JMU CS Systems Track Proposal

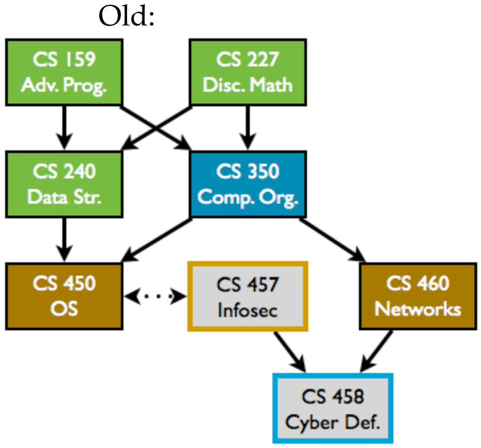
Overview and general proposal

Our overall intent is to provide a systems track that balances breadth of coverage with technical depth. In the existing structure of CS 350, CS 450, and CS 460, all students are required to study three different types of systems in detail. However, unifying themes and principles are insufficiently stressed, and many core and emerging areas of CS are not addressed. To address this, we propose the following course structure:

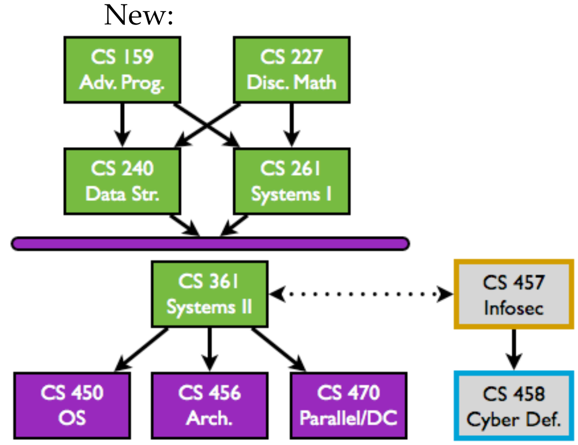
	Course	Description
Systems Core (all required)	CS 261 Computer Systems I	Computer organization, binary data representation, assembly language, multithreading, interrupts
	CS 361 Computer Systems II	State modeling, information exchange, synchronization and deadlock, protocol design, 5-layer Internet model
Advanced Systems (pick one of three)	CS 450 OS	Process management, system call interface, virtual memory, I/O and file systems, virtual machines, security
	CS 456 Computer Architecture	CPU design, ISA implementation, memory subsystem, instruction-level parallelism, parallel architectures
	CS 470 Parallel and Distributed Systems	Parallel/distributed paradigms,parallel software patterns, distributed file systems, performance considerations

In the current model, CS 350, CS 450, and CS 460 are offered only once per year, making them high-stakes courses. CS 350 is particularly bad, as it is a prerequisite for the others. In the new model, CS 261 and CS 361 would be offered every semester. The advanced systems courses would be offered on a rotating basis as needed. This illustration shows the differences in the prerequisite structures:

Old:



New:



The prerequisite depth does not change, nor does the time to the security courses. The following tables illustrate sample plans of study under the old and new models. Under the new model, two sample plans are shown:

Old Model Sample Plan

Year	Fall	Spring
1	CS 139/149 Calculus One GenEd C1 GWRIT 103/GenEd	CS 159 CS 227 One GenEd C1 GWRIT 103/GenEd
2	CS 240 CS 228 CS 260 Optional Calculus One/two GenEd	CS 345 CS 350 Stats Two GenEd
3	CS 450 CS 460 Minor, GenEd, Electives	CS 430 CS 474 Minor, GenEd, Electives
4	CS Electives Minor, GenEd, Electives	CS Electives Minor, GenEd, Electives

New Model Sample Plan 1

Year	Fall	Spring
1	CS 139/149 Calculus One GenEd C1 GWRIT 103/GenEd	CS 159 CS 227 One GenEd C1 GWRIT 103/GenEd
2	CS 240 CS 228 CS 261 Optional Calculus One/two GenEd	CS 345 CS 361 Stats Two GenEd
3	CS 260 Advanced systems Minor, GenEd, Electives	CS 430 CS 474 Minor, GenEd, Electives
4	CS Electives Minor, GenEd, Electives	CS Electives Minor, GenEd, Electives

