**TIM-8120 Distributed Systems**

**(3 credits)**

**Course Description:**

Distributed computing is becoming more commonplace as cloud and mobile technologies, super-speed wired and wireless connection interactions continue to evolve. In this course you will focus on strategies to coordinate information technology efforts over vast distances. You will study the emerging trends, tools and technologies that allow systems working from different geographic locations, platforms or configurations to integrate and work as a single system. You will apply these strategies and techniques to effectively manage a distributed computing platform.

**Number of Assignments: 8**

**Learning Outcomes:**

1. Adapt traditional computer science theories to distributed and massively parallel processing environments.
2. Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
3. Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
4. Design solutions to computer problems using distributed systems.

**Course Concepts:**

1. Fundamentals of Distributed Systems
2. Hardware and Software Differences in Distributed Systems
3. Sharing Resources
4. Solving Problems Using Concurrent Processes
5. Communications in a Distributed System
6. Applications of Distributed Computing to Solve Problems

**Primary Resource/textbook:** **N/A**

**Course Overview**

**Section 1: Fundamentals of Distributed Computing**

**Week 1: Distributed Systems Architecture: Hardware**

Week 1 Assignment: Create a Conceptual Diagram of a Basic Distributed System (10 Points)

**Week 2: Distributed Systems Architecture: Software**

Week 2 Assignment: Plan for a Distributed Software Architecture in a Distributed System (10 Points)

**Section 2: Shared Resources in Distributed Computing**

**Week 3: Network Communication Mechanisms**

Week 3 Assignment: Compare Network Protocols Applied to a Distributed System Configuration (10 Points)

**Week 4: Shared Objects and Processes**

Week 4 Assignment: Compare Network and Node File Systems and Apply to a Distributed System Configuration (10 Points)

**Week 5: Shared Memory and Naming**

Week 5 Assignment: Create a Network Map Diagram for Naming Schemes and Shared Memory Management (10 Points)

**Section 3: Concurrency and Applications**

**Week 6: Concurrent Processing**

Week 6 Assignment: Research Clock Synchronization Models for Processor Concurrency (15 Points)

**Section 4: Network Communications, Resource Allocation, and Control in Applications**

**Week 7: Data Structures and Fault Tolerance**

Week 7 Assignment: Design a Tree Structure for File and Data Allocation and Implement a Fault Tolerance Technique (10 Points)

**Week 8: Distributed Systems Resiliency and Security**

Week 8 Signature Assignment: Enhance a Distributed Systems Architecture for an Enterprise to Solve Potential Problems (25 Points)

**Section 1: Fundamentals of Distributed Computing**

Distributed systems have now become a common fixture in domestic, commercial, and professional fields. With the introduction of desktop computing to the household in the late 1970s and early 1980s, the introduction of the open systems interconnection (OSI) model marked a transformation of information systems. Until then, the corporate environment operated in a centralized infrastructure, or mainframe. But it quickly started evolving, first with the concept of downsizing, and later rightsizing (1990s). The Internet and its resources are viewed as the global environment in which the computations occur. As higher-level protocols and standards are developed, the lower-level couplings that might make distributed computing difficult are becoming less consequential.

The rapidly evolving network and computer technologies have led to new possibilities in almost every area of society. Wide area networks are as fast as local area networks of just a few years ago, and they continue to increase in speed. Computer processors have increased in speed and ability to handle multiple processes to such a level that systems are converging into a modern mainframe-like system called cloud computing. The challenge of distributed system technology is to provide flexible and reliable infrastructures for such large-scale systems that meet the demands of developers, users, and service providers.

The first classification for distributed networks are local area networks and wide area networks. Local area networks (LAN), while being on a local geographic location, allow multiple nodes or devices to interconnect and share resources. Data is transmitted in small packets at a transfer rate of bits per second (bps). Typically there are two major classifications for LANs:

1. Peer-to-Peer Network: Each node is independent but shares data and resources across the network. The nodes are connected all the time to determine the path for message transmission and package routing.
2. Client/Server Systems: All users and resources (nodes) are distributed in different servers at different geographic locations. They are authenticated through a domain controller.

Wide area networks (WAN) and cloud networks allows millions of nodes from multiple geographic locations to share resources, as well as route and send packages of messages. The future of distributed systems might be ubiquitous computing in the conversion of the Internet and mobile computing. Imagine millions or even billions of devices working together in a worldwide distributed system several orders of magnitude larger than today’s Internet.

Mobile and cloud computing have transformed the way network distributed infrastructures are designed, configured, implemented, and managed. Van Steen and Tannenbaum (2017) define a distributed system as a collection of computers, devices, software, and services—often referred to as nodes—that handle and store service requests, queries, processes, and storage in different computers located in different geographic locations, but giving the impression that it is handed by only one computer in one single location (transparent). Distributed systems and distributed computing have become common models for computer center installations in all enterprises, government institutions, organizations, and academia.

Anthony (2015) defines a distributed system as one where an unknown failure in an unknown device can render a system unusable. With this risk identified, a distributed system is an arrangement of interconnected nodes that runs multiple processes at multiple geographic locations (Anthony, 2015).

Samples of distributed systems could be in-house corporate systems, enterprise resource planning (ERP) package implemented in different plants, electronic health patient records for a hospital chain with presence in various countries, grid and cloud computing, e-commerce, e-government, e-banking, higher education, and many others. These collections of computers, devices, appliances, and services are interconnected and functioning as a single centralized system (for the user, that is) when they are really distributed or scattered in many locations. One example could be social media, where companies providing the servicing for social media portals have servers working in coordination and even redundancy to create a balance in request, queries, and processing queues by having servers distributed all around the globe, but transparent to most users.

A distributed system consists of appropriate hardware and software components that must work in tandem to provide an efficient operating environment. The essence of these systems is that many processors or entities (often called nodes) are active in the system simultaneously. These nodes commonly have autonomous hardware and software. Nevertheless, the nodes often share resources and information to solve a problem by reducing the time to solve it. To ensure that the resources work efficiently, there must be a reduction in the load on each individual node, increase system reliability, and so on. In this section, you will explore further the fundamentals of distributed computing, which include the hardware and software of the system.

Reference:

Anthony, R.J. (2016). Systems Programming: Designing and Developing Distributed Applications. Walthan, MA. Elsevier-Morgan Kaufmann.

Tanenbaum, A. S. & van Steen, M. (2017). Distributed Systems. (3rd ed.). Maarten van Steen Publisher.

**Week 1: Distributed Systems Architecture: Hardware Architecture**

A model is an abstract representation of a functional system. It starts with a high-level brainstorm or original design, referred to as a conceptual model, and later evolves to the physical specifications, location, and arrangements or parameters of nodes, devices, communications, and software.

Distributed systems may be composed of hardware (nodes) of similar or distinct internal architecture and configuration that communicates with other hardware (nodes) in the network that may be significantly different in its internal configuration or architecture, operating system, and Internet connection protocols. These nodes can act independently from each other, but act as a single unit for the users—but with the advantage that if one node fails there is no direct impact on the performance of the rest of the connected nodes.

Distributed systems also feature the capability of load balancing, that is the capability of breaking down messages in small chunks, distributing the load through different channels to leverage processing speed and load, making it faster and safer; if one node fails only the message transmitted through that channel is lost, but it can be reconstructed by the rest distributed by different channels.

Communication is processed by message passing. Message passing consists of breaking down messages in packets and transmitting them across different channels; but to the user, it is transparent. But if one node fails, the message can still be reconstructed and delivered to the destination. This multiple message distribution strategy also leverages transmission load, and it is known as load balancing. The benefits derived from distributed systems or architecture are consistency, replication and caching, fault tolerance, security, and access controls, among other processing and data storing capabilities. This capability of message passing is coordinated by synchronizing each node clock and authentication to prevent message collision or bottlenecks.

