ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

CONTENIDO DE CURSO

SISTEMAS DISTRIBUIDOS

A. IDIOMA DE ELABORACIÓN

Español

B. DESCRIPCIÓN DEL CURSO

Este curso introduce principios fundamentales de los sistemas distribuidos y paralelos, extendiendo los conceptos de concurrencia y paralelismo, consistencia en el manejo de los datos, y latencia, estudiados en materias anteriores. Se exploran conceptos de comunicación y coordinación entre procesos, utilizando los modelos de paso de mensajes y memoria compartida. Bajo este contexto, se estudian los conceptos de atomicidad, consenso y espera condicional. Se recalca que resulta imprescindible el usar paralelismo para conseguir mejoras de rendimiento, y se estudian las estrategias de de-composición, diseño y arquitectura de sistemas, incluyendo estrategias de implementación, análisis de rendimiento y mejoras (tuning). Se estudia también los conceptos de seguridad y tolerancia a fallos, con un énfasis en el mantenimiento de estado replicado, introduciendo conceptos que proporcionan un enlace con los conceptos estudiados bajo el contexto de las redes de datos.

C. CONOCIMIENTOS PREVIOS DEL CURSO

• Saber programar en al menos dos de los siguientes lenguajes de programación: Java, C, C++, Python, Go, Clojure. • Sistemas operativos. • Organización de computadoras. • Linux. • Altamente recomendado: Programación multi-hilos, concurrencia, sincronización entre hilos/procesos.

D. OBJETIVO GENERAL

• Proveer a los alumnos con las herramientas y habilidades necesarias para integrar y utilizar sistemas distribuidos en su carrera profesional.

E. OBJETIVOS DE APRENDIZAJE DEL CURSO

El estudiante al finalizar el curso estará en capacidad de:

1	Comprender los diferentes conceptos y tecnologías en las que se utilizan los sistemas distribuidos así como las que integran dichos sistemas distribuidos.
2	Comprender los problemas que se presentan durante implementación de proyectos basados en tecnologías para sistemas distribuidos y paralelos.
3	Paralelizar un algoritmo mediante la aplicación de paralelismo de datos o descomposición basada en tareas.
4	Analizar las ventajas y desventajas entre las sobrecargas, escalabilidad y tolerancia a fallos, al elegir un diseño estado completo vs. un diseño "sin estado" para un servicio determinado.
5	Analizar los retos de escalabilidad asociados con un servicio creciente para acomodar muchos clientes, así como aquellos retos asociados con un servicio transitorio para acomodar varios clientes.

F. ESTRATEGIAS DE APRENDIZAJE

Aprendizaje asistido por el profesor	✔
Aprendizaje cooperativo/colaborativo:	✔
Aprendizaje de prácticas de aplicación y experimentación:	✔
Aprendizaje autónomo:	✔

G. EVALUACIÓN DEL CURSO

Actividades de Evaluación	DIAGNÓSTICA	FORMATIVA	SUMATIVA
Exámenes	✔		✔
Lecciones			
Tareas		✔	
Proyectos		✔	✔
Laboratorio/Experimental		✔	
Participación en Clase	✔	✔	
Visitas			
Otras			

