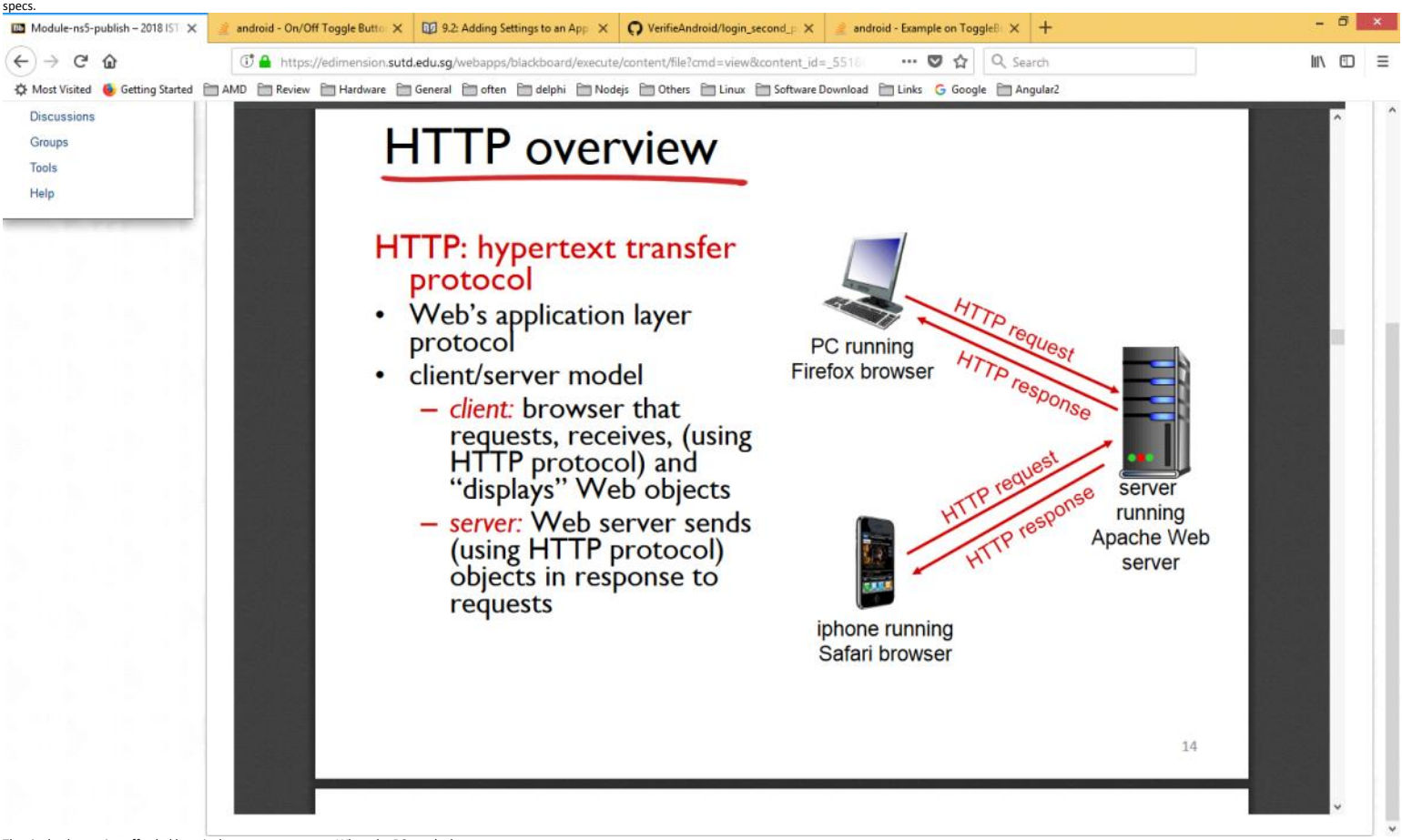
Once you go beyond this, the server has processed the request from the client. The server would then do the accept here which is matched with the client's socket communication here. The client can then start the client communication. Sentence = raw input, sending that sentence to the server end. Server will receive that, capitalize it and send to the clinet. There is a send followed by a receive here. A packet is sent here and then received. The server is doing just the reverse; it is receiving first and then sending.

Is there any notion of connection on UDP? All the UDP does is to send the packet and forget about it. There is no handshaking. Of course the sender needs to have destination IP address and port number. Otherwise, you would not know where the IP address is headed. As I said before, UDP packets get lost along the way as it is the application who would take care of it. From the application's point of view, you have unreliable data transfer.

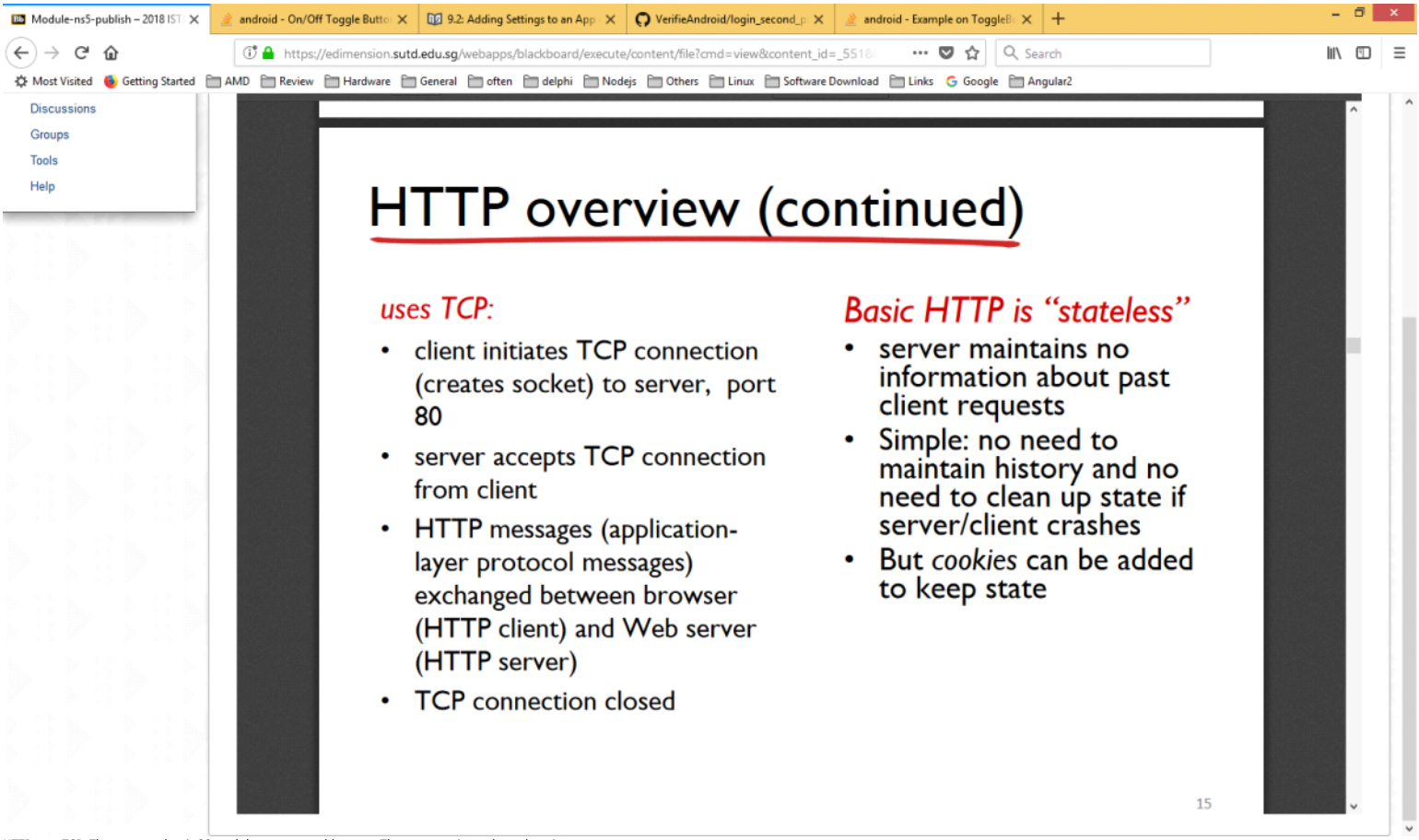
Now we come to the application of this software model in TCP. Web and HTTP are designed for use of TCP in this manner. We shall see that in the minute. What is a web page? You have a url. It's in text, its HTML pages are always in text. There are further URLs. You can have images, other web sites and various objects of various kinds. So a web page is basically an object which consists of many other objects. Some of which could be other pages. Binary objects such as video segments and so on,. Web page consists of a base HTML file and other reference objects. When I make requests, I'm basically requesting a server to send me that page.

There is a very good property that have the web page and the request response that makes it useful and dependable. Representational State transfer. REST. Why is REST so useful? There is no memory. You don't have to store any information anywhere in the network. I want a web page, the server, at this point in time, these are the contents of the web page, I will not take responsibility. It's the representation of the state. It is useful, dependable and easy to relate to.

What are the drawbacks of REST? You have online purchases made across the web? Anything that requires memory cannot be done. At least not in a simple way. Of course you can do everything through email. So well there are pros and cons and let's talk further about HTTP and the web. You have a host name and you have a path name. Every URL consists of a host name, the web server at that side and the path name. You have a host name, some department, and then a picture of the professor and whatever. So this is an overview of HTTP. You have a client running firefox browser. You have an Iphone running a Safari browser and a server running Apache Web server. This protocol has been in a standard's argument, and anyone can produce software that runs. That is freely available, and used as standard specs.



The single abstraction afforded here is the request response. When the PC needs the request, server response with a HTTP response. If this happens over UDP, will not be a good web experience. Can you use UDP for it? You will have a lot of packet losses. The image will not be quite what it should be and might lose out of some text here and there.



HTTP uses TCP. The port number is 80, and the server would accept. The server waits at the welcoming socket and accepts TCP connection from the client. Once the connection has been established, the server sends across the requested page. What does it mean that connection has been established? This term is a three-way handshake. And it consists of the client making the requests to the client. Server responds with synchro + response. Only when three steps has finished, would server send the request page.

Three way Handshake
- Client and then the server
- Client makes request to the server called Sync
- Server responses to the sync +ack
- Typically, this leads to acknowledged, but may send some data
- In the case of HTTp, no need to do that.
- Along comes your first web page.
- Transmission takes place.

So this part is called the three-way handshake.