There are various factors that can affect the performance of a distributed system. The configuration of the hardware is important because it will determine appropriate factors that need to be considered to create an optimal distributed system. The components that make up the hardware of a distributed system can include the processor, memory, and interconnection network. The types of distributed systems are:

Table 1. Types of distributed systems

|  |  |
| --- | --- |
| **Type** | **Description** |
| Cluster Computing | A cluster is a collection of computers or nodes with similar configuration running the same operating system organized by a central node, sometimes a supercomputer or main server. |
| Grid Computing | Collection of computer nodes with different or dissimilar configuration or from different segments running different operating systems or services, but still interconnected and sharing data, applications, and services. |
| Pervasive System | Variations from grid computing, adding home systems, car systems, patient electronic healthcare records systems, sensor networks, and so on. |
| Cloud | Corporate outsourced model of shared network assets (hardware, software, and services) available on demand from multiple locations or access points. |
| Parallel | Type of computation model where many calculations are performed at the same time but in different devices or nodes. |

Smart devices such as cell phones, TVs, cars, watches, etc. have flooded society. While they are built with specific tasks in mind, they are commonly built with embedded microprocessors and connected wirelessly to a LAN or the rest of the world through the Internet. As the power of these microprocessors increases and power requirements decrease, these special-purpose devices are taking on more and more roles as parts of large-scaled and complex systems. Phones that once were only capable of playing Mine Sweeper are now connected to the rest of the world and are used to navigate, shoot films, and create a whole new paradigm for how software is created and sold.

This week, you will reflect on the hardware differences and how distributed systems have changed hardware and computing services at home and the enterprise. This will allow you to imagine and define a distributed system of your own to solve a problem. You will start by examining how microprocessors are designed and are able to handle processes. Then, you will be able to envision how the future of distributed systems will look.

Be sure to review this week’s resources carefully. You are expected to apply the information from these resources when you prepare your assignments.

**Heads-Up to the Signature Assignment**

Your culminating Signature Assignment (due in Week 8) will be a reflection of all that you have learned within the course, and it may require that you complete some work ahead of time. To ensure you are prepared and have adequate time to complete this assignment, please review the instructions by looking ahead to Week 8. You can contact your professor if you have questions.

**Resources:**

Annadurai, P. & Vijayalakshmi, S. (2016). Context centric cluster computing in AdHoc Network (C4). *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).* 518-522.

##### Celar, S. Mudnic, S. & Seremet, Z. (2016). State of the Art of Messaging for Distributed Computing Systems. *Annals of DAAAM & Proceedings.* 27298-307.

Mousavi, K. & Sharifi, M. (2014). AMRC: An algebraic model for reconfiguration of high performance cluster computing system at Runtime. *Journal of Supercomputing, 67*(1), 1-30.

Rodriguez, J.M., Godoy, D., Mateos, C. & Zunino, A. (2017). A Multicore Computing Approach for Large Scale Multi-Label Classification *Intelligent Data Analysis*. 21(2). 329-352.

Thiele, L. Wirth, T., Olbritch, M. Schierl, T. Haustein, T. & Frascolla, V. (2014). High performance cluster computing as a tool for 4G wireless system development. *Intel Technology Journal, 18*(3), 98-117.

Van Steen, M. & Tanenbaum, A.S. (2016). A Brief Introduction to Distributed Systems. *Computing.* 98(10), 967-1009.

Venkatesan, D. & Sridhar, S. (2017). A Novel Programming Framework for Architecting Next Generation Enterprise Scale Information System. *Information Systems & E-Business Management.* 15(2), 489-534.

Yang, M., Min, G., Yang, W. & Li, Z. (2014). Software rejuvenation in cluster computing systems with dependency between nodes. *Computing, 96*(6), 503-526.

**Software:**

For many of the assignments, you will need to use a network diagraming tool such as Microsoft Visio, Smartdraw, LucidChart, Dia, among others. Here are some suggested tools to use:

* MS-Visio – trial version
* OpenOffice
* LucidChart
* ConceptDraw
* Solarwinds Network Topology Mapper
* Cade Software
* Dia
* Network Notepad
* LanFlow
* EDraw

**Week 1—Assignment: Create a Conceptual Diagram of a Basic Distributed System (10 Points)**

Consider a scenario for a retailer enterprise with different plants and stores distributed across many U.S. states and launching a globalization strategy. This strategic plan includes increasing online presence and expanding their distribution channels. The retailer chain manufactures and distributes three clothing lines that are also marketed in other major retailer chains and online. The company has its headquarters in South Carolina; a major manufacturing plant in Seattle, Washington; another in Maine; plus 30 stores nationwide. They’ll soon be opening three international stores—one each in Spain, England, and New Zealand. They have approximately 3,000 employees. Their computing infrastructure is approximately 500 computers, 200 point-of-sale terminals connected to servers distributed at the headquarters in South Carolina, Seattle, and Maine. There are an estimated 300 printers also distributed and shared across the network.

Part I:

Using diagraming software such as Visio or other diagramming software, create a distributed system model diagram for the proposed corporate scenario. Everyone in the headquarters should have access to the clothing-line catalog, shared files, and resources. Printers, scanners, routers, and switches need to be included, plus the corporate webpage and the corporate intranet for sales, shipping, and processing and internal administrative tasks. Once completed, copy and paste the image to a Word document, and write a paper detailing the advantages and disadvantages of your solution for distributed computing for the company. Submit the original diagram file plus the summary document with your analysis and interpretation of the solution.

Part II:

Vertex, or graph coloring, is often used in scheduling computation tasks on a processor. On modern computer processors, there are multiple cores, and on Intel chips something called “Hyperthreading” that makes each core effectively act like two cores. For this task, you will research how the scheduling is done on a modern Intel Quad Core processor. You will draw a Gantt chart representing 10 processes of varying length executing on the processor, and describe how the scheduling is accomplished, listing the average start time and completion time on the processor.

Length: Network diagram, Gantt chart, and a 5- to 7-page paper where you will present a summary stating the benefits, limitations, and challenges of a distributed system, plus your interpretation.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Upload your document and file, and click the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.

**Week 2: Distributed Systems Architecture: Software**

The organization of a distributed system is as simple or complex as the number of systems, services, or applications being executed and shared across nodes and locations, both internal and external. The software architecture depicts the structure, configuration, and management of software and tools synchronized with the operating system and how they interact across the network and with other nodes.

One of the core goals or objectives of a distributed systems architecture is to segregate applications in a separate layer from the user interface, databases, operating systems, and tools. This segregation across nodes paves the way for faster and improved performance in the execution of applications, while a number of access to each application is easier to manage if applications run in one node and are referenced by the rest who need access to it.

Among the software components that make this interconnection feasible across diverse and similar nodes are:

1. Components: Modular units with very specific contents or services referred. An example of components could be an enterprise resource planning (ERP) implementation, where each core processing area is divided into modules, each with its programs suite and database interconnected between modules. Another example could be views, where the data needed for a particular transaction is shown, preserving the integrity and security of the data managed by the operations and applications.
2. Interfaces: Small scripts or code that facilitates communication or presentation to users across the network.
3. Connectors: Small scripts designed to establish communications and share information across dissimilar networks and applications. Written in low-level programming languages like C, C++, or Java, connectors include code and protocol to establish communication between dissimilar systems, operating systems, and applications; and once that communication is settled, file sharing and processing are executed or transmitted.

