IT441: Network Servers & Infrastructure

CLASS 1: 29 Aug 2005
13:30 - 16:15

Except where otherwise noted, these slides are © copyright Charles Snow, 2005. Permission to copy these slides is granted so long as each slide contains the copyright notice “© C. M. Snow 2005” appears on each page and the copies are made freely available, i.e., not sold. Students are advised that they should not use University consumables (ink, toner paper) to produce printed versions of these slides. Rights for all other use of this material are reserved by the author; permission for any other use must be obtained in writing from the author.

IT441: Where and When

• live classroom presentation, Mondays 13:30 to 16:15
• originating from Prince William campus: BRH:258

IT441: Where and When

• live classroom presentation, Mondays 13:30 to 16:15
• live-by-Internet
• recorded as given in class, may be played back anytime

IT441: Where and When

• live-by-Internet uses GMU’s Network EducationWare NEW software for distance learning; http://netlab.geo.georgetown.edu/distant

NEW: Network EducationWare

• provides, live, in real-time:
  • overheads of class presentation with
  • instructor annotations and
  • instructor voice, synchronized to overheads
  • audio graphics
  • live video of classroom
NEW: Network EducationWare

- provides, live, in real-time:
- lets students ask questions:
  - by typing them in
  - by speaking them (if client does audio)

- provides chat-room resources for students
during lecture (at any time, actually)

- entire lecture recorded for later Internet playback (excluding live video)

Using NEW

- need client software: currently runs only on Windows platforms (98, ME, NT, 2000, XP)
- minimal machine configuration:
  - P350 or faster + ≥ 64 Mb memory
  - IE 4.0 or later, Netscape (6.0 or later)
  - standard sound card
  - ≥ 56 kbps connection to Internet
- see http://netlab.gmu.edu/disted to try it and/or sign up
  (need password to get courses)

Getting and Using NEW

- go to http://netlab.gmu.edu/disted
Course Resources

- required text: *Computer Networking with Internet Protocols and Technology (1st edition)*
  - William Stallings
- course web site: [http://netlab.gmu.edu/it441](http://netlab.gmu.edu/it441)
  - contains printed copies of slides (pdf)
  - other course related info
  - background material

Resources: Instructor

- Instructor: Charles Snow
  - email: csnow@netlab.gmu.edu

Office Hours
  - Prince William Campus
  - Bull Run Hall 222A
  - Friday: 10:00 - 12:00, 14:30 - 15:30

Resources: TAs

- check the course website for updates
  - [http://netlab.gmu.edu/it441](http://netlab.gmu.edu/it441)

Students must:

- course prerequisites:
  - IT341 Network Essentials
  - MATH 108 Introductory Calculus with Business Applications
  - MATH125 Discrete Mathematics

Course Project:

- two options for projects:
  - Plan A:
    - series of small lab exercises (4 or 5) configuring/operating network services on Windows and Linux platforms
    - given out in second half of course
    - each lab has exercise + questions to answer
    - no teams: all labs are individual efforts
  - Plan B:
    - given a scenario, develop a feasible plan with
    - involves finding/specifying all needed resources to implement solution
    - interim project work required (e.g., status report(s))
    - produce a report and a short presentation on the solution
    - may involve programming
    - can work in teams of up to 3 members
    - but effort of each individual must be clearly identified in all work
Course Project:

- ≥ 2 options for projects
- students commit to one or the other plan at start of semester
- no changing of plan thereafter

Samples of Plan B Scenarios

1) airport traveler assistant:
   - provide infrastructure and client-side resources to give a traveler a nearly fully automated travel guide
2) tsunami relief network:
   - implement, in a short time, an Internet system on-the-ground in the area affected by the Dec 26th tsunami for the use of all involved in rescue, refugee management, and reconstruction
   - be aware that the full scenario description may not be nearly so detailed as statement-of-work descriptions you are used to from other courses
   - but you still have to provide a “full” solution...

IT441 Grading:

<table>
<thead>
<tr>
<th></th>
<th>Plan A</th>
<th>Plan B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework Exercises</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Midterm</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>Project</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Final</td>
<td>40</td>
<td>35</td>
</tr>
</tbody>
</table>

no extra credit work option

- assignments due by end-of-day on due date
- assignments handed electronically
- grades are proficiency based

IT441 Grading, cont’d:

- important that you demonstrate:
  - understanding of material
  - ability to reason with it
- final exam
  - is comprehensive
  - scheduled for 16 May 2005, during class time
- basic rule: if we cover it in class, it can be on an exam
  - includes reading material suggested but not necessarily explicitly covered in class

IT441 Exams

- all students must be present, on campus, to write exams in the course
- both midterm and final
- bring GMU ID card and be prepared to show it
- arrangements for missing an exam must be made before the exam

What we look at

client
Server-side Considerations

- server-side questions:
  - who do I listen to? not listen to?
  - does it matter who they are?
  - what kinds of requests do I respond to?
  - how do I make a response?
  - all by myself, or do I get help from others?
  - what do I charge? how do I collect?
  - what formats can I respond with?