H. PROGRAMA DEL CURSO

UNIDADES	Horas Docencia UNIDAD
1.- Introducción	6
1.1.- Conceptos básicos y desafíos (3 horas)	
1.2.- Entidades y paradigmas de comunicación, middleware y modelos (3 horas)	
2.- Confiabilidad	6
2.1.- Modelo de fallas, redundancia y enmascaramiento de fallas (3 horas)	
2.2.- Replicación, consenso y acuerdo, casos de estudio (3 horas)	
3.- Modelos de de-composición de tareas paralelas	6
3.1.- De-composición de tareas (3 horas)	
3.2.- Casos de estudio (3 horas)	
4.- Comunicaciones entre procesos	6
4.1.- Comunicación directa entre procesos (RPC, RMI, pedido-respuesta, casos de estudio) (3 horas)	
4.2.- Comunicación indirecta entre procesos (colas de mensajes, publicar-subscribir, casos de estudio) (3 horas)	
5.- Decisiones de diseño	12
5.1.- Latencia vs. throughput (1 horas)	
5.2.- Consistencia, disponibilidad, tolerancia a particiones (2 horas)	
5.3.- Servicios con estado o sin estado y diseños reactivos (3 horas)	
5.4.- Localidad de datos: problemática y casos de estudio (3 horas)	
5.5.- Caso de estudio: Redes de distribución de contenidos (3 horas)	
6.- Seguridades	4
6.1.- Confidencialidad, disponibilidad e integridad; modelo de ataques (2 horas)	
6.2.- Canales seguros y casos de estudio (2 horas)	
7.- Teoría	8
7.1.- Tiempo físico y tiempo lógico (3 horas)	
7.2.- Coordinación y acuerdo; elecciones (3 horas)	
7.3.- Casos de estudio (2 horas)	

I. RECURSO BIBLIOGRÁFICO

BÁSICA	1.- (1447124154) Birman, Kenneth P.. Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services (Texts in Computer Science). (Hardcover; 2012-01-16).
COMPLEMENTARIA	1.- (0132143011) Coulouris, George F. & Dollimore, Jean & Kindberg, Tim & Blair, Gordon. Distributed Systems: Concepts and Design (5th Edition). (Hardcover; 2011-05-07). 2.- (1118575210) Mukaddim Pathan & Sitaraman, Ramesh Kumar & Robinson, Dom. Advanced content delivery, streaming, and cloud services. (;).

J. DESCRIPCIÓN DE UNIDADES

1.- *Introducción*

Introducción a la unidad

Se presentan los conceptos, modelos y terminologías básicos de sistemas distribuidos. Se discuten las características y complejidades de los sitemas distribuidos, principalmente en relación al tema de concurrencia, independencia de fallas de los componentes y no disponibilidad de un reloj global. Se presentan ejemplos y casos de estudios que permiten entender estos conceptos. Otros conceptos discutidos en esta unidad incluyen (en el contexto de los sistemas distribuidos): heterogeneidad, seguridad, escalabilidad, manejo de fallas y transparencia. Se presentan los modelos arquitectuales y fundamentales que permiten entender y describir a los sistemas distribuidos.

Meta-Lenguaje

concurrencia, paralelismo, sistemas distribuidos, fallas, cluster, cliente-servidor, peer-to-peer, multi-tier, redes de sensores, middleware

Subunidades

1.1.- Conceptos básicos y desafíos (3 horas)
1.2.- Entidades y paradigmas de comunicación, middleware y modelos (3 horas)

Objetivos de Aprendizaje

1.1.- Entender que las características de concurrencia, independencia de fallas y falta de un reloj global surgen necesariamente en un sistema distribuido que consiste de componentes que coordinan sus acciones únicamente a través del paso de mensajes.
1.2.- Colocar a los sistemas distribuidos en un contexto realista, a través de ejemplos como la Web, aplicaciones móviles, redes de sensores e Internet de las Cosas (IoT).
1.3.- Analizar los beneficios de compartir recursos en un sistema distribuido.
1.4.- Analizar los desafíos relacionados a la heterogeneidad, apertura, seguridad, escalabilidad, manejo de fallas, concurrencia y transparencia, y su aplicación a los sistemas distribuidos.
1.5.- Describir los modelos conceptuales (arquitectónicos y fundamentales) que soportan el estudio de los sistemas distribuidos.

Actividades

1.1.- Actividad en clase: Sistemas P2P vs. híbridos

Analizar y discutir las ventajas y desventajas de los sistemas P2P vs. sistemas híbridos, considerando como caso de estudio el diseño de Napster.