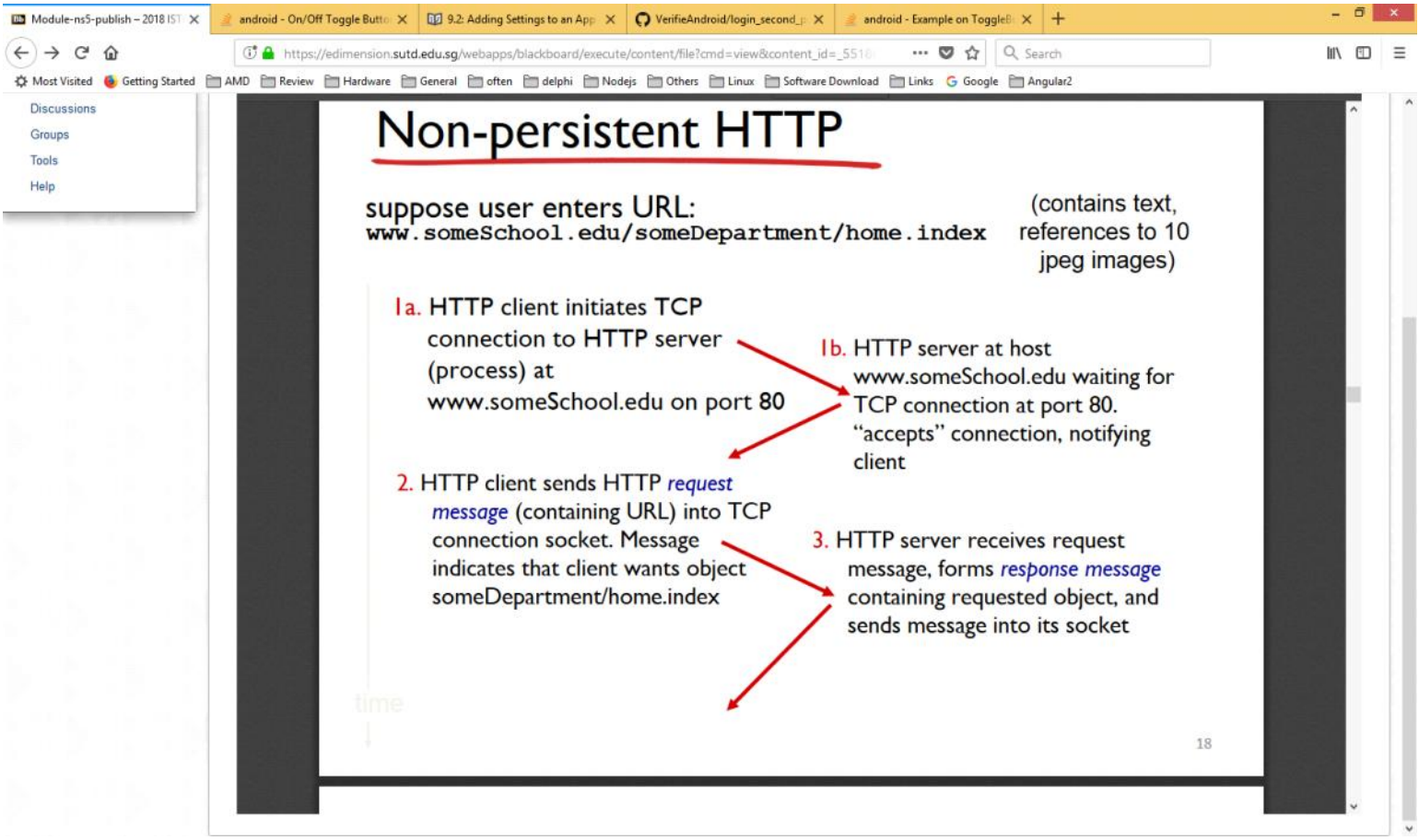
So once the connection has been established and data transmission takes place, that connection takes place. It is stateless, there is no state held in any state within the network about what has transpired. No need to maintain history. There is one way to bypass it. How can you buy your airline tickets or make your purchase books or something good.

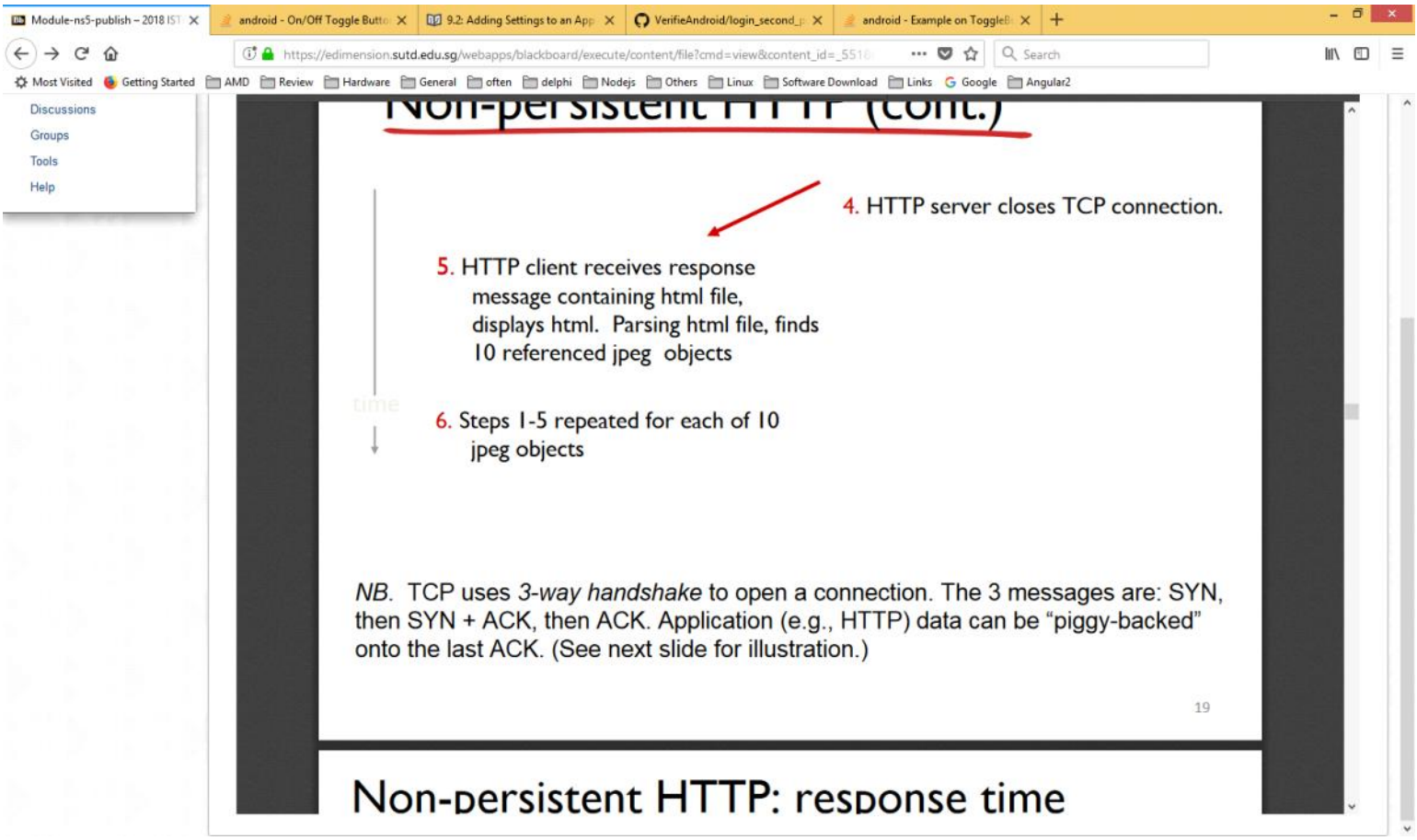
Because of cookies. Cookies is a piece of information stored at the client. Not at the server.

This brings us to two versions of HTTP protocol. HTTP 1.0 and HTTP 1.1. What's the difference?
When we had the first version of HTTP, this is all that was responsible (three way handshake)
Followed by transfer of data. Sync is over TCP, telling the server I want to sync with you. That's an archaic word.
So over TCP, it is closed. You have to create a new connections and download that information. Let's say there are multiple objects, information about various objects like university professors and you have to download each one of them to a separate TCP connection. The TCP connection would be closed and the difference here is because of a persistent HTTP. The connection between the server and the client is maintained for a number of object transfers. Multiple objects can be sent over a single TCP connection between client and server. It has been explicitly closed by client end. Then the connection can be closed. Until then the connection can be held open. Let's go to this particular website.

See this web page displayed in to browser and look at the HTML source. If it is outside of that web server, there are embedded objects within a web page that do not belong to that web site. Now we have this paradigm, we have to get the web page and then have to close the connection and then go to google to retrieve the object from another. And then render some more information on the web page. This can go on for a long time. So the moral of the story is that when we have HTTP 1.0, there is a lot of overhead because a lot of TCP connections held over a number of times. So have one connection and get multiple objects.

You have to have two separate TCP connections. IF the website is the same and multiple objects, you can download multiple objects one after the other and save from having to have multiple TCP requests. Suppose you enter a typical URL. Then enter 10 embedded JPEG images. A connection request is made to the server on Port 80. HTTP server at host Accepts connection and it acknowledges it to the client. And the client sends the HTTP request and HTTP server closes TCP. Each of those 10 Jpeg objects would need to be retrieved through the same protocol. With the download of data for each of those 10 objects, each url would incur all of this overhead over and over again. Each new website would incur a new TCP connection. Why incur it over and over again for the same website?


Module-ns5-publish – 2018 IST
android - On/Off Toggle Butto
9.2: Adding Settings to an App
VerifieAndroid/login_second_p
android - Example on ToggleBu
https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5518
Search
Most Visited
Getting Started
AMD
Review
Hardware
General
often
delphi
Nodejs
Others
Linux
Software Download
Links
Google
Angular2
Discussions
Groups
Tools
Help
Non-persistent HTTP
suppose user enters URL:
www.someSchool.edu/someDepartment/home.index
(contains text, references to 10 jpeg images)
1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client
2. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index
3. HTTP server receives request message, forms response message containing requested object, and sends message into its socket
time
18


Module-ns5-publish – 2018 IST
android - On/Off Toggle Butto
9.2: Adding Settings to an App
VerifieAndroid/login_second_p
android - Example on ToggleBu
https://edimension.sutd.edu.sg/webapps/blackboard/execute/content/file?cmd=view&content_id=_5518
Search
Most Visited
Getting Started
AMD
Review
Hardware
General
often
delphi
Nodejs
Others
Linux
Software Download
Links
Google
Angular2
Discussions
Groups
Tools
Help
Non-persistent HTTP (cont.)
4. HTTP server closes TCP connection.
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
time
6. Steps 1-5 repeated for each of 10 jpeg objects
NB. TCP uses 3-way handshake to open a connection. The 3 messages are: SYN, then SYN + ACK, then ACK. Application (e.g., HTTP) data can be "piggy-backed" onto the last ACK. (See next slide for illustration.)
19
Non-persistent HTTP: response time

Monday

Monday, April 9, 2018 8:37 AM

Version 1.4, where you establish the TCP connection through the three way handshake, and the connection is closed. The server closed the connection. You have 10 images from the previous example, you have about thirty or twenty images, for each image to do that, that would be a time taking process. It also makes the server overloaded. To just do a back fo the envelope calculation. Let's think of the server here. Let's have a simple model of a RTT.

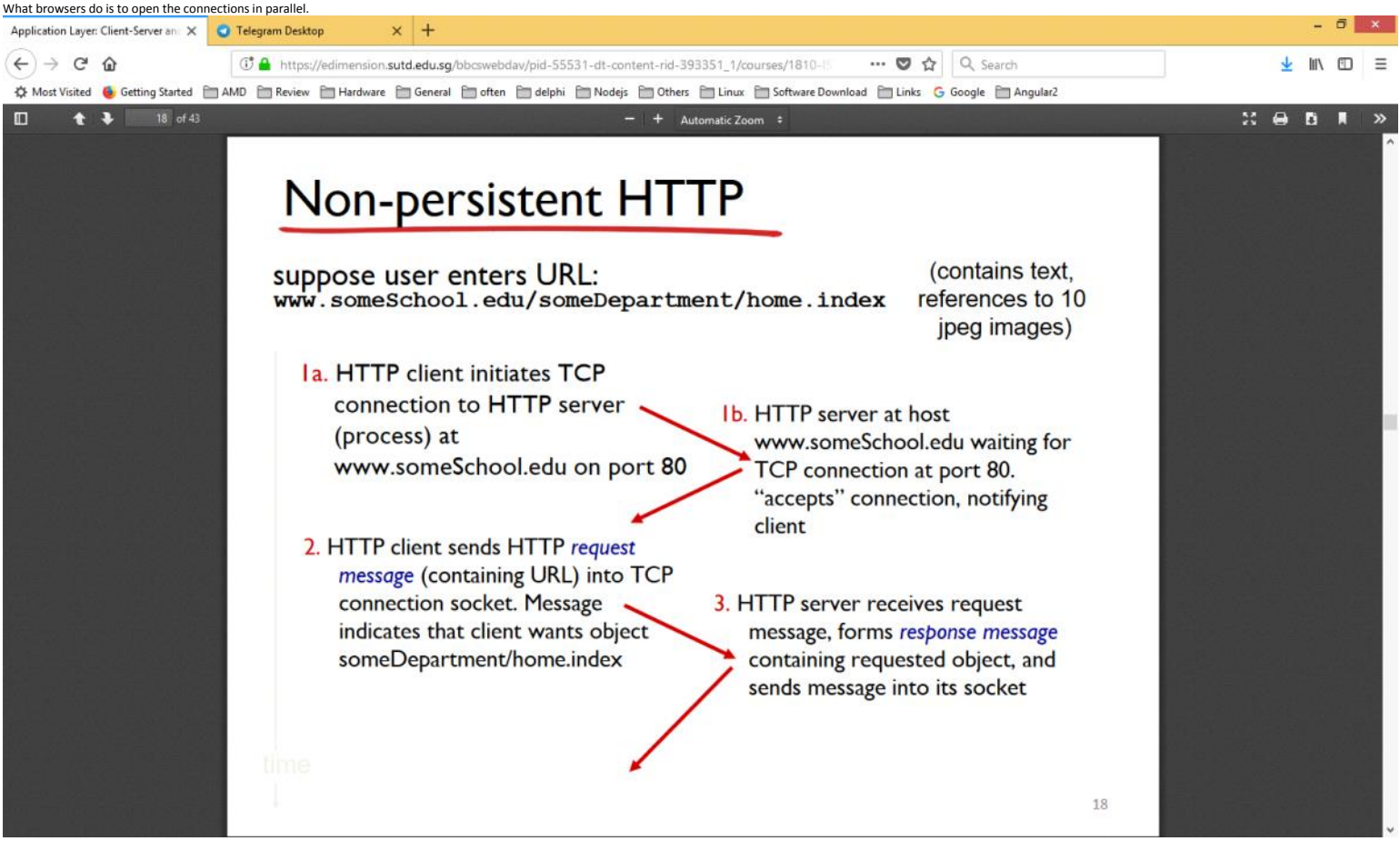
This model, we put together the four sources of delay: processing (which means each router, you have to figure out what's the outgoing port) queuing delay (congestion on the network, packets ahead of you.) transmission (determined by the transceiver bandwidth, 10MB or 1 MB whatever kind of transceiver you have) and propagation delay and we put it together to have the round trip time from forward to return journey.