New Model Sample Plan 2

Year	Fall	Spring
1	CS 139/149 Calculus One GenEd C1 GWRIT 103/GenEd	CS 159 CS 227 One GenEd C1 GWRIT 103/GenEd
2	CS 240 CS 228 CS 260 Optional Calculus One/two GenEd	CS 345 CS 261 Stats Two GenEd
3	CS 361 CS Electives Minor, GenEd, Electives	CS 430 CS 474 Advanced systems Minor, GenEd, Electives
4	CS Electives Minor, GenEd, Electives	CS Electives Minor, GenEd, Electives

A note on transfer students:
Under the current system, VCCS students are advised to complete the following courses prior to enrolling in JMU CS:

- CSC 201 (CS 139)
- CSC 202 (CS 240)
- Math 157 (Math 220)
- Math 173/175/270/271 (Math 235/205)
- Math 286/287 (CS/Math 227)
- ITP 220 (CS 159)

The current documentation indicates students completing this work "are enrolled as juniors and are able to complete the program in two years." To meet this completion time, students must take CS 350 in Spring of their junior year, as well as CS 450 and CS 460 in Fall of their senior year. Failure to pass these courses with a "C-" or better the first time around delays graduation. Under the new structure, these students could complete CS 261 in Fall of their junior year and CS 361 in Spring of the same year. They then have three semesters in which they can complete the Advanced Systems requirement (starting concurrently with CS 361). Thus, this new structure significantly improves the possibility for a VCCS transfer student to graduate on time.

Systems Course Descriptions

Systems core required courses

CS 261 (required)
Introduction to operation of modern interrupt-driven computer systems. Explores the representation of software and information in binary memory, the primary components of a CPU, multithreaded programming, and basic interactions with an Operating System.
Prerequisites: Grade of "C-" or better in CS 159.

Module	Hours	Description
C basics	6	Memory model, pointers
Compiler and debugger use	3	GCC/clang, Makefiles, GDB
CPU/memory organization	3	Registers and cache, locality
Binary representation	4.5	Two's complement, IEEE 754, arithmetic encoding
von Neumann cycle	3	Role of CPU and memory, load/store instructions
Basic circuits	6	Logic gates, adders, ALUs, control signals
Threads vs. processes	6	Fork vs. thread creation, unique memory space
Interrupts and OS principles	4.5	Interrupts, system calls, user vs. kernel mode
System software design, evaluation	4.5	Benchmarks, complexity, static/dynamic analysis

CS 361 (required)
Intermediate exploration of modern interrupt-driven computer systems. Explores models of computation and complex systems, techniques for communication and synchronization of parallel and concurrent software, and the protocols that make up the Internet. *Prerequisites: Grades of "C-" or better in CS 240 and CS 261.*

Module	Hours	Description
Architecture analysis, evaluation, design	3	P2P vs. client-server, layered architecture
State models	4.5	Notion of state, UML, FSM
Mathematical modeling	3	Basic systems theory
Information exchange	6	Communication basics (blocking vs. non-blocking, IPC vs. sockets)
Synchronization primitives and problems	6	Locks vs. semaphores, producer-consumer, readers-writers, dining philosophers, deadlock
Parallel decomposition	3	Data vs. task parallelism, Amdahl's law, fork-join pattern, libraries
Protocol analysis, evaluation, design	9	Protocols and services, timing and statechart diagrams, connections, push/pull, flow control, reliability, handshaking, metrics
The Internet model	4.5	HTTP, DNS, DHCP, TCP, UDP, IP, 802.3, 802.11, ARP

<u>Advanced systems courses (pick 1)</u>

CS 450 Operating Systems

Introduction to the design and implementation of modern operating systems. Explores fundamental concepts of operating systems, memory management, virtualization, resource allocation, file systems, and system protection mechanisms. Course work includes a significant programming component. *Prerequisites: Grade of "C-" or better in CS 361.*