Recursos Bibliográficos adicionales

1.1.- (Artículo) Bumper-Sticker API Design

Describe la importancia y desafíos de construir APIs abiertos.
http://www.infoq.com/articles/API-Design-Joshua-Bloch

Otros Recursos

J. DESCRIPCIÓN DE UNIDADES

1.1.- Introducción a las clases de laboratorio (Laboratorio)

Socialización de las políticas y reglamentos del laboratorio.

1.2.- Cluster I (Laboratorio)

Información de hardware y red. Instalación y configuración.

2.- *Confiabilidad*

Introducción a la unidad

Se estudia qué cosas pueden reducir la confiabilidad de un sistema distribuido, con un énfasis en los mecanismos de manejo de fallas; brevemente se discuten cómo problemas de seguridad informática pueden afectar a la confiabilidad de un sistema distribuido. Se estudian técnicas de enmascaramiento de fallas, en particular, se discute y analiza el rol de la redundancia (de tiempo, física y de información) en la construcción de sistemas distribuidos confiables. Se estudian varios ejemplos en los que la replicación activa y pasiva ha sido usada para construir sistemas distribuidos altamente confiables en presencia de fallas.

Meta-Lenguaje

redundancia, confiabilidad, tolerancia a fallos, enmascaramiento de fallas, replicación, idempotencia

Subunidades

2.1.- Modelo de fallas, redundancia y enmascaramiento de fallas (3 horas)
2.2.- Replicación, consenso y acuerdo, casos de estudio (3 horas)

Objetivos de Aprendizaje

2.1.- Analizar cómo las fallas afectan la confiabilidad de un sistema distribuido.
2.2.- Caracterizar los alcances de las diferentes técnicas de enmascaramiento de fallas.
2.3.- Comparar las ventajas y desventajas de los diferentes tipos de replicación en sistemas distribuidos.
2.4.- Describir cómo diferentes sistemas distribuidos reales logran ser altamente confiables, aún en presencia de fallos.

Actividades

2.1.- Análisis de efecto de idempotencia en operaciones

Realizar el ejercicio 5.7 del libro o algún equivalente, que permita entender la diferencia entre operaciones idempotentes o no idempotentes. Luego, analizar el rol de dichas operaciones en la construcción de sistemas distribuidos confiables, especialmente bajo el contexto de re-envío de operaciones (redundancia de tiempo).

2.2.- Cálculo de nivel de disponibilidad de un sistema distribuido

Resolver el ejercicio 18.1 del libro o uno equivalente, en el que dada una cierta probabilidad de fallo de los componentes de un sistema distribuido, los estudiantes calculen el nivel de disponibilidad del servicio. Luego de realizar el cáluclo, los estudiantes deberán analizar y discutir los resultados obtenidos.

Recursos Bibliográficos adicionales

2.1.- (Artículo) The Byzantine Generals Problem

Leslie Lamport, Marshall Pease and Robert Shostak
The Byzantine Generals Problem
ACM Transactions on Programming Languages and Systems 4, 3 (July 1982), 382-401.

PDF

2.2.- (Artículo) Reaching Agreement in the Presence of Faults

J. DESCRIPCIÓN DE UNIDADES

Reaching Agreement in the Presence of Faults (Pease, Shostak y Lamport). Journal of ACM. 1980. Ganador del 2005 Edsger W. Dijkstra Prize in Distributed Computing. Disponible en: http://research.microsoft.com/en-us/um/people/lamport/pubs/reaching.pdf

2.3.- (Otros) The writings of Leslie Lamport

The writings of Leslie Lamport. Blog. Disponible en: http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html

2.4.- (Artículo) Fidelity and Yield in a Volcano Monitoring Sensor Network

Fidelity and Yield in a Volcano Monitoring Sensor Network, Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006), Seattle, November 2006.