Synchro + synchro + ack which is the response to that. Finally, what often happens is the client sends its requested web page. This third step is ack plus request. So that file is finally shifted over from the server to the client. If that's the case, what would the total time be. So total HTTP response time for non-persistent = 2RTT + file transmission time.

This is for the non persistent case. Each request is served. Most browsers do this. Sequential transfers of objects or HTML pages one after the other. Save on this double round trip time for that many pages.

Multiple objects sent over a single TCp connection between client server. Client sends requests as soon as it encounters a referenced objects. As little as two RTT for all referenced objects.

Of course in the non-persistent case, because it is non persistent and you need so many connections. What browsers do is to open the connections in parallel.



Non-persistent HTTP (cont.)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

time

6. Steps 1-5 repeated for each of 10 jpeg objects

NB. TCP uses *3-way handshake* to open a connection. The 3 messages are: SYN, then SYN + ACK, then ACK. Application (e.g., HTTP) data can be "piggy-backed" onto the last ACK. (See next slide for illustration.)

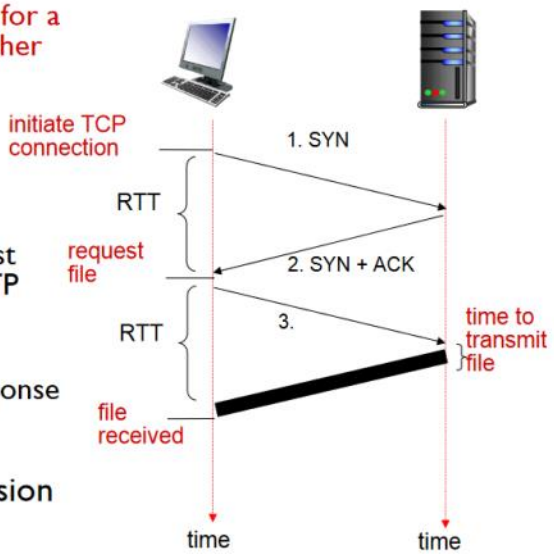
Non-persistent HTTP: response time

RTT = round trip time (time for a small packet to go to another machine & come back)

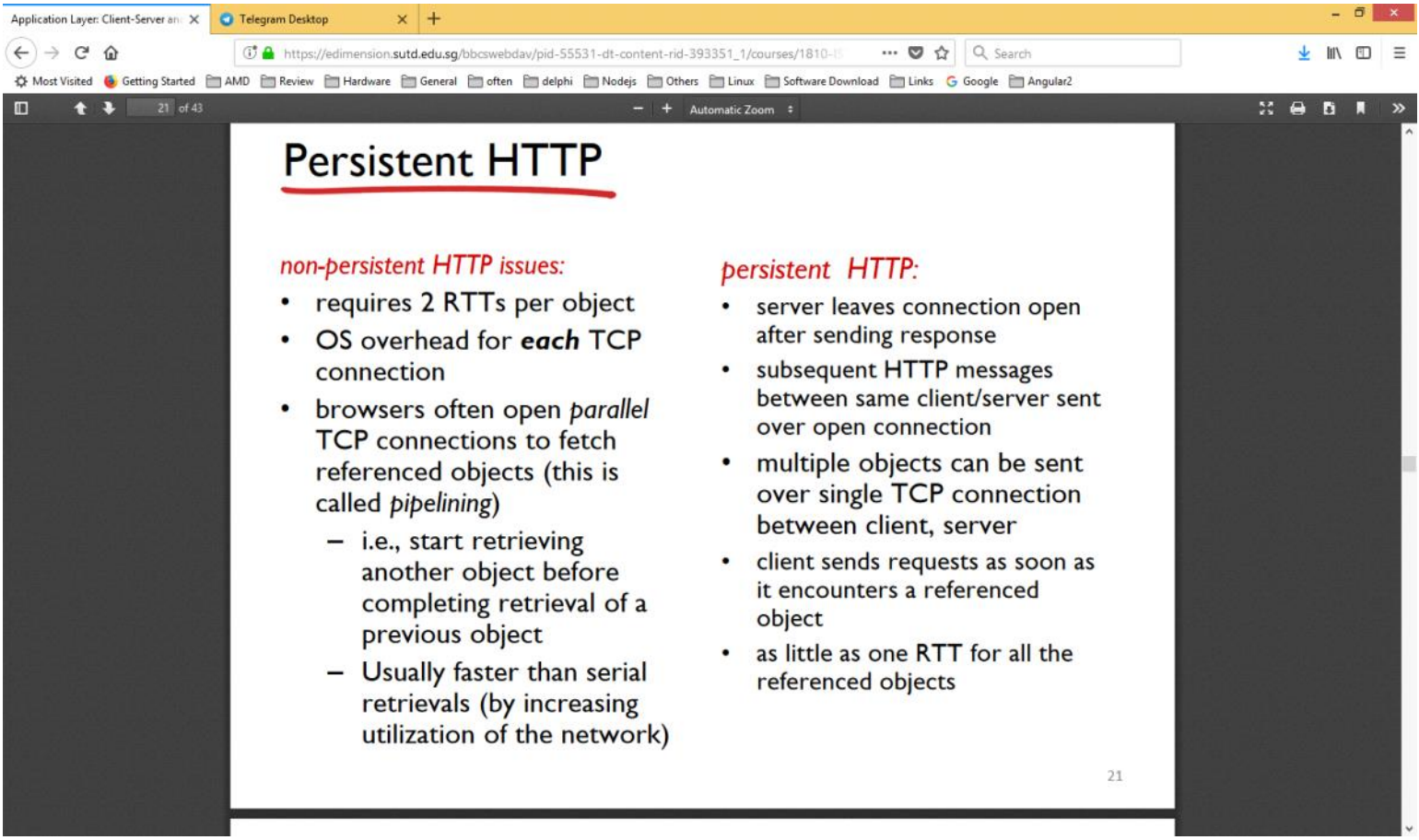
HTTP response time:

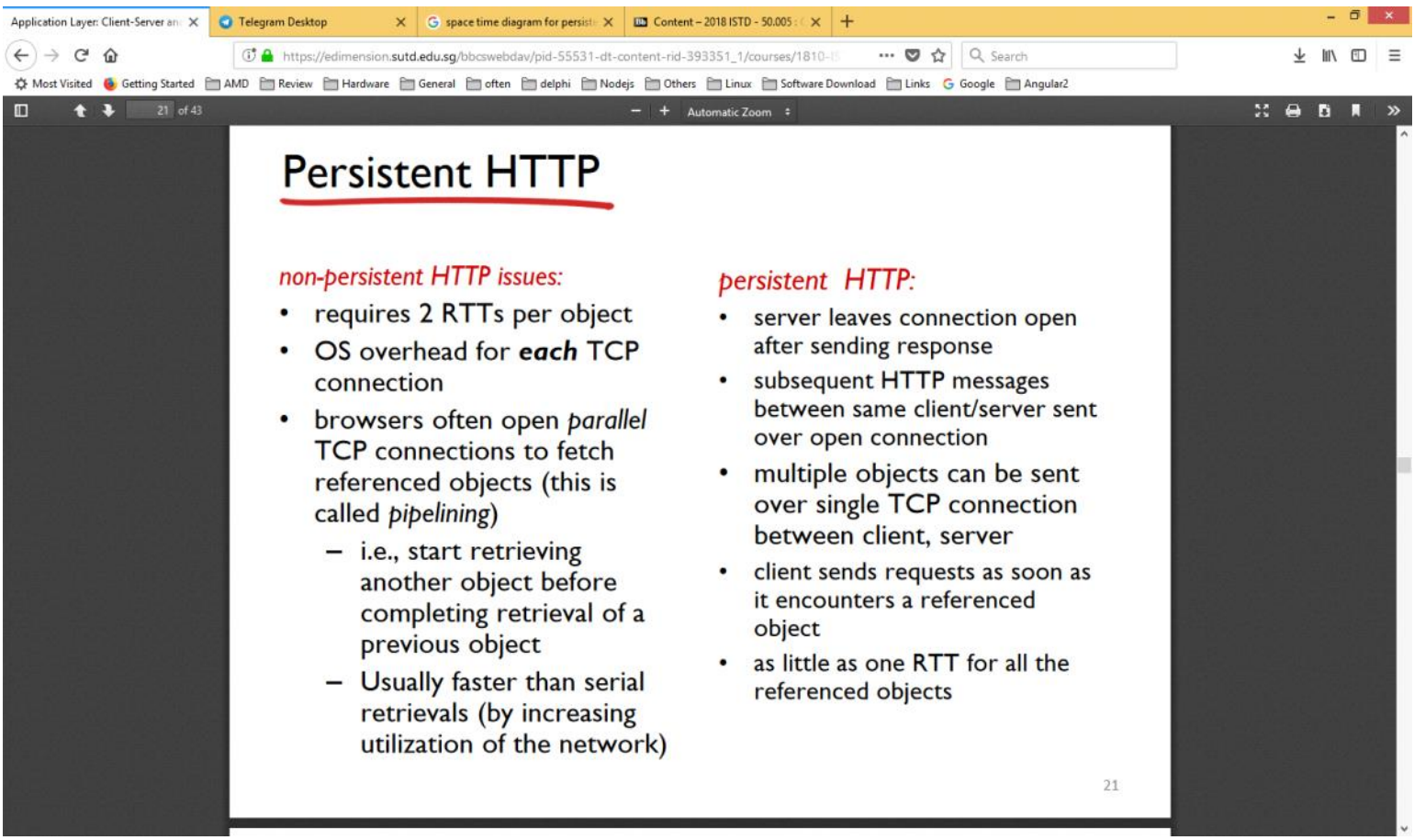
- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time (for *one* URL object retrieved as a file) =