Module	Hours	Description
Thread and process management	4.5	Review of multithreading, OS structures for representing threads and processes, context switches
OS interface and IPC	4.5	Review of system calls vs. interrupts, pipes, shared memory, other forms of IPC
Synchronization implementation	6	Hardware support for implementation of semaphores, locks, spinlocks
Memory management	6	Paging vs. segmentation, virtual memory, demand paging, implementation of shared memory
Virtualization	6	Virtualization vs. emulation, trap-and-emulate, binary translation, hardware support for virtualization
Scheduling	3	Scheduling policies and evaluation
I/O and file systems	6	Interrupt-driven I/O, DMA, RAID, file system implementation and metadata
Security and protection	6	CIA model, access control mechanisms, malware defense mechanisms

CS 456 Computer Architecture

Introduction to the design and implementation of modern CPU architectures. Explores hardware-based parallel execution, quantitative performance evaluation, I/O interfacing techniques, and hardware descriptor languages. Course work includes a significant programming component. *Prerequisites: Grade of "C-" or better in CS 261.*

Module	Hours	Description
Assembly language	6	RISC assembly language and decoding
Building a datapath	6	Logic gates, control unit, ALU construction, register banks, von Neumann implementation
Hardware descriptor languages	3	Verilog, VHDL, RTL
Pipelined datapath and hazards	6	Pipelined datapath and control, data hazards (forwarding vs. stalling), control hazards, exceptions
Memory hierarchy and cache design	6	Quantitative performance measures, cache mapping techniques, cache coherence protocols
Storage and I/O interfacing	4.5	Storage devices, bus protocols, I/O performance
Instruction-level parallelism	4.5	Branch prediction, dynamic scheduling
Data-level parallel architectures	3	Vector, SIMD, GPU architectures
Thread-level parallel techniques	3	Hyperthreading, shared-memory multiprocessors

CS 470 Parallel and Distributed Systems

Introduction to parallel and distributed systems. Explores shared memory, cluster, grid, peer-to-peer, and cloud computing models along with parallel software patterns, distributed file systems, and performance considerations. Course work includes a significant programming component. *Prerequisites: Grade of "C-" or better in CS 361.*

Module	Hours	Description
Parallel/distributed concepts	3	Amdahl's law, critical paths, speedup/scalability, data/task decomposition, applications, research challenges
Parallel patterns	6	Naturally (embarrasingly) parallel, nearest-neighbor, communication, producer-consumer, master-workers, pipelines, map/reduce
Parallel systems	9	Shared memory, SMP, SIMD, OpenMP, GPUs/co-processors, race conditions, mutual exclusion, deadlock, cache effects, dense/sparse matrices
Distributed systems	9	MPI, MapReduce, global address spaces, clusters, topologies, synchronization, collectives, clocks, NUMA, hybrid architectures, fault tolerance
Grid, P2P, and cloud computing systems	6	Heterogeneous systems, decentralized computation, consensus, IaaS, virtualization
Distributed file systems	6	RPC, data replication, transactions, consistency
Parallel performance	3	Tools, measurement, scheduling/load balancing, contention, communication overhead, power usage

Future advanced systems course

In a future proposal, we will introduce the following course as an additional advanced systems course option. At the present time, we are not including it because there is uncertainty regarding similar material between this course and CS 361.

CS 466 Networking Internals
In-depth exploration of layered networking protocols. Explores the internal operation of protocols for routing, packet delivery, end-to-end reliability, and network-based applications, as well as security concerns. Course work includes a significant programming component.
Prerequisites: Grades of "C-" or better in CS 361.

Module	Hours	Description
Network address mapping	7.5	ARP, IP packet fragmentation, ICMP
Network routing	4.5	Routing and routing security
Transport layer algorithms	3	Go-back-N, selective-repeat
TCP	6	Error detection, congestion control, TCP FSM
Distributed file systems	4.5	P2P vs. client-server
Host configuration and DNS	4.5	DNS and host configuration security
File transfer applications	3	SSH, FTP
Network management applications	6	Network management and security
IPv6 transition	3	Addressing, packet format, transition from IPv4

Transition and Implementation Plan

Students completing CS 350-CS 450-CS 460 by Fall 2014: These students will be unaffected.