2.5.- (Otros) Presentaciones varias sobre HDFS

Presentaciones varias sobre HDFS por Konstantin Shvachko, disponibles en: http://www.slideshare.net/KonstantinVShvachko

2.6.- (Artículo) An argument for increasing TCP's initial congestion window

An argument for increasing TCP's initial congestion window, Nandita Dukkipati, Tiziana Refice, Yuchung Cheng, Jerry Chu Tom Herbert, Amit Agarwal, Arvind Jain and Natalia Sutin, ACM SIGCOMM Computer Communication Review, 2010. Disponible en: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36640.pdf

Otros Recursos

2.1.- Master I (Laboratorio)

Instalación y configuración de sistema operativo en nodo master.

2.2.- Nodos I (Laboratorio)

Instalación y configuración de sistema en nodo esclavo.

3.- *Modelos de de-composición de tareas paralelas*

Introducción a la unidad

Se estudia cómo se puede crear programas con tareas que se ejecuten en paralelo; es deci, de-componer un programa en tareas. Se analiza el rendimiento de dichas de-composiciones. Se analiza cómo se puede ejecutar dichas de-composiciones en un sistema distribuido, incluyendo casos de estudio de sistemas reales.

Meta-Lenguaje

concurrencia, paralelismo, planificación, de-composición de tareas

Subunidades

3.1.- De-composición de tareas (3 horas)
3.2.- Casos de estudio (3 horas)

Objetivos de Aprendizaje

3.1.- Definir terminología y conceptos frecuentemente utilizados en la computación en paralelo.
3.2.- Diseñar algoritmos paralelos que resuelvan un problema específico, con un énfasis en de-composición de tareas basada en datos de entrada.
3.3.- Analizar la complejidad y rendimiento esperado de un algoritmo paralelo, en base a la de-composición de sus tareas y de la cantidad de procesadores o nodos disponibles para su ejecución.
3.4.- Describir cómo plataformas de procesamiento distribuido como Hadoop y Spark realizan la planificación de tareas distribuidas.

Actividades

J. DESCRIPCIÓN DE UNIDADES

3.1.- Analizar el rendimiento de un algoritmo paralelo

Dada una de-composición de tareas, analizar el rendimiento de la misma, tanto en un entorno con recursos ilimitados como en un entorno con recursos limitados o reducidos.

3.2.- Convertir un algoritmo serial en uno paralelo

Diseñar un algoritmo paralelo que resuelva el mismo problema que uno serial; analizar el rendimiento de ambos.

Recursos Bibliográficos adicionales

3.1.- (Libro) Introduction to parallel computing

(0201648652) Grama, Ananth. Introduction to parallel computing. (2003).
Introduction to parallel computing, Grama, Ananth Grama, George Karypis, Vipin Kumar, Anshul Gupta, Addison-Wesley. 2003

3.2.- (Artículo) MapReduce: Simplified Data Processing on Large Clusters

MapReduce: Simplified Data Processing on Large Clusters. Jeffrey Dean and Sanjay Ghemawat. Symposium on Operating System Design and Implementation (OSDI). 2004. Disponible en: http://research.google.com/archive/mapreduce.html

Otros Recursos

3.1.- Cluster II (Laboratorio)

Instalación y configuración de middleware y establecimiento de la relación maestro/esclavo.

3.2.- Aplicaciones distribuidas (Laboratorio)

Instalación y configuración de algún software distribuido.

4.- *Comunicaciones entre procesos*

Introducción a la unidad

Se estudian mecanismos directos e indirectos de comunicación entre procesos, haciendo énfasis en cómo se construyen los middlewares que dan soporte a dichos mecanismos, así como las diferencias en rendimiento y garantías proporcionadas por cada uno de estos mecanismos.