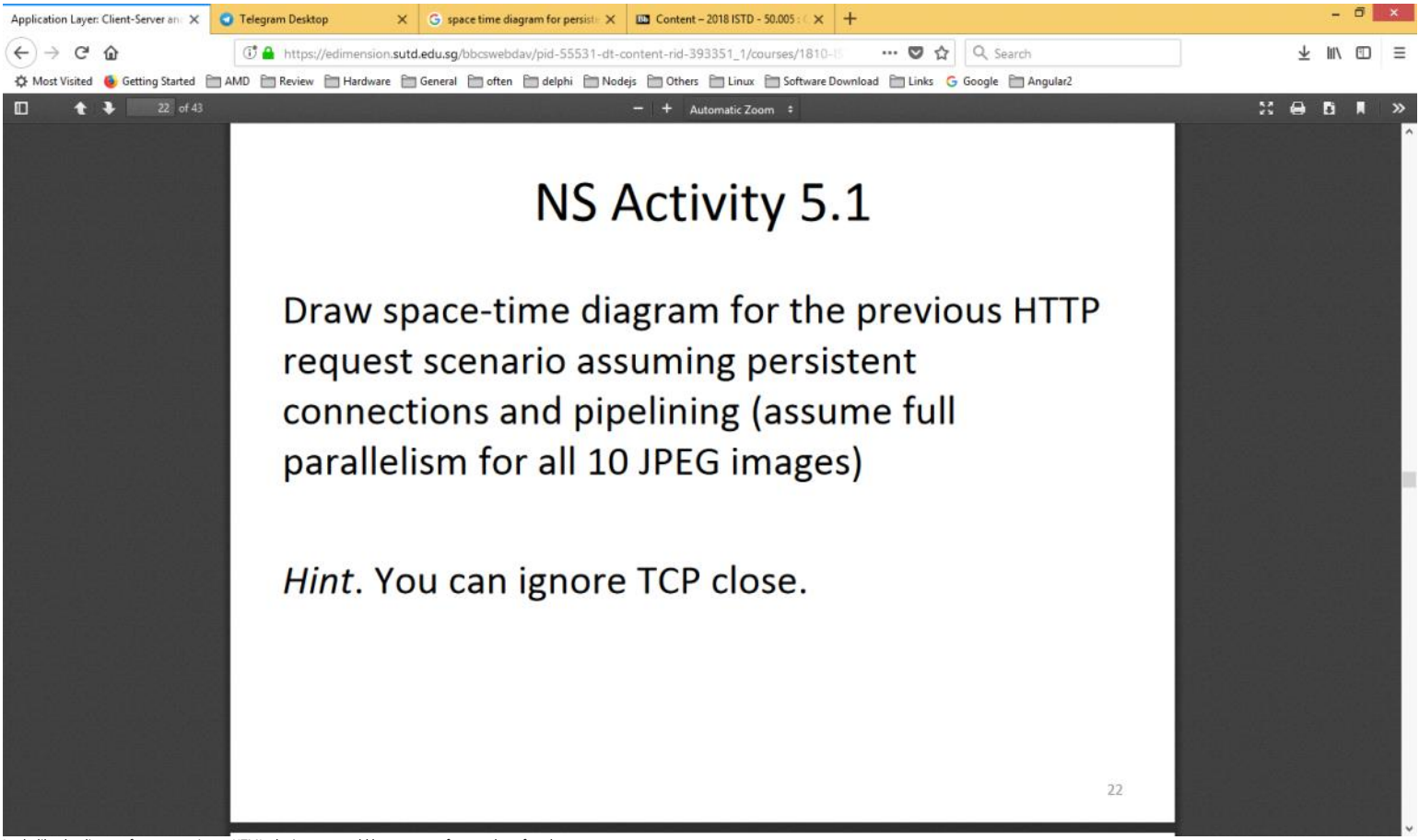
2RTT+ file transmission time



Message 3 = ACK (TCP), with piggy-backed HTTP request for URL


Application Layer: Client-Server an
Telegram Desktop
https://edimension.sutd.edu.sg/bbcswebdav/pid-55531-dt-content-rid-393351_1/courses/1810-
Search
Most Visited
Getting Started
AMD
Review
Hardware
General
often
delphi
Nodejs
Others
Linux
Software Download
Links
Google
Angular2
21 of 43
Automatic Zoom
Persistent HTTP
non-persistent HTTP issues:
• requires 2 RTTs per object
• OS overhead for each TCP connection
• browsers often open parallel TCP connections to fetch referenced objects (this is called pipelining)
– i.e., start retrieving another object before completing retrieval of a previous object
– Usually faster than serial retrievals (by increasing utilization of the network)
persistent HTTP:
• server leaves connection open after sending response
• subsequent HTTP messages between same client/server sent over open connection
• multiple objects can be sent over single TCP connection between client, server
• client sends requests as soon as it encounters a referenced object
• as little as one RTT for all the referenced objects
21


Application Layer: Client-Server an
Telegram Desktop
space time diagram for persist
Content – 2018 ISTD - 50.005
https://edimension.sutd.edu.sg/bbcswebdav/pid-55531-dt-content-rid-393351_1/courses/1810-
Search
Most Visited
Getting Started
AMD
Review
Hardware
General
often
delphi
Nodejs
Others
Linux
Software Download
Links
Google
Angular2
21 of 43
Automatic Zoom
Persistent HTTP
non-persistent HTTP issues:
• requires 2 RTTs per object
• OS overhead for each TCP connection
• browsers often open parallel TCP connections to fetch referenced objects (this is called pipelining)
– i.e., start retrieving another object before completing retrieval of a previous object
– Usually faster than serial retrievals (by increasing utilization of the network)
persistent HTTP:
• server leaves connection open after sending response
• subsequent HTTP messages between same client/server sent over open connection
• multiple objects can be sent over single TCP connection between client, server
• client sends requests as soon as it encounters a referenced object
• as little as one RTT for all the referenced objects
21



Looks like the diagram for non-persistent HTML, the images would be sent one after another after the request. They will have to be received at just one browser. That browser could be parallel receiving multi-threading. We've ignored those details. Don't assume there is no multi-threading going on. Since it is sequential to the server, the client would have to be sequential. Somewhere in the lower layers, they would have to be made in the order of requests made. TCP is FIFO in order of preserving. The object 1 would arrive before object 2 and delivered before object 2. All that reordering would take place at the lower layers.

Let's go on to the format of the HTTP protocol itself. We just briefly alluded to this, but there's a lot more. You do get back the HTML page. There are various ways in which you can request. The request message has to consist of a three-tuple. First there is a command type (Get, post head commands) and then there is a uri (universal resource identifier) and a protocol. That is your three tuple. This is for your request message. You have to have a request message but you have a few optional header lines after the request message. Doesn't make sense not to have a host. The host has to be there otherwise you don't know which URL you are requesting for. The header lines are all optional From the name itself, you can guess which are the header fields.

Host specifies the domain name of the host from which you are requesting the web page. Host here is www-net.cs.umass.edu. User-Agent is telling the browser your user agent. It is what we call the browser negotiation. The browser is aware that you are coming in Firefox and try to send you an appropriate version of URL. Accept: means the type of content. Some of them are important, some of them are nice to have but not mandatory.

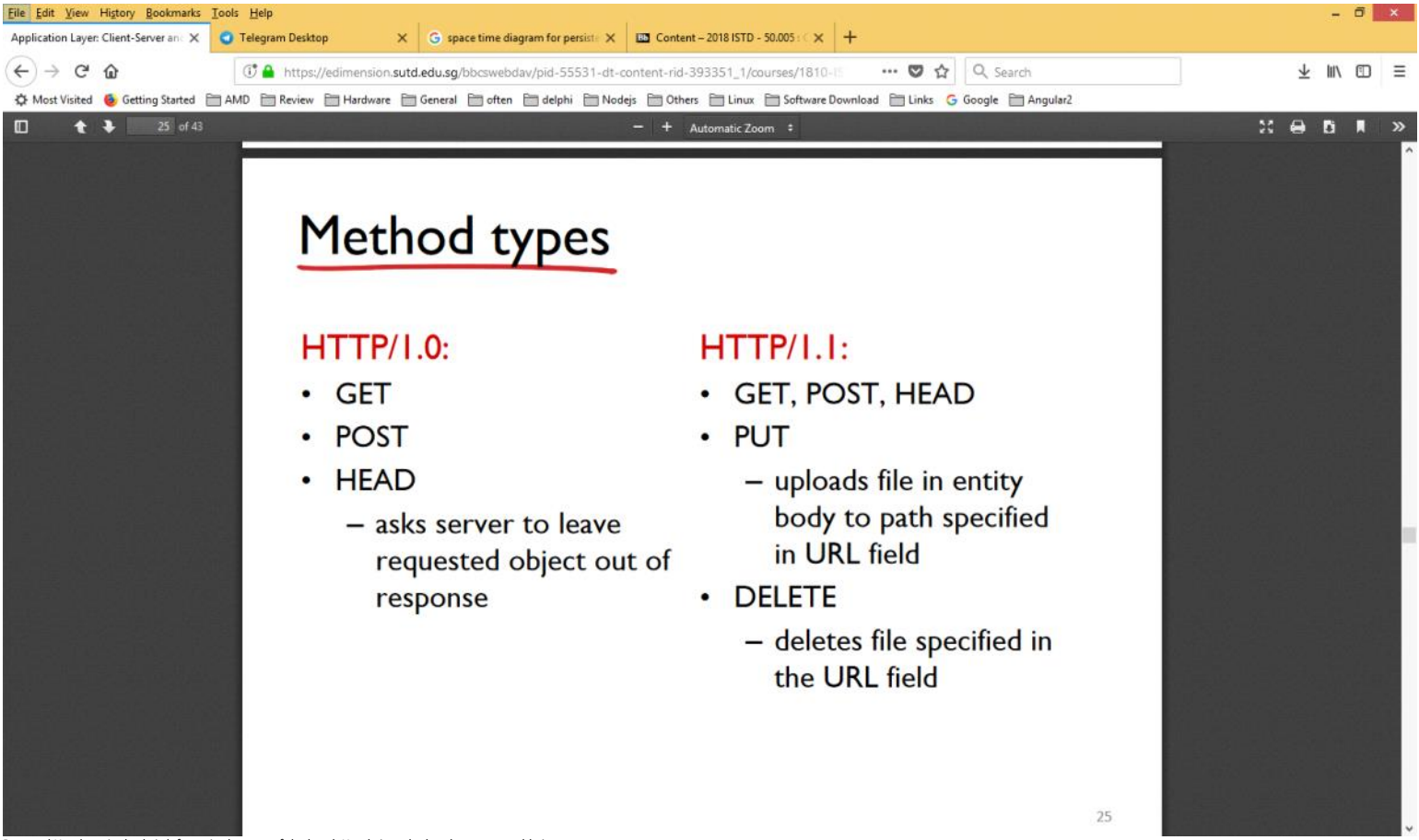
Keep Alive means this is a TCP connection and the connection is to be kept alive. You can see it here, the connection is kept alive, not closed. So even though it is coming on HTTP 1.1, it will keep connection and then close. The browser is requesting the connection is kept alive and specify how long it is to be kept alive. So it is not a good thing if the connection is to be torn down after a 115s. The character set is the kind of character set being negotiated. There are so many standards coming from ISO. There's a way to specify degree of quality by which you accept a particular choice. ISO-8858 is high quality but user is willing to accept anything else with a quality indication of 0.7.

US is en english, then en is general english, and q = 0.5. These are the various negotiations that can be specified. There are many many more. Take a look at these web requests. In this case, what would the url be. It is the host name followed by the resource. Www-net.cs.umass.edu\index.html. That is the whole url.

The good thing about this, it is all in ASCII, very easy to debug, very easy to specify as well. We are talking about the world wide web. It has to connect computers all over the world. In Tim Berners Lee's CERN, this is the way he specify that. He had the HTTP server and client, came up with the HTML message and it is still surviving till today. Carriage return and line feed came from the days of electronic type writers. Specify whole carriage. You only have the line feed character which combines both, but for backwards compatability, you have to specify the carriage return character.

There is a POST, Get and there is put. Put means new resource. We shall look at what the host does. The more important thing is the response.

You first have the method (command) followed by a number of spaces. And then you have the URI/URL/ space , version, carriage return and the line feed.
You have a field name, space, value, and then carriage return and line feed. This is what distinguishes entity from the request (cr and line feed inbetween the two).