Students who would take CS 350 starting Spring 2016: These students will be transitioned to take CS 261 in Spring 2016. They can then take CS 361 Fall 2016, along with either CS 432 or CS 456. Note that we would need to offer one of those two that semester. Otherwise, they may need to overload their Spring 2017 schedule with a systems course in addition to CS 430 and CS 474:

	Existing plan		New plan	
	Fall	**Spring**	**Fall**	**Spring**
2015-2016	CS 240 CS 228	CS 345 CS 350	CS 240 CS 228	CS 345 CS 261
2016-2017	CS 450 CS 460	CS 430 CS 474	CS 361 CS 450	CS 430 CS 474

Students who will take CS 350 in Spring 2015: As in the previous case, we could replace CS 450 and CS 460 in Fall 2015 with CS 361 and CS 456. The problem with this is that these students will miss out on some Core Tier-1 material: threads vs. processes, OS principles, and interrupts, which they would get in the current CS 450. To remedy this, we propose these students take CS 361 and the current version of CS 450. That is, CS 361 will count as a substitute for CS 460.

Students who are taking CS 450 or CS 460, but not both in Fall 2014: If they are currently in CS 450, as before, we could count CS 361 as a substitute for CS 460. As they would not be able to take CS 460 until Fall 2015, taking CS 361 at that time would be sufficient. If they are currently in CS 460, they would need CS 450. To accommodate this, a Fall 2015 offering of the current CS 450 would fit this need.

Summary: Fall 2015 would see the replacement of CS 460 with CS 361; CS 450 would be unchanged. Spring 2016 would see the replacement of CS 350 with CS 261. For those students, we will offer CS 361 in Fall 2016 (4 sections) to prevent delay in their schedule. We would also offer two sections of CS 450 for those students who would like to take it that semester. They could take one of three CS 470 or CS 456 sections in Spring 2017 instead. By this point, we can start the regular offering of 2 sections of CS 261 and CS 361 every semester, with 2-3 advanced systems sections pending demand.

	Existing plan		New plan	
	Fall	**Spring**	**Fall**	**Spring**
2015-2016	CS 450 (4) CS 460 (4)	CS 350 (4)	CS 450 [old] (4) CS 361 (4)	CS 261 (4)
2016-2017	CS 450 (4) CS 460 (4)	CS 350 (4)	CS 261 (2) CS 361 (4) CS 450 (2)	CS 261 (2) CS 361 (2) CS 456/470 (2+1)
Steady state			CS 261 (2) CS 361 (2) CS 450/466 (2+1)	CS 261 (2) CS 361 (2) CS 456/470 (2+1)

Sample Faculty Scheduling (for illustration only)

Course	Sections	Instructors
139	5	Mayfield(2), Rahman(2), Harris(1)
149	3	Harris(2), Norton(1)
159	4	Norton(2), Grove(2)
228	3	Mata(3)
240	4	Sprague(2), Lam(2)
260	3	Sorge-Way(2), Henriksen(1)
261	2	Aboutabl(2)
330	1	Staff
345	3	Fox(2), Bernstein(1)
361	2	Kirpatrick(2)
347	1	Grove
349	1	Bernstein
444	1	Sprague
447	1	Frysinger
450	2	Tjaden(2)
457	1	Aboutabl
480	1	Buchholz
512-D	1	Heydari
515-D	1	Wang
560-D/610	1	Tjaden
660-D	1	Wang
685	1	Wang
550	1	Kirkpatrick
552	1	Heydari
Sec/Prog	1	Bernstein
640/640-D	1	Buchholz
	47	

Course	Sections	Instructors
139	3	Harris(3)
227	3	Mata(3)
228	1	Wang(1)
159	5	Mayfield(1), Sprague(2), Norton(2)
240	2	Rahman(2)
260	1	Staff
344	1	Grove
345	2	Bernstein(2)
261	2	Kirkpatrick(2)
354	1	Sprague
361	2	Simmons (2)
430	4	Grove(2), Fox(2)
456	2	Aboutabl(2)
458	1	Wang
462	1	Bernstein
470	1	Lam
474	4	Mayfield(2), Norton(1), Rahman(1)
482	1	Lam
633/633-D	1	Buchholz
625-D/635	1	Tjaden
557	1	Aboutabl
630	1	Lam
550-D	1	Heydari
652-D	1	Heydari
627-D	1	Wang
555-D	1	Staff
685	1	Buchholz
685	1	Kirkpatrick
	49	

Other Impacts

Telecom minor: We have informed ISAT of our intention to de-crosslist CS/ISAT 460. Based on current enrollments, there will be a demand for one section of ISAT 460, which will continue to be required for the telecom minor. This section would be staffed by ISAT faculty, including Emil Salib and Samy. The ISAT AUH and program chairs are aware of this staffing requirement and have accepted it.