Meta-Lenguaje

comunicación entre procesos, RPC, RMI, marshalling, publish-subscribe, colas de mensajes

Subunidades

4.1.- Comunicación directa entre procesos (RPC, RMI, pedido-respuesta, casos de estudio) (3 horas)
4.2.- Comunicación indirecta entre procesos (colas de mensajes, publicar-subscribir, casos de estudio) (3 horas)

Objetivos de Aprendizaje

4.1.- Estudiar las características generales de los mecanismos de comunicación entre procesos.
4.2.- Entender las decisiones de diseño que se deben tomar al implementar protocolos pedido-respuesta, así como las complejidades relacionadas a la transmisión de estructuras de datos complejas en una red de datos.
4.3.- Identificar los componentes necesarios para que un middleware pueda implementar una absstracción de invocación remota.
4.4.- Analizar las ventajas y desventajas de la comunicación indirecta entre procesos distribuidos (versus implementar un sistema fuertemente acoplado usando comunicación directa).

Actividades

4.1.- Discusión de TCP vs. UDP

Discusión sobre cuándo usar TCP vs. cuándo usar UDP.

J. DESCRIPCIÓN DE UNIDADES

4.2.- Caso de estudio: Protocol Buffers y Apache Thrift

Analizar el rol de Protocol Buffers y Apache Thrift en el diseño de sistemas distribuidos modernos.

Recursos Bibliográficos adicionales

4.1.- (Artículo) The many faces of publish/subscribe.

The many faces of publish/subscribe. Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec.. ACM Comput. Surv. 35, 2. 2003. Disponible en: http://dl.acm.org/citation.cfm?id=857078

Otros Recursos

4.1.- Cluster III (Laboratorio)

Simulación de funciones y análisis de rendimiento del cluster.

4.2.- Virtualización I (Laboratorio)

Información de hardware y red. Instalación y configuración de sistema operativo hipervisor.

5.- *Decisiones de diseño*

Introducción a la unidad

Se analizan varias decisiones de diseño de sistemas distribuidos que afectan su: escalabilidad, rendimiento, tolerancia a fallos, usabilidad y seguridad.

Meta-Lenguaje

diseño

Subunidades

5.1.- Latencia vs. throughput (1 horas)
5.2.- Consistencia, disponibilidad, tolerancia a particiones (2 horas)
5.3.- Servicios con estado o sin estado y diseños reactivos (3 horas)
5.4.- Localidad de datos: problemática y casos de estudio (3 horas)
5.5.- Caso de estudio: Redes de distribución de contenidos (3 horas)

Objetivos de Aprendizaje

5.1.- Determinar, para un sistema en particular con metas concretas, si es preferible optimizar la latencia o el throughput del sistema.
5.2.- Determinar, para un sistema en particular con metas concretas, si es preferible optimizar la consistencia o la latencia del sistema.
5.3.- Determinar, para un sistema en particular con metas concretas, qué garantías de consistencia son convenientes.
5.4.- Determinar, para un sistema en particular con metas concretas, si es preferible usar un diseño con o sin estado para el servidor.
5.5.- Diseñar sistemas distribuidos de alto rendimiento a través del uso de cachés, CDNs, u otro tipo de soluciones que mejoren la localidad de los datos.

Actividades

5.1.- Latencia vs. throughput

Analizar varios sistemas distribuidos y determinar si, para cada uno, es preferible optimizar la latencia o el throughput.

5.2.- Consistencia vs. latencia

Analizar varios sistemas distribuidos y determinar si, para cada uno, es preferible asegurar una alta consistencia o sacrificar consistencia en aras de reducir la latencia.

J. DESCRIPCIÓN DE UNIDADES

5.3.- Bases de datos NoSQL

Analizar el diseño de dos bases de datos NoSQL como DynamoDB y MongoDB. ¿Cómo manejan la tensión entre confiabilidad, disponibilidad y (posibles) particiones debido a fallas? Usar terminología del teorema CAP.

5.4.- Diseños con estado vs. sin estado

Analizar las ventajas y desventajas de una implementación de un servicio con estado (vs. una sin estado).

5.5.- Caso de estudio de redes de distribución de contenidos

Analizar el diseño e impacto de CDNs como Akamai.

Recursos Bibliográficos adicionales

5.1.- (Artículo) The Akamai network: A platform for high-performance internet applications

Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. "The Akamai network: A platform for high-performance internet applications." ACM SIGOPS Operating Systems Review 44.3 (2010): 2-19.