Post and Head: entity body is left out in the case of the head. Head gives the header, post would give the header along with certain items in the entity body. So at this point, let me just give you some examples of this, just to show you what happens in a real example.

Let's say we have a web browser (Google). Google has a search engine where we can enter something. That would trigger a get request. This search page is a form, which is handled in the request by sending in some fields as name value pairs. This is the way it works. When I have a form, I can have more complicated forms such as let me just show you "fill in blanks" there's a form that show name password and certain other fields, buttons and certain text. Very bad to send password in HTML because that is not encrypted.

Once this form is filled, the underlying browser puts in a get request and a post request. Sometimes it is in a post request in a name request. That ampersand, name = value is in the post request.

So let's get back to this. When I have a post, I am putting things in entity body, but when I have a get, everything goes into the header. That's the main difference.

With HTTP 1.1, they added a few more verbs like put. Put uploads the file in the entity to path specified in URL field: even if URL doesn't exist, put would create a url and put it on the server. Put uploads file in entity body to path specified in URL. Delete deletes file specified in URL field.

Let's do an activity just to show you that HTTP, a request is actually going over a TCP connection and see how to do it in a secure way. We will go to first of all, which page to access is (www.sutd.edu.sg/education) There's a lot of graphics, and we can do it with a command interface in our Virtual Box. You can do it on your virtual box. Telnet to sutd and pretend you are a browser. Which request are you going to send. Send a get request with the same web page I have just shown you a while ago and that would be education. When I get the second carriage return, it gets this. That was a request HTTP

```
ongajong@ubuntu16: ~
ongajong@ubuntu16:~$ telnet sutd.edu.sg 80
Trying 192.168.2.39...
Connected to sutd.edu.sg.
Escape character is '^]'.
GET /education/ HTTP/1.1
Host: sutd.edu.sg

HTTP/1.1 302 Redirect
Content-Type: text/html; charset=UTF-8
Location: https://sutd.edu.sg/education/
X-Powered-By: ASP.NET
X-Frame-Options: SAMEORIGIN
Access-Control-Allow-Origin: *
Date: Mon, 09 Apr 2018 01:10:32 GMT
Content-Length: 153

<head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found <a HREF="https://sutd.edu.sg/education/">here</a></body>
```

When I hit an empty line feed, the request went over the network and returned to me something. The response can give me information about the request. The response is 302. 302 specifies what kind of request it was. The response had been redirected. This request has been redirected. This is the real location where I should be looking at. This is a secure server, which is why it redirected me.

Telnet doesn't even encrypt your password. Obviously the web server here expects you to come over a secure layer. And so it is redirecting you to a https and followed by resource you are looking for. I timed out in this request. Suddenly I get back the OS prompt. Because by default, it had a TTL. It was a persistent session, but there must have been a default keep alive, and the record got terminated.

Why is it a text, why is it all text? It's not the same as what I got from my browser. When I type in this URL from my browser, I got this nicely formatted page full of graphics, but I got back text and not the other pictures?

Because I'm coming in on a CL shell, the browser knows Jpeg images, knows how to negotiate with the server. It is a lot more that is going on in a browser then what's going on in a simple shell. The essence is I'm retrieving from the backend. This one is just redirecting me with an error code and to another web page and give me some other text information. It also stays over here document moved. Just like in the case of the http request, in the case of the response, we also have a response field, a header section, a blank line. This document is going to be huge, for the response to an http. The whole web page is being sent in the response. Because this is a redirection you have very little response over here.

Let's move to this. We need to connect to the HTTPS, secure server. It runs at port 443. I can't use telnet because it is not secure. I will need to use ssl, and will need to use openssl (secure socket layer, which encrypt communications). s_client gets the functionality of a client, you get the same website you are trying to connect to, where the HTTPS server runs. Secure server runs at 443.

```
ongajong@ubuntu16: ~

<head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found <a HREF="https://sutd.edu.sg/education/">here</a></body>Connection closed by
 foreign host.
ongajong@ubuntu16:~$ openssl s_client -connect sutd.edu.sg:443
CONNECTED(00000003)
depth=2 C = BE, O = GlobalSign nv-sa, OU = Root CA, CN = GlobalSign Root CA
verify return:1
depth=1 C = BE, O = GlobalSign nv-sa, CN = GlobalSign Organization Validation CA - SHA256 - G2
verify return:1
depth=0 C = SG, ST = Singapore, L = Singapore, O = SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN, CN = *.sutd.edu.sg
verify return:1
---
Certificate chain
 0 s:/C=SG/ST=Singapore/L=Singapore/O=SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN/CN=*.sutd.edu.sg
   i:/C=BE/O=GlobalSign nv-sa/CN=GlobalSign Organization Validation CA - SHA256 - G2
 1 s:/C=BE/O=GlobalSign nv-sa/CN=GlobalSign Organization Validation CA - SHA256 - G2
   i:/C=BE/O=GlobalSign nv-sa/OU=Root CA/CN=GlobalSign Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIF0jCCBLqgAwIBAgIMeh3aRJiElzSNEMvzMA0GCSqGSIb3DQEBCwUAMGYxCzAJ
BgNVBAYTAkJFMRkwFwYDVQQKExBHbG9iYWxTaWduIG52LXNhMTwwOgYDVQQDEzNH
bG9iYWxTaWduIE9yZ2FuaXphdGlvbiBWYWxpZGF0aW9uIENBIC0gU0hBMjU2IC0g
```

After you open this shell, that's kind of doing the telnet. After that you do the get request and see what you get.

```
ongajong@ubuntu16: ~
eGgvLQk+KrpLv6Gm0pBLgAYSlqaLlFVmMeoAmd6WWTK1V7d4GrdHoOqPs/miCBJO
enVyMm6o
-----END CERTIFICATE-----
subject=/C=SG/ST=Singapore/L=Singapore/O=SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN/CN=*.sutd.edu.sg
issuer=/C=BE/O=GlobalSign nv-sa/CN=GlobalSign Organization Validation CA - SHA256 - G2
---
No client certificate CA names sent
Peer signing digest: SHA1
Server Temp Key: ECDH, P-521, 521 bits
---
SSL handshake has read 3238 bytes and written 555 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-SHA384
    Session-ID: BC2200001D321110D2853295E68C87FA188031BEA4BFA77A30B63B4664134DBA
    Session-ID-ctx:
    Master-Key: 2595DAB315D9F52775538E18FCC7037932694766888886D9DD6DBE87A8EA4D77DAE7201A08964714674978F0BA30F68A5
```

After entering the same GET /education/ HTTP/1.1, Host: sutd.edu.sg and
Over the telnet session, we try to access the web page. The web page has been surfed by a web page which is secure. This web page has been migrated to that site and use open ssl to get that page instead.

Let's move on to the HTTP response message. In HTTP, we are talking about request and response over TCP. Now I showed you a few options under the HTTP request like get, post put and delete. The response message contains a status line and then a few header lines, and then the response itself. We have first of all a status line, and then a few header lines and then an empty line. Then the data itself, which could be a huge HTML page like the one we just saw. The status line says the protocol, followed by a number whether it was OK and also some text description of what that number is. In this case, whenever you get 200 and ok, the request is alright and treated carefully and the response is there.

There are certain other fields in the response message. This is the date of the response. Date means date and greenwich mean time. Not the time in which object was created. There's a difference between created and last modified time. This is simply the response time (object retrieved and sent to the browser. What software the server is running (apache is running) We will come to web caching, and etag has to do with that. E tag maps on server side and cache side. Otherwise the cache has been revalidated and has to be renewed. Accept ranges has to do with the types of data. Content has to return with the type of data. And then keep alive. This is a negotiation for a persistent connection. It sets the time out to be just 10, maximum is a 100. Keep alive is a persistent connection, and content type is the same as before with the text, charset.

This is a protocol for negotation between server and client, and server would respond with selection of these header views.

Example of get and get response. Here is an example of get request. Encoding sometimes not specified; the browser is saying I can specify gzip and deflate, and connection is keep request. The response to that is right here. The response is followed by a body. This is the object requested here. This is an HTML page that has been delivered in response to that. Any questions? We have get request and request handlers. It is not followed by a body. Get is not followed by a body, it is an exception. Post would be followed by body; you have these fields and the value pairs, but get doesn't.

Put request: this is an example of put. It has a body. It is putting in this website hello.htm. On this host, it is putting this object or html document. In that particular page. That is simple. The response to that

particular put request is this. It says 201, it is not responding before, it tells it was just created. This is just a response to the client that the file was created. Notice that everything was sent in HTML format.

Let's have some review questions

Send some text, three images over. How many response message? Where you have a request. Say a user requests a web page that consists of some text and three images. For this page, the client will send one request message and receive four response message. Each of the objects is sent as a separate object on same TCP conection. They would all come together on the same TCP connection.

Two distinct web pages can be sent over the same persistent connection. True if embedded.

With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages. Totally false.

The date: header in the HTTP response message indicates when the object in the response was last modified. This is false. Just shown when it was retrieved, not modified.

The HTTP response messages never have an empty message body. False

Consider an HTTP client that wants to retrieve a Web document at a given URL. The IP address of the HTTP server is initially unknown. What transport and application - layer protocols are needed in this scenario. What application and transport protocols are needed?

You don't know the IP address, you need to resolve the IP address. It needs two layers: run at the application layer (DNS) but it uses UDP. For resolving the IP address, you will need the domain name system (at application layer) and transport the packets for the DNS. You need UDP. For the HTTP itself, what protocol would you need? TCP. TCP and HTTP is in your protocol
Solution
    For resolving IP address: application layer- DNS and at transport layer UDP

Domain name gets first followed by URI.
Version of HTTP is 1.1
Non persistent or persistent. Persistent (keep Alive)
IP address on which host is running: You don't know the IP address of where the host is running. It is nowhere in the negotiation.
What type of browser initiates this message: This one: user agent is mozilla and why is the browser type needed in an HTTP request message? So that the server can send in the appropriate version of the object. Remember I showed you different versions. That's the negotiation.

Cookies

HTTP is essentially a stateless protocol. Whenever object is requested, it is sent over by the server. Without blinking an eyelid, the server would send it again. It can be so efficient because it is memoryless. We may want to store some information about a web session. For example, let's say that a shopping cart. I'm at amazon and I buy books and I browse to some other part and add to shopping cart. With every mouse click, an email was sent. That is one way of handling it. You could think of having a form. The main idea is that the server doesn't have any memory of the client and sometimes it is desirable of the client to have this memory. Server might want to know transaction one or two weeks ago.

The server might want to keep track of ideas of behavior of client. What happens is the person's identity is not tracked but the browser is tracked; client is tracked. Whoever is using my laptop would get the same response I am getting. It is the browser running on my laptop that is identifying through my server. It works through my cookie. There are four components. The cookie header line of the HTTP response message. Then there is a cookie header line in the next request message. The server is sending 'here's a cookie, if you are sending, please store in the local file'. The cookies are stored in the client, not the server.

When the client makes the request to the server in the same or subsequent connection, the cookie (identifier) is kept along with it. The cookie file is kept at where the user is. The file here is managed by the user's browser. Th back end here, the server also has their own database.

This database would keep an association between this client and the various transactions the urls the client has visited and the database has a summary of the client's past behavior. That's the backend database.