CS minor: The current CS minor requirements are as follows:

- CS 139/149 and CS 159
- Four of:
    - CS 228, CS 240
    - Any CS course 300-level or above

The new minor requirements would be as follows:

- CS 139/149 and CS 159
- Four of:
    - CS 228, CS 240, CS 261
    - Any CS course 300-level or above

Information security certificate: The current requirements are as follows:

- Completion of CS major (including CS 450 and CS 460)
- CS 457
- CS 458
- One of:
    - CS 482, CS 461, CS 462, CS 463, pending approval of the section offered

The concern here is whether replacing CS 350, CS 450, and CS 460 with CS 261 and CS 361 provides sufficient mapping to the NSTISS 4011 requirements. The relevant portion of the standard is Section V, 14(b) AUTOMATED INFORMATION SYSTEMS (AIS) BASICS. As these require only awareness level, these mappings are sufficient. To remove concerns, we propose that the certificate requirements be adjusted as:

- Completion of CS major, using CS 450 to complete the advanced systems course option
- CS 457
- CS 458
- One of:
    - CS 482, CS 461, CS 462, CS 463

Graduate program instructional staffing: The tension with graduate course staffing relates to the "one-off" sections of the 3-section, 2-prep structure. The challenge is that every graduate course, with 9 per semester, is a one-off course. Of those, 5-6 per semester have been covered by Hossain, Steve, and Florian. At the undergraduate level, we have four in the fall (347, 349, 444, 457) and four in the spring (344, 354, 458, 462) that are regularly offered, in addition to special topics. When we omit the sections taught by Hossain, Steve, and Florian, there are 8-9 one-offs per semester to be covered by 14 faculty members. Adding one additional one-off (to accommodate the 2+1 structure of the advanced systems offering) is feasible. Furthermore, as these courses can also be used as electives, the offering schedule of other electives could be amended as needed. Another possibility would be to treat a joint 450-550 offering as 2-sections, 1 prep. Such a move would only increase the one-off count by 1 instead of 2 per year.

Repeat-forgive option: One concern with the scheduling of the new advanced systems courses is that some students would need to do a repeat-forgive for these courses. If they are not sufficiently offered, these students may have difficulty exercising this option. To mitigate this risk, the group recommends offering each advanced systems course once per year.

There is also a one-time concern for students taking CS 460 this semester. If they fail this semester, they will not be able to use a repeat-forgive by taking CS 361 instead. To address this, we propose delaying the de-cross-listing of ISAT/CS 460 until after next fall, but placing a registration restriction that CS majors cannot register for the course. That way, if they fail CS 460 this semester, they could be given an override to take that version.

Textbook selection: One potential difficulty with the Systems I-II structure is the selection of an appropriate textbook. There are several options, such as *Computer Systems: A Programmer's Perspective* by Bryant & O'Halloran, that are moderately good foundations. However, supplemental reading material, such as tutorials, would probably be necessary to complete the coverage necessary. The advantage of the Bryant & O'Halloran book would be that it could be used for both semesters, thus reducing textbook costs for students.