Certain styles for software architecture within a distributed system are to be explored this week. Among the most used architectural environments in terms of software processing are:

1. Layered Architecture: In this arrangement, components or modules are organized in layers, where one module executes a call to another and waits for a response with the expected output or processing message. Calls could be downwards or upwards, but the most common is downcall (request-response), or one-way-call. For this software architecture arrangement, the transmission control protocol of the TCP/IP suite intervenes to facilitate the exchange of calls and message parsing across resources.
   1. Application layering structures are distributed or structured in three layers, application interface layer, processing level, and data level.
2. Object-Based Architecture: In this style, each object is defined as a component, and each is connected using a procedure called *mechanism*, permitting encapsulation of data—that is, they can be seen as a single entity. This arrangement makes it possible to run the interface script from one node or machine, and the object or data is located in another.
3. Service-Oriented Architecture (SOA): The services are separated from one another so that they can run independently. In a service-oriented architecture, a distributed application is built as a composition of many different services. These services are encapsulated into a single unit becoming a single interface that can be run by the users. (van Steen, & Tanenbaum, 2017).
   1. With the trends in hardware moving toward smaller devices, the desire for more dynamic distributed systems and services has arisen. Instead of fixed systems, we are looking to design spontaneous systems that add and remove clients regularly. Jini is an old model that is getting new life with the Apache foundation as River. The distributed model in Jini is based on Java allowing any device running a JVM to dynamically participate in the system. Services can be assigned to entities on the network to be used by individuals, programs, and/or other services. Jini and similar systems are better suited than CORBA for dynamic distributed systems where components may be mobile and form spontaneous networks, an important trend in distributed systems when you consider trends in hardware. Techopedia defines Jini as a service-oriented architecture that defines a programming model that exploits and extends Java technology. This enables the construction of secure, distributed systems by building scalable and flexible networks. (Techopedia, 2018).
4. Resource-Based Architecture: In this arrangement, services are compiled and managed as modules or components. These modules or components can be changed, added, or removed from the module or component and identified by using a naming scheme. Messages are described and sent across the nodes using basic commands like PUT, GET, DELETE, POST.
5. Middleware Organization: This arrangement is independent of the overall organization of the distributed system, as it is aligned in a separate processing layer. Two types of patterns in middleware are wrappers (a component of which options are transformed into variables of the component), and interceptor (code that will break the usual flow of execution, and control is transferred to another module for execution). While hardware designs can easily be exploited to work together, it takes special software to actually coordinate distributed problem-solving. Middleware and other software infrastructure components provide the basic communication facilities to overcome the differences in hardware and software that you will find when talking about creating networks of hundreds or thousands of devices. DCE, CORBA, and Jini are a few of the more established software suites used to coordinate heterogeneous systems for distributed problem-solving. Middleware applications have been around for some time, and one of the earlier ones was called DCE. While DCE is old, you can still find it in use today. DCE is based on remote procedure calls (RPC) and provides an array of services, including directory services, security services based on Kerberos, and distributed file services. You will take a closer look at RPC later in the course. But for now, it is enough to know it is used in some of the most popular systems in use today.
   1. CORBA is one of the most widely used and well-known distributed systems in used today. CORBA stands for common object request broker architecture and is supported by a consortium of industry professionals. Where it differs from DCE is that is it based on object-oriented concepts. Because CORBA is maintained by a consortium, it remains current and open. However, static distributed systems were the model when CORBA was designed. These systems require considerable resources while running and retain the classic client/server model as its basic design.

Reference:

Techopedia. (n.d.) Definition of Jini.

Be sure to review this week's resources carefully. You are expected to apply the information from these resources when you prepare your assignments.

**Resources:**

Chae, B., Olson, D., & Sheu, C. (2014). The Impact of Supply Chain Analytics on Operational Performance: A Resource-Based View. *International Journal of Production Research. 52*(16), 4695-4710.

Hannay, J. E., & Mevassvik, O. M. (2017). A Hybrid Architecture Framework for Simulations in a Service-Oriented Environment. *Systems Engineering. 20*(3), 235-256.

Li, X. & Madnick, S. (2015). Understanding Dynamics of Service-Oriented Architecture and Implementation. *Journal of Management Information Systems.* 32(2), 104-133. Business Source Complete EBSCOhost.

Martinez, A., Vazquez, B. Estrada, H., Santillan, L. & Zavala, C. (2017). Incorporating Technology in Service Oriented i\* Business Models: A Case Study. *Information Systems & E-Business Management.* 15(2), 461-487.

Peng, Q (2018). Multimedia Teaching Platform Construction for Fashion Design Based on Simulation and Synchronous Teaching System. *International Journal of Emerging Technologies in Learning.* 13(5) 212-223.

Perren, R. & Kozinets, R.V. (2018). Lateral Exchange Markets: How Social Platforms Operate in a Networked Economy. *Journal of Marketing.* 82(1), 20-36.

Sandkuhl, K. Lin, F., Shilov, N., Smirnov, A. Tarasov, V. & Krizhanovsky, A. (2013). Logistics-As-A-Service: Ontology-based Architecture and Approach. *Investigacion Operacional.* 34(3), 188-194.

Syssner, J. & Meijer, M. (2017). Informal Planning in Depopulating Rural Areas: A Resource-Based View on Informal Planning Practices. *European Countryside.* 9(3), 458-472.

Tanenbaum, A. S. & van Steen, M. (2017). Distributed Systems. (3rd ed.). Maarten van Steen Publisher.

Vanosch, W. & Steinfield, C. (2018). Strategic Visibility in Enterprise Social Media: Implications for Network Formation and Boundary Spanning. *Journal of Management Information Systems.* 35(2), 647-682.

Xu, D. Nageshwara, N.S. & Son, Y. (2016). A Service Oriented Simulation Integration Platform for Hierarchical Manufacturing Planning and Control. *International Journal of Production Research.* 54(23) 7212-7230.

**Week 2—Assignment: Plan for a Distributed Software Architecture in a Distributed System (10 Points)**

Part I:

Using the diagram designed during Week 1, expand the conceptual distributed systems model, to include software considerations for effective operations and service management using the network. The diagram and narrative should cover the following criteria for configuration:

* The system must support multiple, autonomous agents for shared resources while accessing the shared states to perform real-time updates.
* The state of the system should be distributed across multiple client or server nodes.
  + There should only be one centralized service that supports users. It should support logging on, adding clients or servers, deleting clients or servers, and any other tasks to keep the system organized.
* The system should be robust and include provisions to manage fault-tolerance or communication gaps.
  + In the case of a node crashing the system, how could the remaining network connections continue to operate?
  + After a node crashes, how can it recover to its normal state and resume operations (tolerance levels)?

Part II:

You could consider CORBA and Jini as competing options for creating distributed systems. However, while CORBA has taken off, Jini has been slow to be adopted and has actually been turned over to the Apache Foundation and is called River. For this task, implement a Hello World program in each technology, and then reflect on the current state of market share for each technology. In your research, identify any threats to CORBA’s dominance of the market. You should submit screenshots of the working Hello World programs, code, and an explanation of how each was constructed. Finally, include a reflection on the two technologies and if you could identify any future technology that may replace CORBA.

Length: 4-9 pages, not including title page and references. The document must include a screenshot of the updated or expanded diagram, plus the diagram file must also be submitted. Must also include an explanation of the programs, screenshots of the programs with the explanation and reflection on the exercise.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Upload your document and click the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.

**Section 2—Shared Resources in Distributed Computing**

You already studied that distributed systems contain a set of resources interconnected on a network. To adequately manage resources across the network, a resource management system must be implemented to control requests, processing, and storage for files, packets, and messages. Different mechanisms, procedures, or protocols must be configured to leverage different connection protocols and speed depending on node configuration and processing capability. Also, access controls must also be implemented for perimeter protection and establishing boundaries between intranets or internal processing and communications with external processing and message transmission.

One of the most important goals of a distributed system is to efficiently and effectively use shared resources in a system. The memory and processors of individual nodes on the network are often at rest and can have work offloaded to them to increase usage. However, this sharing does come with a price. It often results in memory faults due to contention. The use of a protocol to manage the memory to allow for fast access to shared objects is desirable, this protocol must take into account the memory contention and the possibility of nodes faulting or losing communication.

Another issue is sharing processors in a distributed system. Load balancing algorithms that optimize usage of the processor are desirable with constantly changing load requirements. Furthermore, local and global balancing must be considered in asynchronous and synchronous computation models.