User-server state: cookies

many Web sites use cookies

four components:

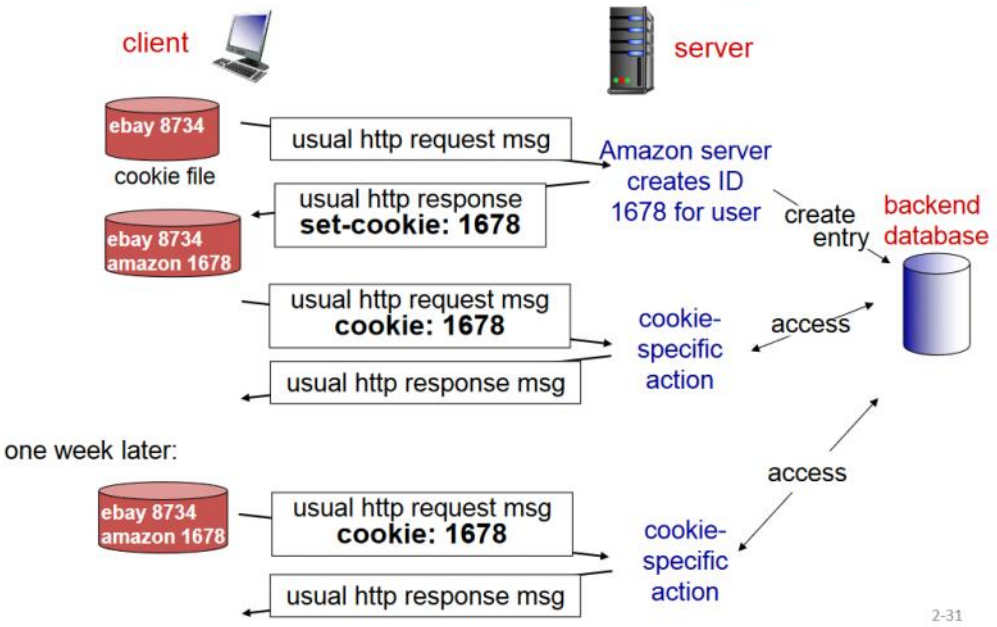
1) cookie header line of HTTP *response* message
2) cookie header line in next HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site

example:

- Susan always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

2-30

Cookies: keeping "state" (cont.)

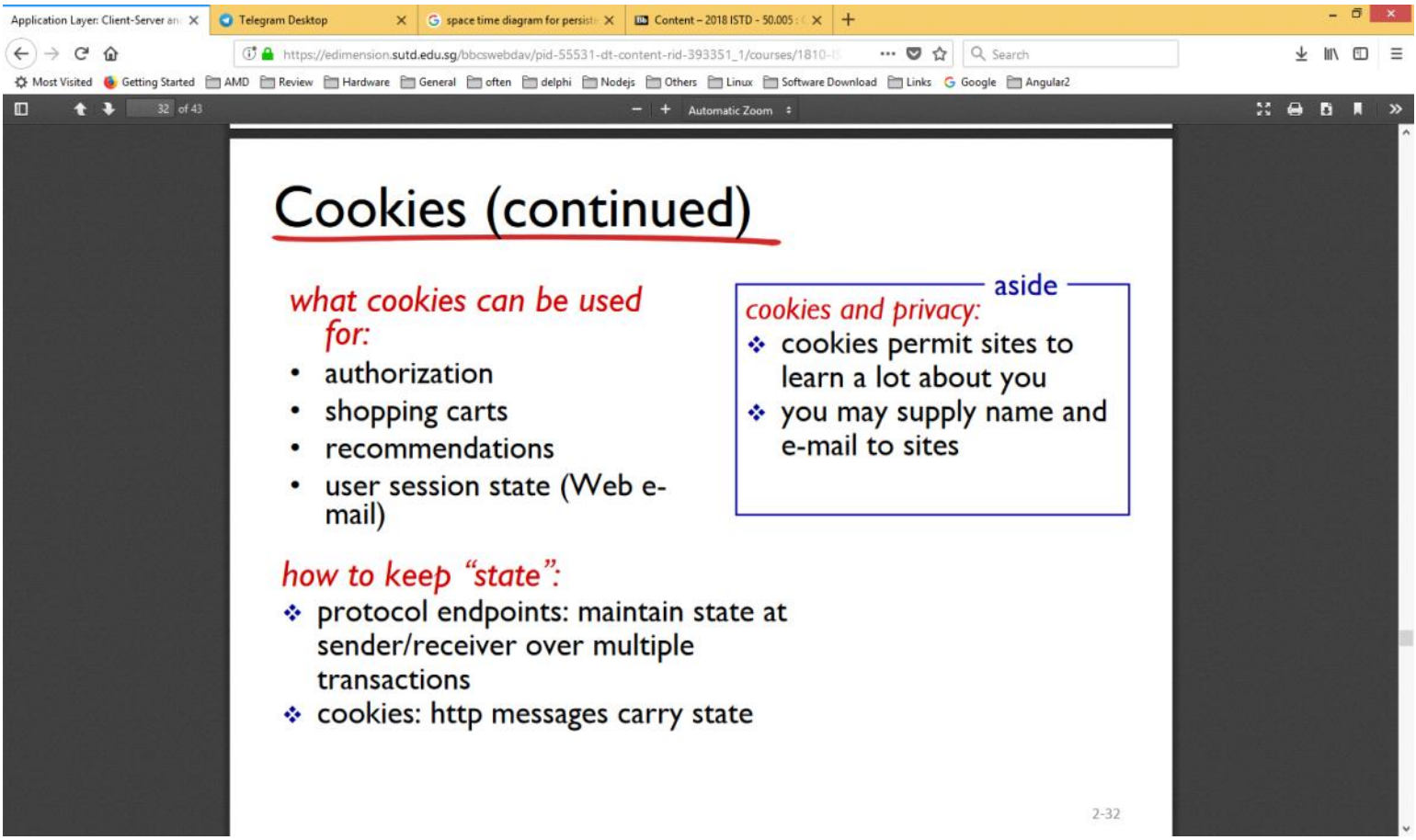


2-31

The very first thing the message is: set cookie to 1678. The client enters amazon with the id 1678 in the cookie file. Thereafter, this could be one week later or in the same session. Whenever client enters the same site, every message is sent with the appended line. What could be the cookie specific actions?

If the user is willing/agreeable to store personal info like name, credit card number, then all of that is

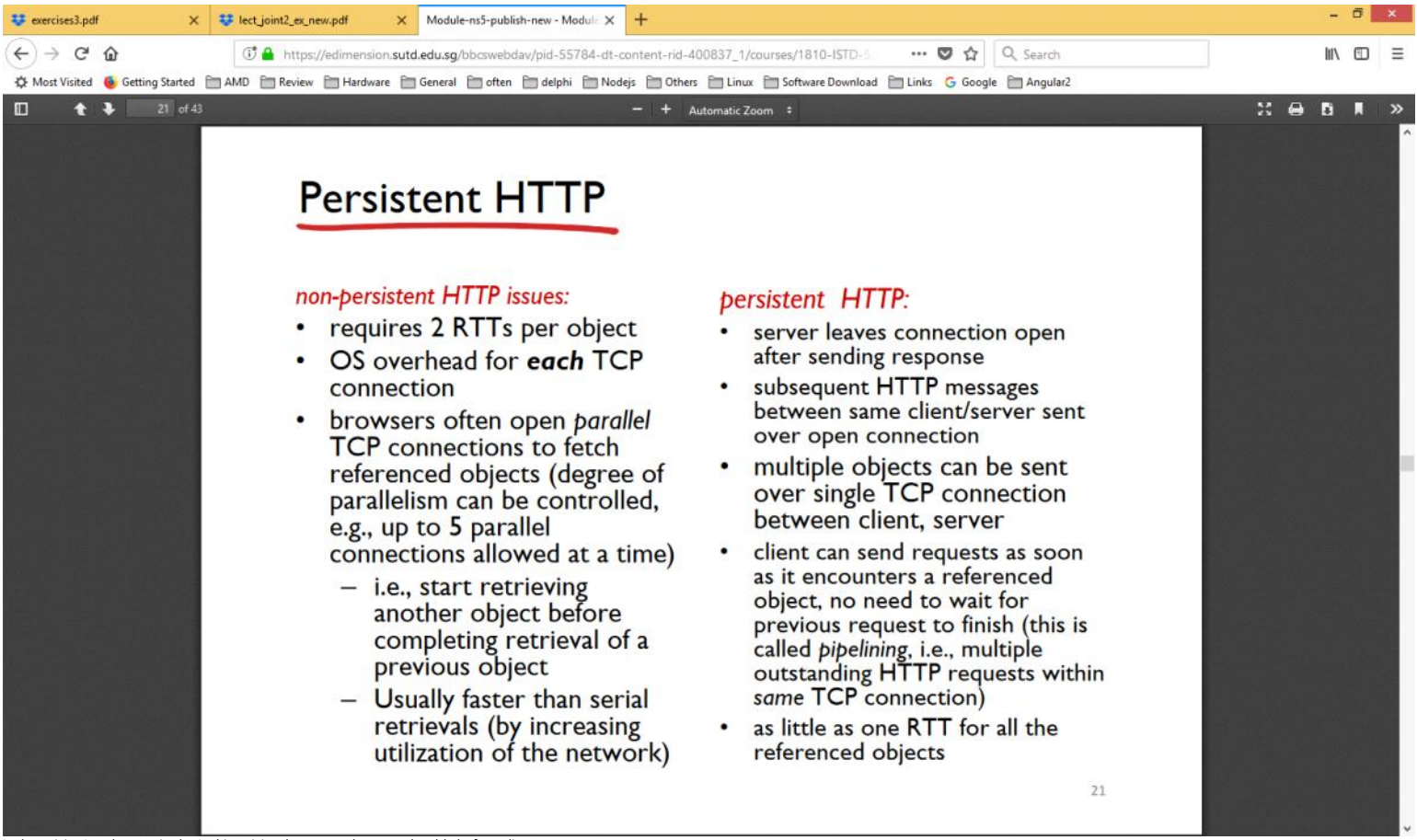
stored in the backend database of the server. Now the server does not ask for info altogether, and faster recommendations, submission etc.



Wednesday

Wednesday, April 11, 2018 11:35 AM

In a sense, we've been using the terms synonymously (pipelining and parallelism). Anything that is pipelining is done in parallel. There is something happening to each one in parallel. But we've decided to come up with this decision. Even if they are one and the same name. in Http1.0, we would only use term parallel. For concurrent TCP connections at the same time. One request response that is sent over TCP connection. Then TCP connection is closed. Possible to have multiple TCP connections without waiting for the response of the first TCP connection. I'm allowed to start the second third and fourth TCP connection. These requests can go in parallel. We will see a detailed response when we use wireshark. The example on wireshark will make that very clear. When we say parallel, we talk about HTTP 1.0, and when we say pipelining is possible in HTTP 1.1. I can send out requests on the same TCP connection.



In the activity, I made one mistake. In this activity, there was a three -way handshake from client to server. This response that comes along with that web page is one HTML file. Until client has examined that HTML file, the client won't know that the embedded message has 10 objects. It needs to fetch the page (URL), and gets back one HTML page, finds out 10 objects sitting inside. Then 10 requests sitting inside, for the 10 objects and that comes back as a sequence of pipelined JPEG objects. If you really want to put the whole thing together, you have three round-trip times. And then you have the time for this first HTML, so plus t0. Each one of them takes t1. That takes 10 x t1.

That would be the total time of that URL.

So cookies are the way to maintain state. Within a session. How is it done? It is done with four components. The first is a cookie sent from server to client.

Server is interested in maintaining the state; wants to know things about the client. Client knows everything about server. So Server sends cookie to the client. If client willing to accept cookies, it uses that cookie number in every subsequent response to that server.
Also contains one additional header line. That gives that to be identifier. There are two items of data. One at the client's end, one at the backend database. Both of them bear the cookie identifier. The client has the cookie filed already. The client visits another website. Then for the first time, it is going to visit Amazon. It creates a cookie, an identifier. 1678. and creates an entry in the backend database. And then it sends a cookie.

In its regular HTTP response, then, perhaps in the same session. The client has added amazon now in its cookie file and you can see the next request message already has a cookie. Since it already has a cookie here, the server finds the cookie in its database and conducts a cookie-specific session.

If it is an online purchase, it could be a view of the shopping cart. Shopping cart is being added to at every transaction. It could be a recommendation. Recommending the end-user to certain additional merchandise. This session is over; one week later, this client registers the same site at amazon.

Since it is going to amazon, it supplies the cookie in a header line. Again the server takes a specific cookie action.