Network Considerations

- some network questions:
  - how connect?
  - route? does it matter?
  - how long stay connected?
  - context?
  - quality of service?

What we look at

- what do we need:
  - to make this work
  - to make it work better? faster? cheaper? bigger?
  - to expand to do more?

Technology Focus

- this course focuses on the technology needed to support client ⇆ server interactions
  - theory
  - hardware
  - software
  - costs
  - capital
  - on-going
  - operating
  - expanding/contracting
**Why Theory?**

- I just heard that light travels faster than sound.

- Or, knowing what I should know when I speak, just do it: lips appear to sync up with my words.

- A little knowledge can be a dangerous thing.

- He proved it when he heard me yell.

---

**What’s a Server?**

- something that provides a service

  - elements:
    - hardware
    - software
    - energy
    - $\rightarrow$
  
  - impose constraints on ability to provide service

---

**What’s a Server?**

- something that provides a service

  - elements:
    - hardware
    - software
    - energy
    - $\rightarrow$
  
  - impose constraints on ability to provide service

---

**Server: Inside to Outside**

- examine briefly a server from the inside out
- server’s purpose is to provide a service
- service is provided by a program running on the server

---

**A basic computer:**

```
memory

input device

CPU

output device
```
A basic computer:

- **CPU**: executes instructions retrieved from memory.
- **bus master**: controls most of the time, may surrender bus to other devices for brief intervals, can generate traps for failure conditions.
- **memory**: stores bit patterns (instr, data), organized as flat, linear sequence of bytes, is byte addressable (usually), may generate traps for error conditions.
- **input device**
- **output device**
- **memory mapped I/O**: 'creates' the address space: some I/O devices are in memory, some are I/O devices

Memories...

- place to hold a value (data or instruction) while it is worked on or until it is needed to be worked on
- may require electrical power to hold a value: volatile memory
- may not require electrical power...: non-volatile memory
- earliest computer memories, magnetic core, were non-volatile

Memories...

- not all memories are the same:
  - smaller access time → higher cost per bit
  - bigger capacity → smaller cost per bit
  - higher capacity → higher access time
Memories...

**REGISTERS:**
- contained within CPU
- GPRs: general purpose registers
- PC: program counter
- PSW: program status word
- access at CPU speed
- volatile

**CACHE:**
- small amount of high-speed mem between CPU and bus
- may be multi-staged
- volatile

**MAIN MEMORY:**
- byte addressable
- flat, linear address space
- typical sizes: 256 Mb to 2 Gb
- volatile

**DISKS (magnetic, optical):**
- are peripheral devices
- interact through interfaces/ controllers
- magnets: now in 100s of Gb
- opticals: now in 15s of Gb
- mechanical access method (e.g., read/write head must be moved to correct position)
- non-volatile
- magnets subject to magnetic fields

**MAGNETIC TAPE:**
- are peripheral devices
- interact through interfaces/ controllers
- now in 100s of Gb
- mechanical access method: tape must be moved to correct position)
- **sequential access only**
- non-volatile
- magnets subject to magnetic fields
Choreography

- so you have this hardware
  - CPU, cache, bus, memory, I/O devices...
  - do you have a server?

Memory Is Not Enough

- does everything needed by computer fit into main memory?
- what if it doesn’t?

Virtual Memory

- memory usually organized as set of pages or segments
  - or combinations of these
- part of program being run placed into memory
  - don’t need it all there at once
  - keep other parts elsewhere (somewhere cheaper)
- meaning?

Virtual Memory

- use of disk space to provide additional memory space
- tradeoffs:
  - memory pricing: $130/Gb
  - disk pricing: $0.46/Gb
  - memory avg access: 10s of ns
  - disk avg access: 13 ms (13,000,000 ns)
- when program needs a page not resident, must wait > 10^6 memory-times for it
- can be twice this bad
Virtual Memory

- incidence of these page faults not reliably predictable
- so bad news for real-time applications
- is this still a big deal?
  - 32-bit processors can only access 4 Gb anyway, so for $550 can have all RAM
  - what about 64-bit processors? (can access 17,179,869,184 Gb)
- bottom line: these days, use RAM to span entire address space for 32-bit machines

Running a Program

- a running program = process
- what resources does a process need?
  - cpu
  - memory
  - devices
- once “running,” does a process run non-stop until done?
  - what happens if it doesn’t?

Running a Program

- OS needs to switch from the context of the waiting process to one that is runnable
- context switches are $5 overhead
- other context switches?

Running a Program

- devices are not directly accessible to processes
- accessed via system calls
  - ask the OS to work on process’ behalf
- system call = context switch
  - at least two
- what does process do while waiting for system call to return?