Interprocess communications parameters for file sharing across applications support operations and transaction processing must be correctly configured, implemented, documented, and managed. The same happens when this landscape transcends the internal boundaries of the enterprise to external or remote connections. Access controls and mechanisms must be embedded to secure perimeter, operations reliability, and integrity of data and processing.

To effectively establish and manage a distributed system, a model or framework must be correctly configured. The network architecture must be understood by all system administrators, developers, architects, and operators for proper network resource sharing management. Tools like network management suites with capability for load balancing, sharing, and leveling—when implemented in support of system administration—may ease the process of managing complex installations.

In this section, you will study what elements need to be considered to connect each node on the system to make interprocess and intraprocess communications, protocols, and procedures possible. This will ensure that processes are managed through the right communication channels.

**Week 3: Communication Mechanisms**

Sharing of information in a distributed system is important. Access to shared data is regularly required by processes and users in the system for meaningful cooperation. These entities in the system expect that there will be guarantees about how data is shared and managed. If a system is designed to share data, it should satisfy a number of requirements before allowing data to be shared. Clients should have access to data in its most recent form and the degree of availability must be high (the first condition is in direct conflict with this). If these are met, a client will be able to reach a copy of the file at all times and even remotely stored items will be treated as local.

In networks and distributed systems, *communication* refers to the transmission of packets and messages. This week, you will study what elements need to be considered to connect each node on the system to make interprocess and intraprocess communications possible. Protocols and procedures make sure processes are managed through the right communication channels. As studied in Section 1, there are three major types of networks today: LAN, WAN, and Cloud. The main categories or type of nodes in a distributed network are communications, servers, clients, and transmission media. All these are components of a client/server architecture.

* Server: Handles network operating systems, enterprise applications, file and printer sharing, storage, communications, and mails servers, among other services. Some tools used to manage shared resources are semaphores (NOS); remote procedure calls (RPC); message-oriented middleware (MOM) for application, database, and file servers; and transmission protocols like TCP/IP for transmission media.
* Client: Nodes or computers that connect either locally or remotely and share resources, request services, files, and data for transaction processing.
* Transmission Media: Peripherals, nodes, wires, and connectors needed to establish the connection and manage resources.

In networks and distributed systems, a protocol is a special set of rules, standards, and policies used to establish a connection, communications, and distribution channels between nodes in a network. Widely used protocols are packet-switching, TCP/IP, Wi-Fi, Bluetooth, routing protocols, among many others. These protocols are implemented across all layers of the distributed system.

To establish and manage communications in distributed systems, a model or framework must be correctly configured, understood, documented, and managed. Management suites with resource management capabilities assist in managing complex installations. A model formally presented during the 1980s is the open systems interconnection, or OSI, model. It is a representation of network layers and how messages and packets are sent over a network. The OSI model is comprised of seven layers, one dependent on the other, which is why they are called a protocol stack:

* Physical Layer: Cables, connectors, and electromagnetic conversion of data to be transmitted over physical wires or aerial connections.
* Data Link Layer: Protocols to send and receive packets.
* Network Layer: Protocols for routing messages and handling congestion.
* Transport Layer: Protocols for setting the layer transition to application packets, as well as media streaming of data.
* Session Layer: Protocols to set the stage for packet conversion to applications.
* Presentation Layer: Protocols for application interaction and communication for data processing.
* Application Layer: Protocols for transaction processing, file transfers, email, web access, among others.

Other important protocols to study in this chapter are:

* TCP/IP (Transmission Control Protocol—Internet Protocol)—the standard used these days for most Internet connections.
* Remote Procedure Call (RPC)—a small script or code developed in a low-level language that locally calls a procedure to request service connection to another instance in the distributed system landscape (internal or remote), and then establishes the handshake or connection prior to sending packets of data in its raw form. One of the goals of an RPC is to act as if the call is always local. The basic steps to execute an RPC are:
  + The initiating procedure calls the client or data manager from the receiver side.
  + The initiating procedure or instance sends a message to the receiver operating system to establish a secure connection before starting packet transmission.
  + The remote or receiving node sends a response to the initiating node.
  + The initiating node packs and transmits the packets to the receiving node.
  + The receiving node unpacks or decodes the packets to build the message or file and get it ready to upload either to the database server, email server, or application.
  + The receiving node sends an acknowledgment to the sending node to confirm successful receipt of the packets.
* Middleware Protocols—applications used or embedded within the OSI model.
  + Domain Name System (DNS)—distributed service to look at and validate network address or domains.
  + Persistent Communication—message submitted and retransmitted until it reaches its destination.
  + Transient Communication—message transmitted only when both sender and receiver processes are running.

**Resources:**

Alapat, A. (2018). Network Applications Are Interactive. *Communications of the ACM.* 61(1), 46-53

Brown, P. N. & Marden, J.R. (2017). The Benefit of Perversity in Taxation Mechanisms for Distributed Routing. 2017 IEEE 56th Annual Conference on Decision and Control (CDC). 6229-6234.

Chen, J., Xi, W., Mo, W., & Sun, X. L. (2014). Modeling and analysis of mobility management in wireless communication networks. *Applied Mechanics and Materials.* 4546-4550.

Chen, M. & Sung, F. (2016). Integrated Service Discovery Architecture for Heterogeneous Networks. *International Journal of Communication Systems.* 29(4), 772-786.

Gonzalez-Perez, P. & Bilas, A. (2016). Mitigation of NUMA and Synchronization Effects in High-Speed Network Storage Over Raw Ethernet. *Journal of Supercomputing.* 72(11), 4129-4159.

Jin, W. & Kim, D. (2018). Development of Virtual Resource Based IoT Proxy for Bridging Heterogeneous Web Services in IoT Networks. *Sensors.* 18(6) 1-21.

Lazaro, D., Marques, J.M., Jorba, J. & Vilajosara, X. (2013). Descentralized Resource Discovery Mechanisms for Distributed Computing in Peer-to-Peer Environments. *ACM Computing Surveys,* 45(4), 54:1-54:40.

Li, J. Cui, Y.& Ma, Y. (2015). Modeling Message Queueing Services with Reliability Guarantee in Cloud Computing Environments Using Colored Petri Nets. *Mathematical Problems in Engineering.* 2015,1-20.

Meng, T. Li, X., Zhang, S. & Zhao, Y. (2016). A Hybrid Secure Scheme for Wireless Sensor Networks Against Timing Attacks Using Continuos Time Markov Chain & Queueing Model. *Sensors.* 16(10). 1-15.

Pierson, J. (2015). Large-Scale Distributed Systems And Energy Efficiency: A Holistic Vie*w*.

Sachs, K., Kounev, S. & Buchmann, A. (2013). Performance Modeling and Analysis of Message Oriented Event-Driven Systems.

Shahzad, A., Malrey, L., Neal Naixue, X., Gisung, J., Young-Keun, L., Jae-Young, C., & ... Ahmad, I. (2016). A Secure, Intelligent, and Smart-Sensing Approach for Industrial System Automation and Transmission over Unsecured Wireless Networks. *Sensors, 16*(3), 1-18.

**Week 3—Assignment: Compare Network Protocols Applied to a Distributed System Configuration (10 Points)**

Create a table in which you compare the main features of the main protocols being used in networks these days for varied platforms (Windows-based, Unix and Linux-based, Apple-based, and mainframe nodes), list their main features, purpose, and corresponding level in the OSI model. Then research network management suites available in the market with capability for resource sharing and multiprocessing management. Create a table listing the main features, advantages, and disadvantages of each. Include in the table the main reasons why most operating systems have not been successful in supporting distributed systems. Make sure you include major protocols representing client/server architecture, mainframe processing (systems network architecture [SNA] and its mapping to the OSI model), and Apple-based network protocols.

Research and present a table of popular programming languages for systems programming and their features, along with a diagram illustrating how the system shares the information on the hard drive. Include the sources.