ACM 2013 KA		Current	Proposed
Algorithms and Complexity			
	Distributed algorithms (T1)	none	470
Architecture and Organization			
	Machine-level representation of data (T2)	350	261, 456
	Assembly level machine organization (T2)	350	261, 456
	Interfacing and I/O strategies (interrupts) (T2)	350	456
	Memory architecture (T2)	350	261, 456
	Functional organization (ILP/datapaths) (E)	350 (some)	261, 456
Operating Systems			
	OS overview (T1)	450	261
	OS principles (APIs, processes, interrupts) (T1)	450	261, 450
	Concurrency (T2)	450	361, 450
	Scheduling (T2)	450	450
	Memory management (T2)	450	450
	Security and protection (T2)	450	457/450
	File systems (E)	450 (some)	450
	System performance evaluation (E)	none	261
Network-centric Computing			
	Introduction (ISPs, circuit vs. packet) (T1)	460	361
	Network applications (HTTP, sockets) (T1)	460 (most)	361, 466, 470
	Reliable data delivery (TCP, flow control) (T2)	460	361, 466
	Routing vs. forwarding (T2)	460	361, 466
	LANs (T2)	460	361, 466
	Resource allocation (T2)	460	361, 466
	Mobility (T2)	460	361, 466
Parallel and Distributed Computing			
	Parallelism fundamentals (T1)	450 (most)	261, 470
	Parallel decomposition (T1,T2)	none	361, 470
	Communication and coordination (T1,T2)	450 (some)	361, 450, 470
	Parallel architectures (T1,T2)	350 (little)	261, 470
	Parallel performance (E)	none	470
	Distributed systems (E)	none	361, 470
System Fundamentals			
	Computational paradigms (T1)	350 (some)	261, 361, 456, 470
	Cross-layer communications (T1)	450 (little)	361, 450, 466, 470
	States, transitions, state machines (T1)	350 (some)	361, 456, 466, 470
	System support for parallelism (T1)	350 (little)	361, 456, 470
	Performance (T1)	none	261
	Resource allocation and scheduling (T2)	450	450
	Proximity (T2)	350	450, 456, 470
	Virtualization (T2)	none	450
	Reliability through redundancy (T2)	460 (little)	361, 456, 470

Systems Core Principles and Vision Statement

To reflect trends within the CS field, the JMU CS systems core courses should strive to produce the following characteristics of CS graduates:

- technical understanding of computer and network systems
- appreciation of the interplay between theory and practice
- system-level perspective (thinking in levels of abstraction)
- understanding of how to identify and evaluate trade-offs in design and implementation
- ability to identify common problem patterns and apply appropriate solutions
- demonstrable experience with large software projects
- commitment to individual skill development, such as learning new languages
- commitment to professional responsibility
- understand the differences between systems and application programming
- understand the hardware characteristics that dictate the requirements/features of the controlling software (*e.g.,* sampling frequency, signal propagation delays, arithmetic precision)

In support of this goal, the following principles should guide the systems curriculum:

- open-ended programming assignments that require analysis, design, and testing
- opportunities to apply systems concepts to realistic problems
- a combination of individual- and group-based projects
- experience both reading and writing code
- exposure to standard industry tools and techniques
- exploration of the design considerations underlying existing systems software
- emphasis on developing skills for independent learning
- flexible curriculum that serves the needs of students with varying technical talents
- appropriate coverage of fundamental computer and networking concepts

<u>Systems Core Vision Statement:</u>

Every JMU CS student who satisfactorily completes all required systems courses should be able to:

1. Summarize the technical foundations of how software executes on hardware, how parallel and distributed software communicate, what patterns and structures are used to construct systems and concurrent programs, and what layered abstractions support modern computing systems.
2. Explain how bits can represent information, how instructions and communication can be seen as a sequence of state transitions, how mathematical models can describe system behavior, and how reactive programming practices used in systems programming differs from other approaches.
3. Read and understand existing protocols, critically evaluate existing protocols, and select appropriate protocols.

4. Describe the ways in which information can be exchanged between different parts of an existing system, select appropriate ways to exchange information in proposed systems, and implement systems that exchange information.
5. Describe the architectural style/high-level design of a system using appropriate terminology, and evaluate the architectural style/high-level design of existing and proposed systems.
6. Describe the architecture of a given computer system and design one that conforms to such architecture. This covers design of the CPU, the Main Memory, as well as performance-improvement approaches (e.g. cache, pipelines, etc.)
7. Identify sources of confusion when exposed to large problems, analyze what information is needed to design a solution, and select appropriate references to overcome these challenges.
8. Analyze compiler warnings and errors, and apply this information to fix problems in their own source code.
9. Read and understand source code for a known problem, critically evaluate the design choices, and modify the code to solve a new problem.
10. Demonstrate the ability to work collaboratively to develop robust software prototypes that illustrate how computer and network systems operate.
11. Understand the difference between deterministic and stochastic systems and be able to build and use simple models of these systems.
12. Explain the difference between safety and security and understand the systemic nature of safety and security.