Life of a Process

Process Administration

- to manage processes, OS needs to keep information about the proc
  - size of memory used
    - context info (suspend/resume)
    - others?
  - where does OS keep all this?
    - process control block (PCB)
  - each new process requires creating a new PCB and initializing it
Workload on Server

- server is a process running on a computer
- process has certain resources:
  - CPU
  - memory
  - I/O access
- process runs instructions along exactly one execution path

Another view of a Process

- distinct:
  - process has
  - process does
- the term “task” often used to refer to what the process has
- look at the “does” part...

Another view of a Process

- suppose we can watch the process as it wanders through memory:
  - imagine we can leave a dot at each location the process visits for instructions:

Another view of a Process

- and if we connect the dots, we get a stringy line tracing the execution of the process: a thread

- "normal" processes we’re used to have exactly 1 thread of execution
- can we have > 1?

Threads, Processes

- can we have > 1?
- YES
  - but what do we need to make that work?

- each thread needs to have:
  - a PC value
  - a PSW value
  - a GPR set
  - tiny amount of stack space
  - a TID: thread ID

  overhead is lighter weight than conventional process

Multi-threaded Processes

- now can have multiple threads of execution in the same memory space
Multi-threaded Processes

- now can have multiple threads of execution in the same memory space
- so all variables are shared across all threads in the process:
  - there is only one copy of the data

- great for sharing data between different threads
  - different threads within a process may do different things but need to exchange info

- fraught with peril
  - what 1 thread does to data affects every thread using that data

Threads

- so a thread is a unit of execution within a process
- where do threads come from?
  - created by parent threads
  - a server may need to invoke a process to handle a client request
    - e.g., a database lookup
    - e.g., perform some computation with client data to produce result for response
  - or, perhaps, create a new thread to handle request

Thread Control: a User Issue

- how expensive is a thread-switch compared to a process-switch?
  - not all threads are created equal
    - user-level threads
      - kernel doesn’t know they exist
      - user process controls creation/switching between threads
      - problem when thread does I/O or makes syscall
    - kernel-level threads
      - kernel does know they exist
      - kernel controls creation/switching between threads

Where the Threads are...

- kernels need not know anything about threads
  - all the thread support exists in a set of library functions, e.g., pthreads
  - these threads often called user-level threads
  - disadvantage:
    - when thread blocks, all threads block
    - can’t really benefit from multiple CPUs if avbl
    - advantages?
User Level Threads

- advantages:
  1. thread switching is very fast, happens entirely within user process/application, no mode changes
  2. thread scheduling handled by user/application so can implement scheduling model best suited to app
  3. library supported, so highly portable

Where Else the Threads are

- move all thread support into kernel: **kernel level threads**
- no thread management code in application
- app uses API to kernel thread functions
  - e.g., linux, Win2K
- kernel schedules individual threads
  - thread-level, not process-level, scheduling granularity

Kernel level threads

- advantages:
  1. kernel can schedule multiple threads concurrently on multiple CPUs if avbl
  2. blocking one thread need not impact processing of other threads in the process
- disadvantages?
  - thread switching involves kernel intervention, mode changes

Thread Performance

- how do user-level/kernel-level threads compare
  - with each other?
  - with use of separate processes?
- measurements reported by Anderson et. al. (1992)
  - on a VAX where a procedure invocation takes 7μs
  - kernel trap takes 17 μs
  - tested:
    - null-fork: time to create child thread/process that, once created does nothing except exit,
    - signal-wait: time for a thread/process to signal a waiting thread/process and wait for an event

<table>
<thead>
<tr>
<th>TEST</th>
<th>USER-LEVEL THREAWS</th>
<th>KERNEL-LEVEL THREADS</th>
<th>PROCESSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>null-fork</td>
<td>34</td>
<td>948</td>
<td>11300</td>
</tr>
<tr>
<td>signal-wait</td>
<td>37</td>
<td>441</td>
<td>1840</td>
</tr>
</tbody>
</table>

Server Performance

- how do servers handle requests?
**Server Performance**

- how do servers handle requests?
  - queue up and process sequentially through single server instance

**Server Performance**

- how do servers handle requests?
  - queue up and process sequentially through single server instance
  - server makes copy of itself to handle a received request
  - i.e., a new server process instance for that request

```
server
server
server
server
```

- bottom line: go for multi-threaded servers to best realize performance on given platform

**Simple Interaction Example**

- http request
- http response
- http server
- db server

**Interaction Example**

- http request
- http response
- http server
- db server
Role of Middleware

- middleware server knows what computations to do to get the data and then how to generate the required HTML to return to client.

Interaction Example

- http request
- http response
- db server

For next time...

- Assignment 1 is available on the course web site
- it is due 07 Sep 2005
- see instructions on course web site for how to submit