Using the distributed system landscape scenario used in Weeks 1 and 2, assume the network includes an ERP server with two instances or segments of the same systems for load balancing, a database server, an email server, and a file-sharing system. Expand the distributed network diagram to now include at least three protocols to handle communication between different platforms, varying from each OSI model layer (physical connections, data link and transmission, session, presentation layers, and so on), from the table analyzed as the first part of this assignment, and its function in each. Insert the suggested protocols for each server and explain why in the paper narrative. Capture a screenshot of the diagram and insert it in the document as part of your narrative, analysis, interpretations, and explanations. Make sure you submit both the document and the diagram file

Length: 4-7 pages, not including title page and references. The document must include a screenshot of the updated or expanded diagram, plus the diagram file must also be submitted.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Upload your document and click the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.

**Week 4: Shared Objects and Processes**

For the purposes of distributed computing networks, a *process* can be thought of as a program or service to be executed. To make that processing possible, scheduling, semaphores, and control tables for internal execution control are used. Mobile computing and virtualization are also embedded in everyday processing, running multiple processes with different protocols but independent from other systems housed in the same node or computer.

*Threads* are small sequences of portions of a program code either scheduled or executed. While processes do not share resources, threads can be executed concurrently or scheduled at a certain time, executes faster than a process and share resources and connections.

Distributed shared memory (DSM) allows processes to access shared data. What makes it special is that it shares data with no interprocess communication. The idea is to make the communication between processes transparent within the distributed system from the vantage point of a programmer. The two common approaches were:

* Shared Virtual Memory—This method is very similar to the idea of pages VM on single processor systems. This is accomplished by grouping all the distributed memories into a single address space. The issue with treating the memory this way is that it does not allow for the semantics of shared data to be considered; and the granularity of the data is fixed to some arbitrarily assigned size, regardless of the data’s size.
* Object DSM—In this system, the shared data are treated as objects, similar to objects in programming, with state and actions. The DSM manages the life of the shared data. The DSM works at higher levels than most message-passing systems, giving greater control of the messages.

Whichever approach is used, the implementation of a DSM attempts to address the problem of where the data are located, how the data are accessed, sharing the data through locks, and data coherence. While these are not problems specific to parallel systems, they are closely related to distribution of DBMs, network data migrations, concurrent programming on single processor systems, and distributed systems alike.

Because of virtual processing, the scheduler program or service of the operating system manages a process table to keep track of virtual processors and processes being executed, register locations, open files, requests, and queries, memory dump and access controls for secure execution. Given the complexity of distributed systems, multithreading, or executing multiple threads that even though are executed independently, still, share resources by the inter-process communication (IPC) mechanism. To protect the integrity of thread execution, all system calls are blocked until the thread finishes executing.

When a node or CPU includes virtual configurations, then a process known as resource virtualization is used. Because a virtual machine emulates a physical node, it also considers an instruction set, its own system calls, and an application programming interface. Services like process virtual machines and virtual machine monitoring must also be configured for physical and logical load balancing.

**All resources for this week:**

##### Dua, S., Dessauer, M. & Sethi, P. (2011). Evaluating Cluster Preservation in Frequent Itemset Integration for Distributed Databases. *Journal of Medical Systems.* 35(5), 845-853.

Hazanzadeh, M. & Meybody, M. (2015). Distributed Optimization Grid Resource Recovery. *Journal of Supercomputing.* 68(3), 1538-1555.

Hazanzadeh, M. & Meybody, M. (2014). Grid Resource Discovery Based on Distributed Learning Automata. *Journal of Supercomputing.* 96(9), 909-922

Lee, J. Keleher, P. & Sussman, A. (2014). Decentralized Multi-Attribute Range Search for Resource Discovery and Load Balancing. *Journal of Supercomputing* 68(2), 890-913.

Kim, T., Khan, A., Kim, Y., Kasu, P. & Atchley, S. (2018). NUMA-Aware Tread Scheduling for Big-Data Transfers over Terabits Network Infrastructure. *Scientific Programming.* 2018, 1-8.

Lazaro, D., Marquez, J.M, Jorba, J. & Villarosa, X. (2013). Decentralized Resource Discovery Mechanisms for Distributed Computing in Peer-to-Peer Environments. *ACM Computing Surveys.* 45(4), 54:1-54:40.

Norouzi, M. & Sharifi, M. (2014). A Model for Communication Between Resource Recovery and Load Balancing Units in Computing Environments. *Journal of Supercomputing, 68(3), 1538-1555*

Pierson, J. (2015). *Large-scale distributed systems and energy efficiency: a holistic view*.

Tinghuai, M. Shi, S., Cao, H., Tian, W. & Wang, J. (2012). Review on Grid Resource Recovery: Models and Strategies. *IETE Technical Review.* 29(3), 213-222.

Zhao, W. (2014). *Building Dependable Distributed Systems: Building Dependable Distributed Systems*.

**Week 4—Assignment: Compare Network and Node File Systems and Apply to a Distributed System Configuration (10 Points)**

There are many file systems available and readable by an operating system. Each file system has its advantages and disadvantages. In this first part of the assignment, you will research file system structures, compare them, and list their advantages and disadvantages or limitations, plus their strengths depending on applications or systems for which they are used for.

* FAT
* FAT32
* NTFS
* NFS (Unix)
* AFP (Apple Filing Protocol)
* AFS (Apple File Service)
* HFS+
* ZFS
* Apple Xsan
* VMFS
* Advanced Peer-to-Peer Networking (APPN)—mainframe
* Advanced Program-to-Program Computing (APPC)—mainframe

You can discuss the features, strengths, and weaknesses by using the table shown below:

Table 2. Example assignment chart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File System | Technical Features | Strengths | Limitations | Applications/Systems |
|  |  |  |  |  |

Using the distributed system landscape scenario used in Weeks 1 and 2, assume the network includes an ERP server with two instances or segments of the same systems for load balancing, a file and database server, an email server, and a remote connection to a mainframe. Expand the distributed network diagram to now include at least five file systems to handle communication between different platforms from the table analyzed as the first part of this assignment and explain why you think is the best network file system for operations, files, and storage management.

For this task, use the NCU Library to identify three recent scholarly papers that theorize about how to implement shared memory and create a diagram of how two processes would share data in the same memory. Insert the suggested protocols for each server and explain why in the paper narrative. Capture a screenshot of the diagram and insert it in the document as part of your narrative, analysis, interpretations, and explanations. Make sure you submit both the document and the diagram file.

Length: 5-7 pages, not including title page and references. The document must include a screenshot of the updated or expanded diagram, plus the diagram file must also be submitted.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

*Upload your document and click the Submit to Dropbox button.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.

**Week 5: Shared Memory Management and Naming**

There are many models of distributed computing to manage shared memory and naming schemes. It is critical to implement a viable shared memory management model to maximize resource utilization in support of the applications to be executed across the network. These memory models need to occur between the processor and memory component internals of the systems. Three basic models for shared memory used are loosely coupled systems, closely coupled systems, and tightly coupled systems.

* Loosely Coupled Systems—In this model, nodes can asynchronously communicate and exchange messages. Each system is independent and autonomous while connected to others in the network to share resources.
* Closely Coupled Systems—In this model, nodes run only one operating system with a single domain name scheme and security domain, but still share physical and logical resources like processors, memory, and files.
* Tightly Coupled Systems or Parallel Systems—In this model, nodes share memory in a synchronous manner, has a single address space and single bus to connect and manage processors and memory.

The network operating system applies processes for network memory management like a single contiguous location, partitioned allocation, paged and segmented allocation for network activity management and data manipulation and processing.