There are two types of cookies; the cookie we see over here is a permanent cookie/persistent cookie because it doesn't have a TTL. Whereas a session cookie that are terminated and are actually removed at the end of the session.

In this case, we have a persistent cookie. We talked briefly about this. We mentioned authorization; if I visited site before, I've authenticated the site. Making submission for conference publication. Shouldn't ask me for all my info all over again. We can infer what my authorizations are. When we use for

shopping carts, recommendations. And we add a session for messages.

Email is supposed to be memoryless. Using the SMTP protocol, my last recorded email/status of email mailbox, I retrieve it from the mail server. If it is google or hotmail or whatever. I would like to send a session specific cookie, to know that Google knows which client I am coming from. It identifies the HTTP connection by the client. We talk a bit about the potential problems with cookies. Because it is compromising information about you. Could be given to other third parties. There are problems there.

Cookies can be used for certain kind of inferences. Here is an example where a market research company is trying to find out effectiveness of an Advert. The time duration of placement of web and the purchase of a piece of merchandise. If the advert is shown at this time, the purchase is shortly thereafter. This kind of analytics can be done using cookies.

This brings us to the topic of proxy server. What is a web proxy server? Normally, if a HTTP protocol is basically stateless, whenever an object is fetched from the network, basically, in the long distance. Not intranet; we are talking about something that is far far beyond like Europe. Because it is stateless, network doesn't keep any state of HTTP transaction.

We could say by having a server somewhere in between, close to me, or a web cache, and what it does is to store that object in the time in which it was first fetched, so it would not need to go all the way to the web origin server. They are very much used by content delivery networks. Companies are building huge servers just for caching web pages. But it could also be like small companies using a web cache because it improves performance.

The way it works is when an object is requested and not found in the proxy, it is fetched from the origin server. The very first request is fetched from the origin server. Subsequent requests need not go to server, but the proxy here is both a server and a client. It is a server to this community of users, and it is a client to all the servers on the other side. It has to act as a server as well as a client. We look at TCP connections. It can be done in a mode which is not waiting and you can move on and do something else. The fact that a server acts as a client shouldn't surprise us.

Where are the benefits coming from? These are the benefits of having a proxy server. It is faster for the clients. Especially if there is a bottleneck between proxy to origin. The access link is usually a lot slower. If you have a cache, a certain percentage of the objects are already found within the cache. We look up a small example to see what happens. Besides the fact that is server.

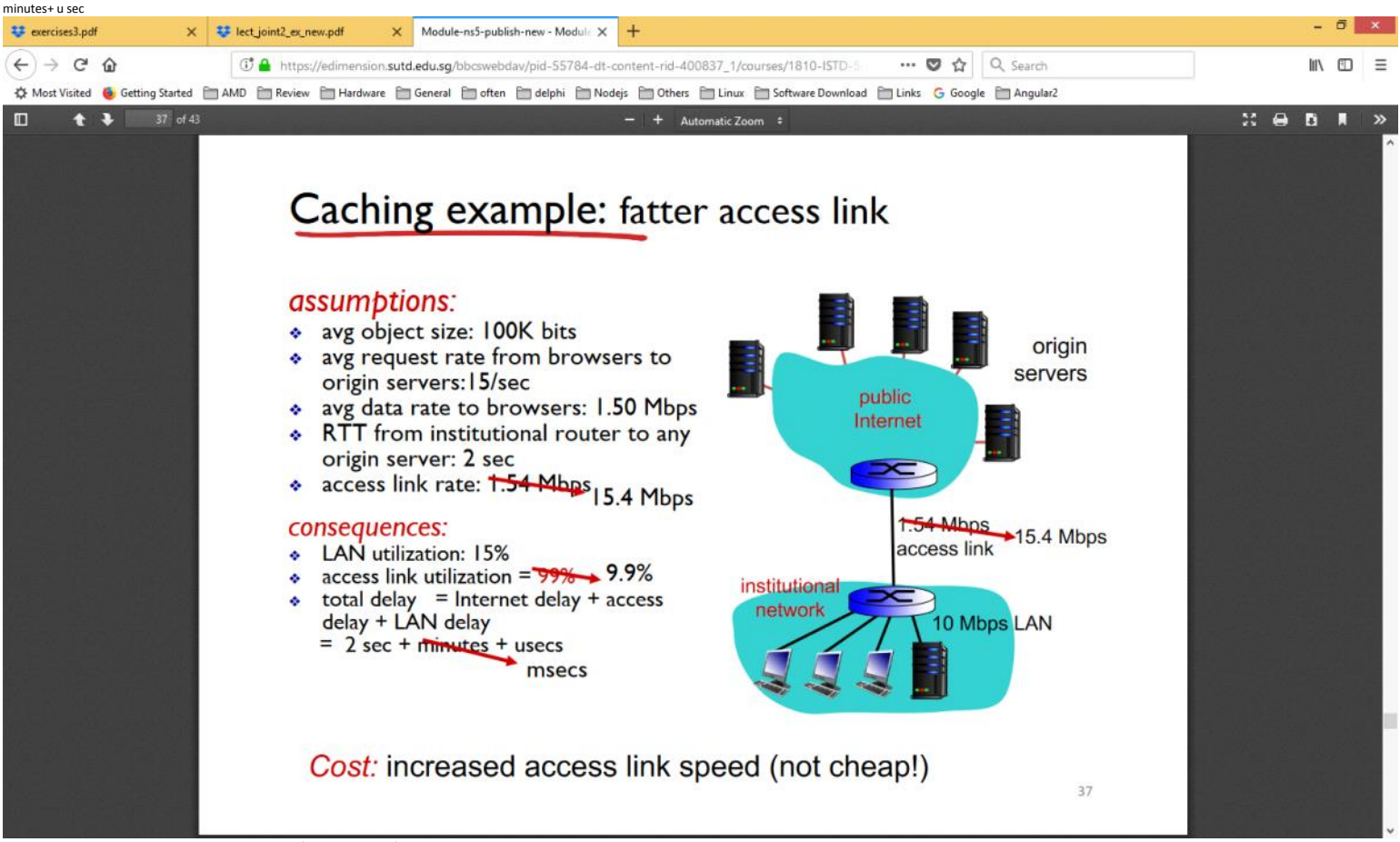
It can be said to distribute the load; not all clients have to go to the origin. Load on the origin server is reduced, not distributed. Well you've distributed it to the proxy server. (reduces traffic on institution's access link) Typically cache is installed by ISP. We are using a loose definition, where SUTD, company, residential is interested in deploying a web cache.

If proxy server is down, you won't be able to access the server? Every access to web goes through proxy server, so I have a way to work around proxy server or a route to work it through.

Average object size is 100K. Rate is 15 objects per second. The avg data rate to browsers is 1.5Mbps. Access link is 1.54 Mbps, and RTT from institution router to any origin server: 2 seconds. The delay from the router here, to and from, is going to be a few order of milliseconds. The delay over here in the public internet, the first hop router, back to the origin server and back.

We know that the public internet is highly overprovisioned. At the core end routers, routers are lying at a low end utilization. That is what gives you this reliability of 2 seconds. That is a good estimate. The real bottle neck is between the data rate to browsers and the access link rate.

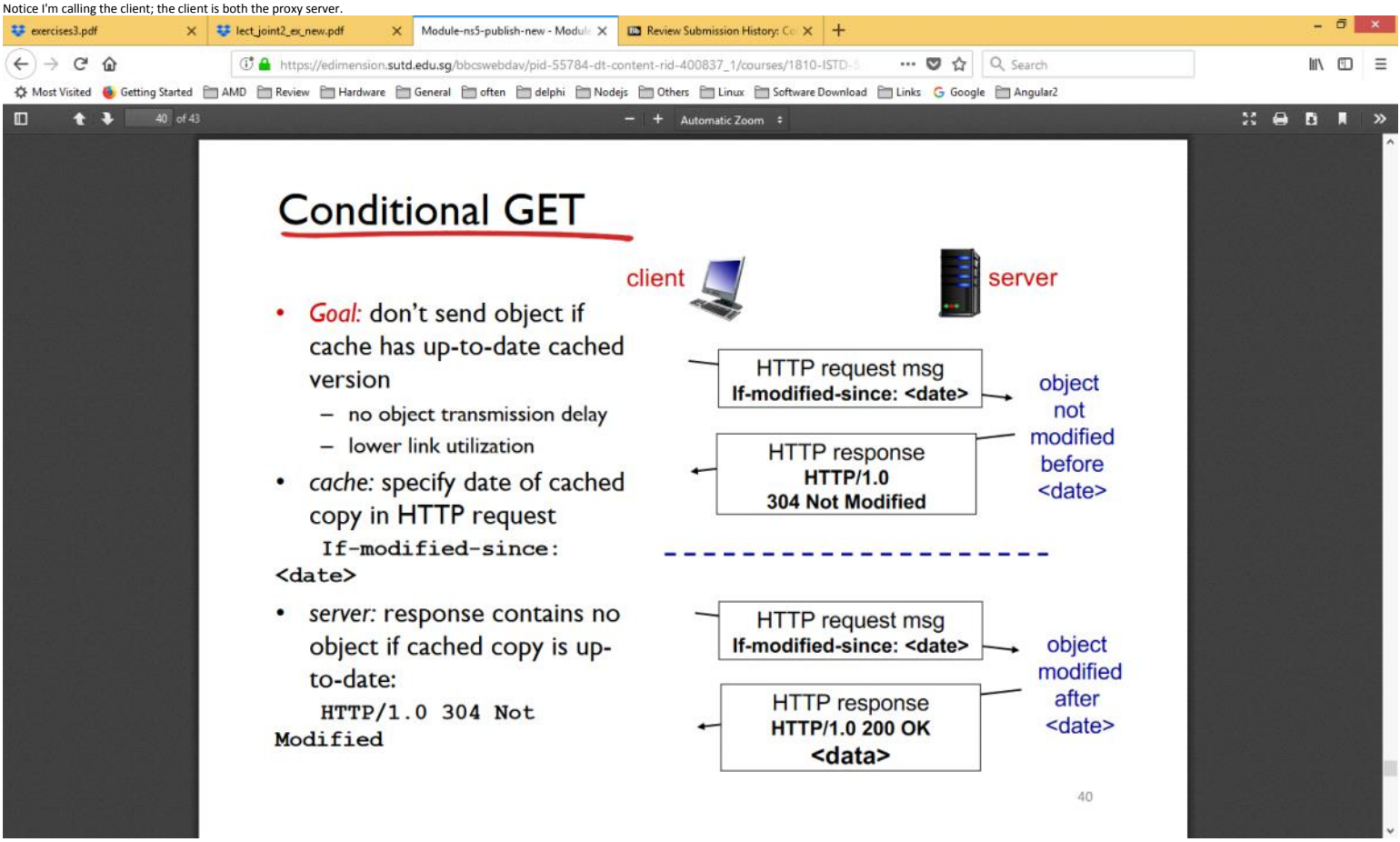
So the link utilization is 1.5/1.54 = 0.974 The total delay = Internet delay + access delay+ LAN delay = 2s + minutes+ u sec

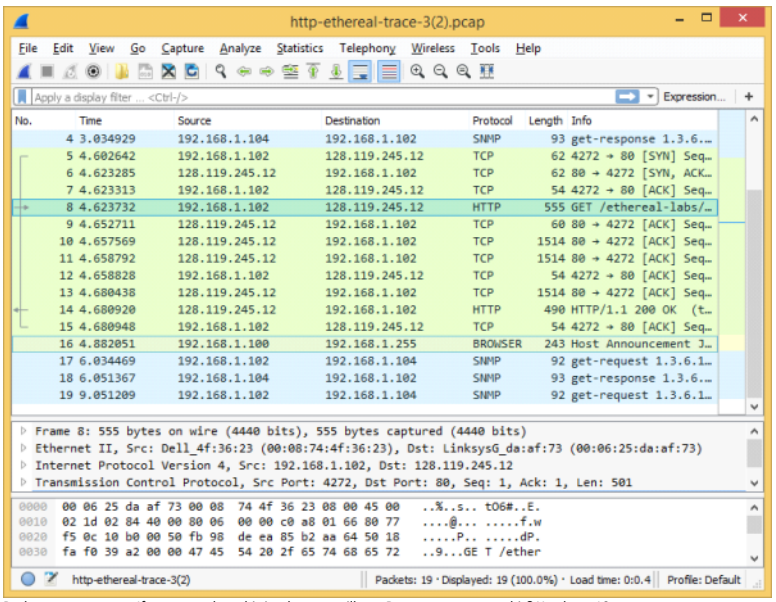


This bigger bandwidth is charged higher. In the order of 100s or 1000s of dollars per month.