In distributed systems, the implementation and management of a solid naming scheme system are essential for optimal network performance, scalability, integrity, and security. A name in a distributed system is a string of bits or characters that is used to refer to an entity (Van Steen, Tanenbaum, 2017). Each node in the system must have a name associated with it to avoid network collision from the node or entity side. Names must be assigned from physical nodes (computers, printers, switches, routers, firewalls) to logical nodes (files, programs, web services, network connections). Node names can be independent or flat but will still identify the node location in the system and helps maps the location and function of each within the distributed system. This is an important function for scalability of nodes in the system.

When establishing naming schemes, it is important to follow a model or naming standard to differentiate each node and its function without needing to know its exact location—but while still being able to map it in the network. These naming standards must be meaningful and representative of the nodes or entities. Address resolution protocols are used to identify the origin of the nodes initiating a message before granting access through the network. This is the concept of broadcasting messages in a network.

This week, you will explore the benefits of systems memory sharing models and naming schemes.

Be sure to review this week’s resources carefully. You are expected to apply the information from these resources when you prepare your assignments.

Reference:

Ghosh, S. (2015). Distributed Systems—An Algorithmic Approach. 2nd.Ed. Boca Raton. CRC Press.

Tanenbaum, A. S. & van Steen, M. (2017). Distributed Systems. (3rd ed.). Maarten van Steen Publisher.

Resources:

Alonso, R., Monroy, R. & Trejo, L. (2016). Mining IP Name Interactions to Detect DNS Flood Attacks on Recursive DNS Servers. *Sensors.* 16(8), 1-24.

Munir, A., Gordon-Ross, A., & Ranka, S. (2015). Modeling And Optimization Of Parallel And Distributed Embedded Systems. New Jersey. John Wiley & Sons.

Schulman, L. (2017). Move over, .com, .org and .net: As the Domain Name System Expands, what Opportunities Await for Business and Consumers. *Editor & Publisher.* 150(11), 30-33.

Sheridan, S. & Keane, A. (2015). Detection of DNS Based Covert Channels. *Proceedings of the European Conference on E-Learning.* 267-275.

Varela, C. A. (2014). Programming Distributed Computing Systems: A Foundational Approach.

Wang, K, Chen, X. & Zhu, Y. (2017). Random Domain Name and Address Mutation (RDAM) for thwarting Reconnaissance Attacks. *Plos One.* 12(5), 1-22.

Zhao, W. (2014). Building Dependable Distributed Systems: Building Dependable Distributed Systems. New Jersey. John Wiley & Sons.

**Week 5—Assignment: Create a Network Map Diagram for Naming Schemes and Shared Memory Management (10 Points)**

For this assignment, you will explore the network diagram created in the previous assignment and design a naming scheme for all nodes represented in the network diagram. You will write a narrative explaining the naming structure and meaning of each allocated node name associated with its function within the network. Once the naming scheme has been developed, map each node in the diagram with its node name and the operating system running in each. This is the concept or one of the differences between a conceptual and physical network diagram.

Using the same network diagram, you will explore and implement a memory management model. First, you will compare memory management approaches like single contiguous allocation, partitioned allocation, and segmented allocation.

Study memory models like POSIX for Unix systems, and the Windows functions for network memory mapping like CreateSharedMemory, CreateFileMapping, and MapViewofFile to map a file across multiple processors. Describe how each one of these models handles main, cache, and secondary memory; access levels; and processing activities related to each application. Create another diagram depicting the applications running in each node, now listing memory management techniques for scalable processing.

You will submit a paper of 5- to 7-pages (not including cover page and references) presenting the results of your analysis and recommendations, with screenshots of the diagrams designed along with its explanations and interpretations. You will also submit the network mapping diagrams (2) created as part of the analysis.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Upload your document and click the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.

**Section 3: Concurrency and Applications**

Distributed systems are being used worldwide to solve problems and enhance productivity while granting data integrity and operational security. Distributed systems are being used to conduct corporate activities, satellite communications, provide streaming services, scan the universe for signs of intelligent life, model gene behavior, and share files—using the Internet as the vehicle to make distributed systems a reality. While the concepts associated with client/server computing are simpler and implemented more often, there are many systems that truly distribute single problems and make problems solvable that were once considered unsolvable in a realistic timeframe.

Concurrency is the tendency for things to happen at the same time in a system. In the real world, many things are happening simultaneously; and when designing software to monitor and control real-world systems, you must deal with concurrency issues. Furthermore, if you want to solve problems in a concurrent manner, you must artificially create the same issues. Concurrent issues in software systems require handling two aspects that cause the most problems. The first issue is that the system must be able to detect external events that occur in a random order and respond to them. The second is that the system responds to these random events in some acceptable timeframe.

If concurrent events spawned independently, as would be common in truly parallel systems, handling these issues would be relatively simple. To solve the problem, you just create a different program to deal with each event. The issues surrounding concurrent processing involve the fact that, most of the time, they are interacting with each other or sharing data. This requires coordination of the processing. As an example, you can imagine a stream of traffic on separate roadways being easy to handle but as soon as you introduce an intersection where they cross it becomes more complicated.

In parallel applications, expanding the idea of multiple cores to multiple systems means communicating over a network connection so that these systems can communicate. Understanding the limitations of these systems plays an important part in improving distributed system performance. Specifically, understanding these limitations helps to understand the performance bounds of these systems. Additionally, having a grasp of the effects of the limitations can be used to improve problem-solving in the future.

Coordinating the efforts of hundreds or even thousands of systems is difficult; and in this section, you will explore problems that benefit from distributed systems. It is important to consider that there are various applications of distributed systems in the industry. Google makes use of various individual computers that are interconnected together to harness additional computing power. Google also makes use of distributed systems to provide for faster and more efficient access to information.

**Week 6: Concurrent Processing**

The Merriam-Webster Dictionary defines *concurrency* as the fact of two or more events or circumstances happening or existing at the same time. In computer systems networks, concurrency—or *concurrent processing*—can be thought of as having multiple processes being executed simultaneously for performance improvement and network stability. To avoid processing delays and increase speed while executing tasks or processes, the process, message, or task is broken down into small packets or subtasks, distributed in multiple processors across the network, and executed at the same time—giving the impression that it’s only one process or task being executed.

Sometimes, concurrency issues are imposed by the demands of the environment in which they operate. Real-time systems present unique issues where events are occurring constantly in a simultaneous fashion and must be handled within a restricted timeframe. External events must be responded to, and they can occur at any time in any order. Conventional procedural programming languages result in rather complex solutions, so the systems are separated into concurrent units that handle each event as it occurs. The hard part is determining when this solution is the best fit since the complexity of the program will be affected by the number of interacting processes.

Another set of concurrent issues exists where the concurrency is created in the system on purpose. This is because performing tasks in parallel can substantially speed up the computational work of a system if multiple CPUs are available. Problems of single processor systems can be very similar to those faced by distributed systems. Multitasking on a single computer can greatly speed up multiple processes by utilizing the processor while it is doing I/O or a computation to complete. When a system is booted, it has to load many processes as quickly as possible; doing them sequentially would only increase the time to complete the system becoming ready.

Although achieving concurrency is easy with multiple processors and processes, the interactions become more complex. Since the tasks are running on different processors, how will they communicate? When determining how they communicate, how many layers of software must be involved and how will it affect overhead in the system? For instance, the CISC-based processors would appear to be better processors in theory; but in reality, they could be much slower than RISC processors due to concurrency issues.

Processes are an important concept in any system because they can have a large effect on the performance of the system. Pipelining can increase the throughput of instructions because multiple instructions are divided into stages to overlap in execution, depending on the number of times the times the instruction is executed. The goal is to balance execution speed and processor machine cycles.

This week, you will study how processors and processes are organized for allocation and scheduling for optimal execution and smooth network performance. You will also explore how concurrent processes have been handled in the past, and then examine how they may be handled in the future as processing power increases and network connections get faster.

The subject of how the processors and processes are organized will be covered and several different models presented, including how processor allocation and scheduling in distributed systems occurs. You will focus on the issues specific to multi-processors systems like you will find in distributed systems and how they handle multiple threads of control within a process.

**Resources:**

Liu, J., Arden, O., George, M.D. & Myers, A.C. (2017). Fabric: Building Open Distributed Systems by Construction. *Journal of Computer Security.* 25(4/5), 367-426.

Alonso, R., Monroy, R. & Trejo, L. (2016). Mining IP Name Interactions to Detect DNS Flood Attacks on Recursive DNS Servers. *Sensors.* 16(8), 1-24.

Munir, A., Gordon-Ross, A., & Ranka, S. (2015). Modeling and Optimization Of Parallel And Distributed Embedded Systems. New Jersey. John Wiley & Sons.

Pierson, J. (2015). Large-Scale Distributed Systems and Energy Efficiency: A Holistic View.

Wang, T., Cai, C., Guo, D., Tang, X. & Wang, H. (2014). Clock Synchronization on Wireless Sensor Networks. A New Model and Analysis Approach Based on Networked Control Perspective. *Mathematical Problems in Engineering.* 1-19.

Zhao, W. (2014). Building Dependable Distributed Systems: Building Dependable Distributed Systems.

**Week 6—Assignment: Research Clock Synchronization Models for Processor Concurrency (15 Points)**

For this week’s assignment, you will research clock synchronization models and prepare a document discussing protocol and algorithms to manage file and task allocation, distribution, and execution across multiple processors in a system. As you insert the clock model in your network diagram from previous assignments, describe their function, structure, and benefits.

* Physical Clocks—timers, counters, registers, Universal Coordinated Time (UTC)
* Clock Synchronization Algorithms—internal and external synchronization, clock drift, and drift rate
* Network Time Protocol
* Berkeley Algorithm
* Clock Synchronization in Wireless Networks—Reference Broadcast Synchronization (RBS)
* Logical Locks—Lamport’s logical locks, event counters
* Total Ordered Multicasting
* Vector Clocks
* Token-Based Solutions—deadlocks and solutions

The pipeline on a processor is at its core distributing processes. Create a Gantt chart that shows how a modern Intel processor would pipeline incoming instructions and compare it to an AMD processor. RISC-based processors have become important because mobile devices based on the ARM architecture are RISC systems. Use a Gantt chart to map the same set of instructions from the previous chart on the latest ARM system. Then, create a comparison chart of the two competing architectures and try to identify scholarly research that details which architecture best fits a distributed system; and then present a summary.

Write a paper where you will reflect on clock synchronization models processor concurrency and its importance for distributed systems. Include all diagrams and Gantt chart along with the explanations supporting each illustration, including references.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Length: 10-20 slides not including the title and reference slides. Present the results of your analysis and recommendations, with screenshots of the diagrams designed along with its explanations and interpretations. You will also submit the network mapping diagrams (2) created as part of the analysis.

Upload your document and click the *Submit to Dropbox* button.

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.

**Section 4: Network Communications, Resource Allocation and Control Applications**

Why build a distributed system? There are many advantages to solving problems in a distributed fashion. It more fully utilizes resources in the system by creating an open and scalable model that has all of the nodes in the system available to do work and the ability to adapt to a fluid number of nodes in the system seamlessly.

That means a distributed system can be much greater and more powerful given the collective capabilities of the distributed components, than combinations of stand-alone systems. However, for a distributed system to be beneficial, it must be reliable and appropriate for the problem. Unfortunately, reliability is difficult to achieve because of the complexity of the interactions between simultaneously running components and the nature of problems.

To be truly reliable, a distributed system must have the following characteristics:

* They must be fault tolerant.
* They must be highly available and provide services even when components fail.
* They must be able to recover from component failure and rejoin the system when repaired.
* They must be consistent in their ability to coordinate the components of the system in the presence of concurrent processing.
* They must scale well when new nodes join the network or leave it. Large-scale should not result in reduced performance.
* They must have a predictable behavior and respond in a timely fashion.
* They must be secure.

Communication among software located on different machines can be achieved in several ways over a network. A low-level system uses the network layer call interface, a socket, with a custom protocol. In this scenario, the programmer must make connections between the nodes on the system and manage the incoming and outgoing messages through the software. Provisions must be included to manage exceptions and balance packet load.

Instead of using sockets as described above, a more natural approach is to use remote procedure calls (RPC). In this system, every node on the system acts like a server and a client, filling requests when asked and using resources from other nodes when needed, even across the Internet. Depending on the language, the procedure can be written like any other procedure on the computer but executed anywhere in the network. The values needed by the procedure such as arguments and return values are packaged in a universally-understood format sent between the local and remote procedures.

While RPC is reasonably well suited for the procedural programming paradigm, it does not fit well with the object-oriented programming paradigm. Instead, the Java language has implemented an alternative system called remote method invocation (RMI). It is very similar to RPC, but it is integrated into the Java language as a distributed object. RMI is more dynamic than RPC in that it allows the procedures to be called without going through the typical connection establishment. RMI fits well with the trend of distributed systems moving to mobile devices and dynamic participation.

RPC and RMI are synchronous systems; but asynchronous variants based on multithreading, which are problematic to handle and error-prone, are the trend in distributed systems. This is because systems cannot wait for an indefinitely long period for a synchronous call to terminate. Such reactive and asynchronous systems are better served by newer and more abstract communication paradigms based on events. Distributed infrastructures based on the events model are called publish and subscribe middleware, software bus, or event channel. However, it requires a powerful brokering mechanism to take care of event delivery.

The trend in communication paradigms appears to fit into two categories. In simple procedural systems that require little system support, an object-oriented principle is sufficient. However, in systems with more complex run-time environments, the more abstract and loosely coupled asynchronous systems are favored, with the caveat that scalability in such systems will be a challenge.

**Week 7: Data Structures and Fault Tolerance**

In order to effectively manage and control resource allocation and process execution, it is important to first understand how they are organized—plus anticipate potential sources of failure to categorize them and establish correcting or mitigating strategies or processes. To understand and design fault tolerance approaches in distributed systems, first the concepts of availability, reliability, and safety must be carefully considered before choosing the right data structure algorithm to implement.

1. Availability—can be defined as the property or state indicating that nodes, systems, and processes can be used immediately.
2. Reliability—can be thought of like the consistency or state that a node or system is performing well and functioning normally, and processes can be executed with minimal risk of failure.
3. Safety—can be thought of as the guarantee that, should a process or node fail, the integrity of the systems or processes will not be seriously damaged, exposed, or compromised.

Among the types of failure a distributed system can experience are:

Table 3. Types of failures in a distributed system

|  |  |
| --- | --- |
| **Type of Failure** | **Description** |
| Crash Failure | System was working fine, then halts (fail-stop, fail-safe, fail-silent) |
| Omission Failure | Fails to receive incoming messages  Fails to respond to or send messages |
| Timing Failure | The time intervals between inbound and outbound messages is inconsistent or delayed |
| Response Failure | The correctness and integrity of message content or responses are not correct or follows a different control flow |

One of the most basic data structures in computer science is the tree structure. A tree structure is a hierarchical model of the data where each node belongs to a parent node until you reach the root node of the data. Your hard drive, for example, has a root of either C: or \ (depending on operating system) and all files and folders can be accessed by traveling from the root node and following a path of connections to the desired node—in this case the subsequent folders and files, organized in a hierarchical structure similar to branches and leafs in a tree.

Be sure to review this week's resources carefully. You are expected to apply the information from these resources when you prepare your assignments.

**Resources:**

Bo, C. & Curtinola, R. (2017). Remote Data Integrity Checking with Server Side Repair. *Journal of Computer Security.* 25(6). 537-584.

Wang, T., Cai, C., Guo, D., Tang, X. & Wang, H. (2014). Clock Synchronization on Wireless Sensor Networks. A New Model and Analysis Approach Based on Networked Control Perspective. *Mathematical Problems in Engineering.* 1-19.