We have to introduce something called Conditional GET. If a specify in GET request, and it has a certain date and time on it, I'm asking the origin server to send me this object only if it has been modified since this date. If the value in the cache. This is a generic client and server. The client is saying if this object has been modified since this date, please send it to me. Otherwise, send me an empty response. The server says that it is not modified before that date and replies with an HTTP 1.0. It is 304 not modified.

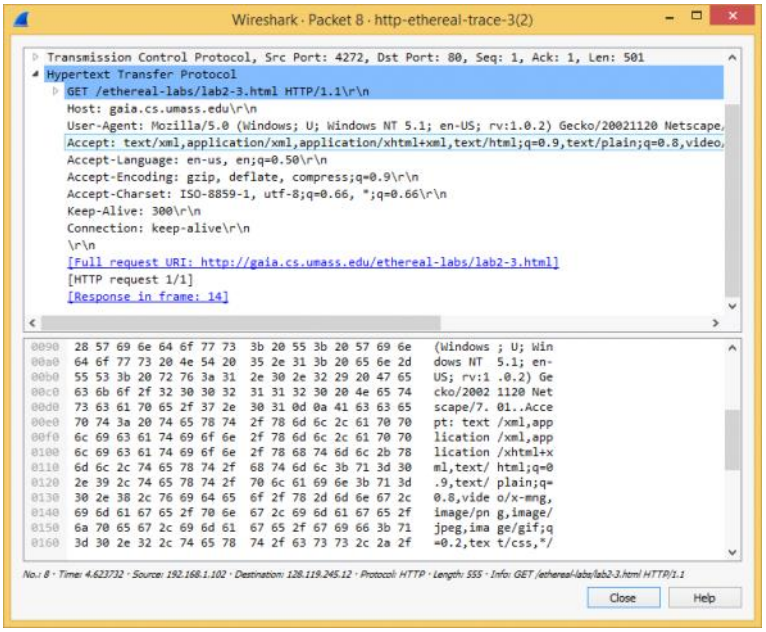
Notice I'm calling the client; the client is both the proxy server.

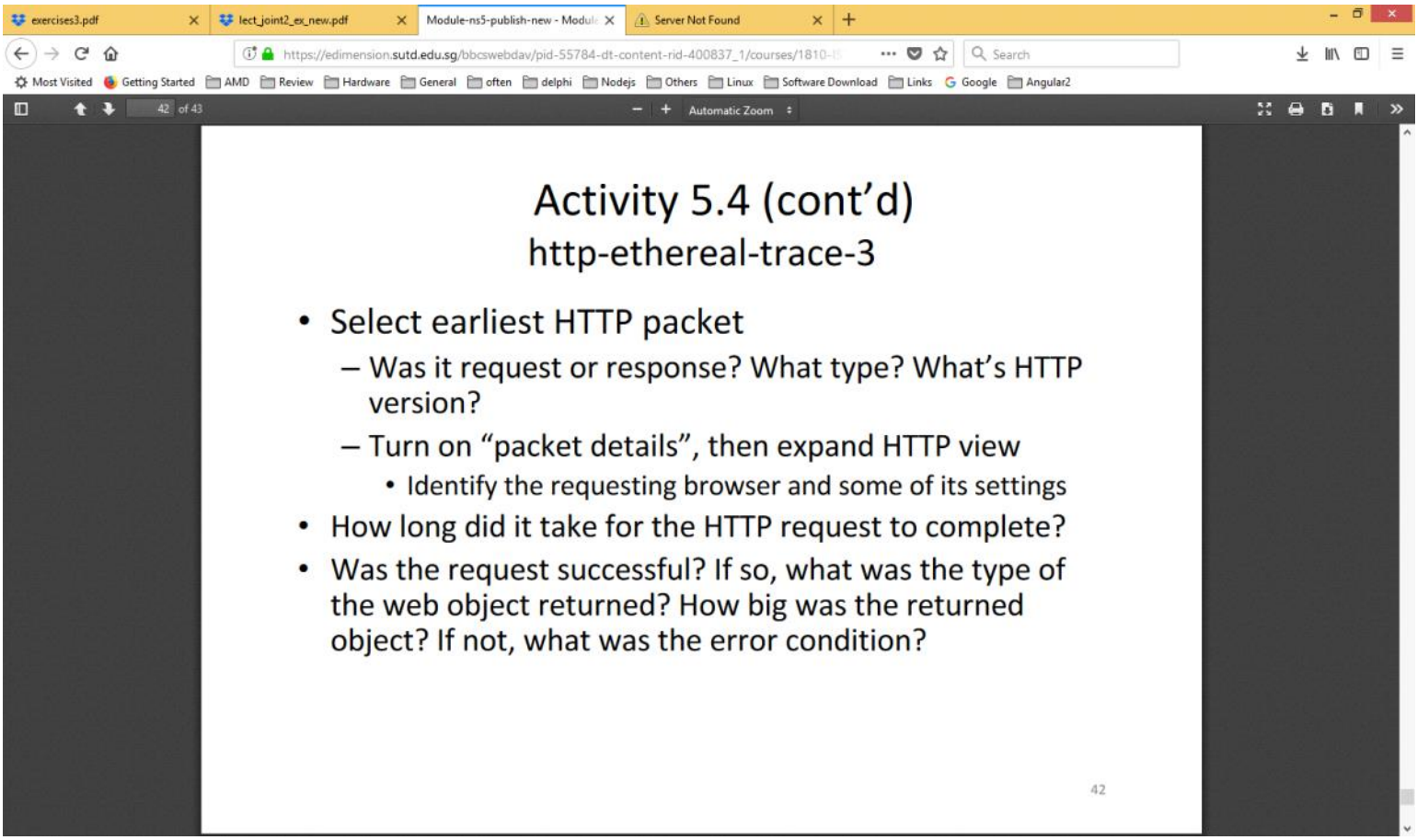




Packet capture, trace. If you open that, this is what you will get. Does every one see this? You have 19 packets in this trace. It is a short one. Packets ordered by their numbers. If you want further details you will see each packet and the first four packets are SNMP packets. Simple message protocol. The very first HTTP packet is number 8.

The number here is the ordinal number, then the source, destination, protocol, length in number of bytes and some additional information. Whatever is appropriate for that particular protocol. That information is presented here. You can see the power of such a device; because it can see each and every packet, every frame, that is going out on the network. IT is also called a packet sniffer because it analyzes information that goes within those packets.





It is a response, Get, Version 1.1
Netscape 4.623 s
Successful, text, 4500 Bytes

exercises3.pdf | lect_joint2_ex_new.pdf | Module-ns5-publish-new - Modul | Server Not Found

https://edimension.sutd.edu.sg/bbcswebdav/pid-55784-dt-content-rid-400837_1/courses/1810-I

Most Visited | Getting Started | AMD | Review | Hardware | General | often | delphi | Nodejs | Others | Linux | Software Download | Links | Google | Angular2

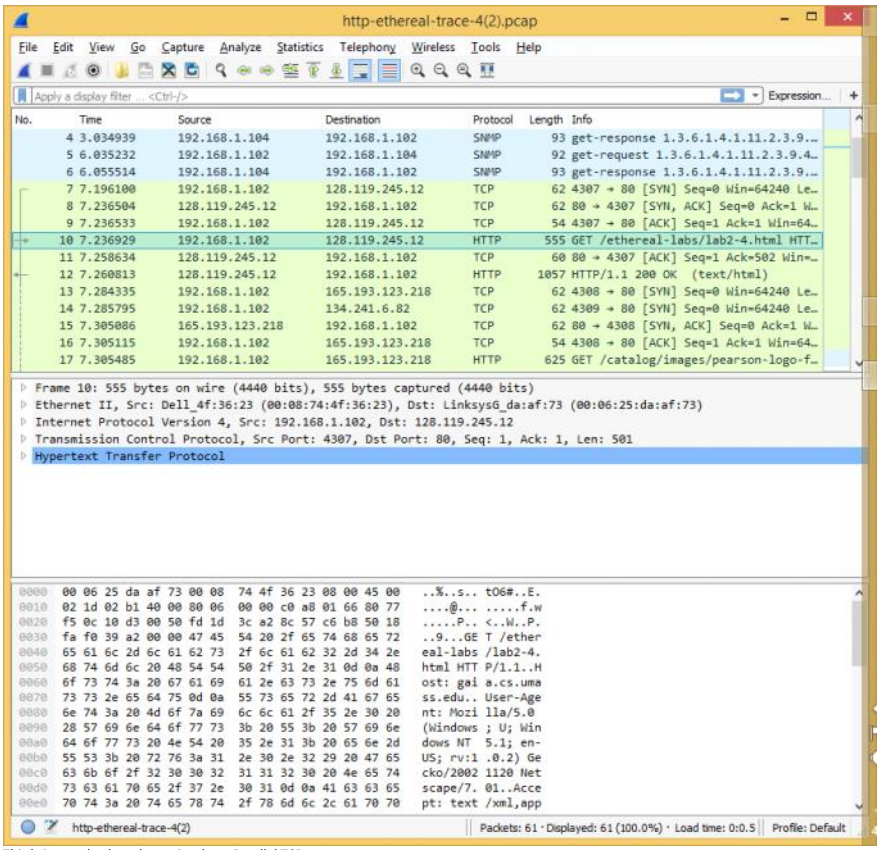
43 of 43 — Automatic Zoom

Activity 5.4 (cont'd)

http-ehtereal-trace-4

- Now open http-ehtereal-trace-4
- What's the URL of the HTML web page being retrieved? What version of HTTP was used?
- How many embedded objects did the HTML page contain? What kinds of objects were they respectively?
- Did the web client use new TCP connections for retrieving the embedded objects? [*Hint*: Look for any TCP SYN message that precedes an HTTP GET to the web server]
  - Why were new connections used or not? [*Hint*: Examine the HTTP GET requests to identify the web server for each object]
- Were parallel connections used?

43



This brings us back to the topic: about Parallel TCP.
Still gaia.cs.umass.edu/etheral-labs/lab2-4.html
HTML version still 1.1
How many embedded objects did it contain? This is a get request: have to get the response for it.
12: is a HTTP response.
2 objects: gif image and a jpeg image.
Use new TCP connections for retrieving embedded objects? We did something like this: We sent out a request, we've gotten a response. When we examine the response, we find that there are two

embedded objects. We need to look further to find out whether to use two new TCP connections or not.

Because we see the ACK coming back. The sync + ack and then the ack. This ack is being sent by the browser to the server and then followed by a get request. What are the contents of get request? It is catalog image pearson. The gif object came from pearson. We are requesting the gif object from pearson. We can see the second sync ack correctly. So indeed, both of these TCP connections were set up in parallel, and the objects are being fetched in parallel. Because they are residing in different objects. What is the original website.

Domain is gaia.cs.umass.edu. Gaia is the host. Cs .umass.edu is the domain. It is like you know: you give names to machine. The second host is manic.cs.edu.
Because this is TCP, TCP socket is identified by host name as well as port number. Port number is 80 (not secure) but the host name has changed. This would need a new TCP connection. If this here was gaia, would it need new TCP connection? No, because we use HTTP 1.1, and we are negotiating, it will be keep alive. If this object is sitting in gaia, it would be in host attack domain. The TCP is associated with host IP and a port number. And now we have a new host IP.

Was parallel or pipelining. There was pipelining because we are targeted three get requests, three servers. The first one finished first. The first get request finished, and then we did two in parallel.