Jimenez, A., Garcia-Diaz, V. & Bolanos, S. (2018). A Decentralized Framework for Multi-Agent Robotic Systems. *Sensors (Basel, Switzerland),* 18(2), 1-20.

Munir, A., Gordon-Ross, A., & Ranka, S. (2015). Modeling And Optimization Of Parallel And Distributed Embedded Systems. New Jersey. John Wiley & Sons.

Zhao, W. (2014). Building Dependable Distributed Systems: Building Dependable Distributed Systems. New Jersey. John Wiley & Sons.

**Week 7- Assignment: Design a Tree Structure for File and Data Allocation and Implement a Fault Tolerance Technique (10 Points)**

Consider a scenario in which you have servers running different processes and applications that handle and share thousands of files. Among the processes or applications are:

* Domain Name Server (DNS)
* Proxy Server
* Enterprise Resource Planning (ERP) solution
* Corporate Project Management System (in-house developed application communicating with the ERP)
* Distributed File and Database Servers (Oracle, Informix, DB2, Sybase, and SQL Server)

Assess the risk factors associated with processing and execution failure to define and implement a fault tolerance strategy for network support. Write a paper presenting the results of your analysis and recommendations for fault tolerance and risk mitigation techniques for network performance reliability.

Explore data structures for file server organization (trees, stacks, queues, linked lists, etc.) and construct a table detailing the characteristics of each. Include illustrations of each, making sure all references and citations are formatted according to APA format. Design a tree-structure for file allocation from at least three applications running on the distributed system model used in the Week 1 and Week 2 assignments.

Research and discuss the most important algorithms for solving shortest path problems and how they benefit distributed systems. Draw diagrams to support your postures with the corresponding explanations:

* Dijkstra's algorithm
* Bellman–Ford algorithm
* A\* search algorithm
* Floyd–Warshall algorithm
* Johnson's algorithm
* Viterbi algorithm

You will submit a paper of 7-10 pages (not including cover page and references) presenting the results of your analysis and recommendations, with screenshots of the diagrams designed along with its explanations and interpretations. You will also submit the tree mapping diagrams (2) created as part of the analysis.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards.

Upload your document and diagram using the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.

**Week 8: Distributed Systems Resiliency and Security**

Once the files are organized and structured for ease of access and use, a network administrator must be aware that systems can and will always fail. In the previous week, you explored and analyzed fault tolerance techniques to use when managing systems and networks for optimal performance and integrity. Now it is necessary to design and implement a sound process resilience strategy and continuity plans to detect, respond, and recover from failures, threats, attacks, or file tampering, should they occur.

Consistency, availability, portioning models, and transmission controls are among the basic criteria to consider when designing fault tolerance frameworks as part of the network configuration and threat mitigation strategies of the contingency and business continuity plans. Provisions for failure detection, disruption in point-to-point communication, network collisions, server or client crashes, systems and application patch failure, broken links or communications, etc., need to be assessed and analyzed to define the course of action to take. After this, the system administrator or service can establish provisions or controls to contain further damage to systems and data integrity—and, later on, recover the network to its optimal operating state.

You will explore the Paxos model for consensus and fault tolerance management, as well as other models and strategies as you get ready for your Signature Assignment, where you will create a proposal for a fault tolerance, protection, and recovery plan.

Be sure to review this week's resources carefully. You are expected to apply the information from these resources when you prepare your assignments.

**Resources:**

Elkabbany, G. F., & Rasslan, M. (2017). Security Issues in Distributed Computing System Models. *Security Solutions for Hyperconnectivity and the Internet of Things* 211-259. Hershey, PA: IGI Global.

Foraiun, I. de Oliveira Albuquerque, R., Orozco, A., Villalba, L. & Tai-Hoon, K. (2017). Endpoint Security in Networks: An Open MP Approach for Increasing Malware Detection Speed. *Symmetry.* (9)9, 1-19.

Jed, L., Arden, O. George, M.D. & Myers, A.C. (2017). Fabric: Building Open Distributed Systems Securely by Construction. 25(4/5), 367-426.

MJIHIL, O., Dung Seong, K. & HAQIQ, A. (2016). Security Assessment Framework for Multi-Tenant Cloud with Nested Virtualization. *Journal of Information Assurance & Security.* 11(2), 87-96.

Munir, A., Gordon-Ross, A., & Ranka, S. (2015). Modeling And Optimization Of Parallel And Distributed Embedded Systems. New Jersey. John Wiley & Sons.

Xiao, Z., Kathiresshan, N. & Xiao, Y. (2016). A Survey of Accountability in Computer Networks and Distributed Systems. *Security and Communication Networks*. 9(4), 290-315.

Zhao, W. (2014). Building Dependable Distributed Systems: Building Dependable Distributed Systems. New Jersey. John Wiley & Sons.

**Week 8—Assignment: Signature Assignment: Enhance a Distributed Systems Architecture for an Enterprise to Solve Potential Problems or Issues (25 Points)**

Use the scenario for the retailer enterprise (XYZ Distributing) with different plants and stores distributed across many states and launching a globalization strategy from the Week 1 assignment of this course. The strategic plan for XYZ Distributing also includes increasing online presence and expanding their distribution channels. The retailer chain manufactures and distributes three clothing lines that are also marketed in other major retailer chains and online. The company has its headquarters in South Carolina; a major manufacturing plant in Seattle, Washington; and another in Maine; plus 30 stores nationwide. And they’ll soon be opening three international stores—one each in Spain, England, and New Zealand. They have approximately 3,000 employees. Their computing infrastructure is of approximately 500 computers, 200 point-of-sale terminals connected to servers distributed at headquarters in South Carolina, Seattle, and Maine. There are an estimated 300 printers also distributed and shared across the network.

The main applications being executed in this network are:

* Domain Name Server (DNS)
* Enterprise Resource Planning (ERP) solution
* Corporate Project Management System (in-house developed application communicating with the ERP)
* Distributed File and Database Servers (Oracle, Informix, DB2, Sybase, and SQL Server)

For your Signature Assignment, identify an operational or technical issue that could be resolved or refined by enhancing or improving the distributed systems model. Develop a proposal including or elaborating on the following items:

* Describe the problem, and then explain why you think taking a distributed approach to solving it will reduce the time to solve the problem and improve performance while ensuring security.
* Identify the resources in the system. They are virtually unlimited, but you must identify how the problem will be distributed, both in the diagrams and narrative.
* Identify and discuss how communication channels and styles with nodes in the system will occur and how access to the data will be managed.
* Identify how the problem will be distributed in the system, and how will each node get its work. Consider recommending an existing algorithm, like the Paxos algorithm, to use.
* Propose fault tolerance models to ensure availability, consistency, and integrity or operations and processes, plus recovery modes and considerations.

As an alternative to the above-listed items, solve the problem in the programming language of your choice for the scripts or logical models, and submit screenshots of the working solution, code, and chart mapping time of completion to a number of participants.

Search the NCU Library to find 10 of the most recent scholarly papers that theorize solutions to NP-Hard problems. Provide a reference and a summary of each source, and then explain what the problem is and how solving it benefits distributed systems or computer science in general. In other words, formulate a concise literature review on the state of solving NP-Hard problems. Include this reflection as part of your paper.

Length: 15-20 pages not counting cover page and references.

Your paper should demonstrate thoughtful consideration of the ideas and concepts presented in the course by providing new thoughts and insights relating directly to this topic. Your response should reflect scholarly writing and current APA standards. The paper must include a minimum of 10 references.

Upload your document and click the *Submit to Dropbox* button*.*

**Course Learning Outcomes associated with this assignment: 1, 2, 3, 4**

* Adapt traditional computer science theories to distributed and massively parallel processing environments.
* Evaluate emerging techniques in distributed computing for their efficiency and usefulness in a given situation.
* Design solutions to computing problems using distributed techniques. Evaluate the impact of emerging technologies on distributed computing.
* Design solutions to computer problems using distributed systems.