Otros Recursos

5.1.- Virtualización II (Laboratorio)

Creación del pool de recursos y establecimiento de la relación maestro/esclavo.

5.2.- Hipervisor I (Laboratorio)

Configuración del sistema maestro/esclavo, y adición de nodos al pool de recursos.

5.3.- Hipervisor II (Laboratorio)

Creación y administración de máquinas virtuales.

5.4.- Procesos dinámicos I (Laboratorio)

Adición y modificación de hardware en los sistemas hipervisores.

6.- *Seguridades*

Introducción a la unidad

Se estudia cómo construir sistemas distribuidos confidenciales, íntegros y altamente disponibles. Se hace un énfasis en la construcción de canales seguros y autenticación usando certificados digitales.

Meta-Lenguaje

confidencialidad, integridad, disponibilidad, certificado digital, canal seguro, ataque de hombre en el medio, hashing criptográfico, criptografía

Subunidades

6.1.- Confidencialidad, disponibilidad e integridad; modelo de ataques (2 horas)
6.2.- Canales seguros y casos de estudio (2 horas)

Objetivos de Aprendizaje

6.1.- Analizar los ataques que puede sufrir un sistema distribuido.
6.2.- Entender cómo se puede construir canales seguros.
6.3.- Describir el rol de los hashes criptográficos en la confidencialidad e integridad de un sistema distribuido.
6.4.- Diseñar sistemas distribuidos seguros.

Actividades

J. DESCRIPCIÓN DE UNIDADES

6.1.- Análisis de HTTPS

Analizar qué situaciones pueden conllevar a que un navegador muestre una advertencia al usuario al establecer un canal seguro.

6.2.- Análisis de vulnerabilidad y riesgo

Usar un caso de estudio real para que los estudiantes hagan un análisis de una vulnerabilidad real y cuantifiquen el riesgo asociado.

Recursos Bibliográficos adicionales

6.1.- (Artículo) Why Johnny can't encrypt: a usability evaluation of PGP 5.0

Alma Whitten and J. D. Tygar. 1999. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In Proceedings of the 8th conference on USENIX Security Symposium - Volume 8 (SSYM'99), Vol. 8. USENIX Association, Berkeley, CA, USA, 14-14.

Otros Recursos

6.1.- Redes de sensores inalámbricos I (Laboratorio)

Configuración de nodos y algoritmos para adquisición de datos ambientales mediante sensores.

7.- *Teoría*

Introducción a la unidad

Se estudian los problemas que pueden surgir en un sistema distribuido debido a la falta de un reloj global. Se discuten las dos maneras de afrontar el problema: sincronización de relojes físicos o uso de relojes lógicos. Se estudia también el problema de coordinación y acuerdo en sistemas distribuidos y las dificultades que conlleva la "imposibilidad de consenso" en la construcción de servicios replicados con replicación activa. Se presentan varios algoritmos de elecciones de un coordinador o maestro y discuten middlewares que implementan la teoría aprendida en este capítulo.

Meta-Lenguaje

tiempo lógico, relojes lógicos de Lamport, relojes vectoriales, coordinación, acuerdo, consenso, elecciones

Subunidades

7.1.- Tiempo físico y tiempo lógico (3 horas)
7.2.- Coordinación y acuerdo; elecciones (3 horas)
7.3.- Casos de estudio (2 horas)

Objetivos de Aprendizaje

7.1.- Entender la complejidad que el no tener un reloj global trae a la construcción de sistemas distribuidos.
7.2.- Analizar las ventajas y desventajas de soluciones al problema de falta de un reloj global basadas en sincronización o relojes lógicos.
7.3.- Describir el problema teorético de coordinación y acuerdo en sistemas distribuidos, así como las soluciones prácticas que existen al mismo.
7.4.- Identificar diseños que requieran el uso de middlewares de coordinación y acuerdo como Zookeeper (ZAB) y las diversas implementaciones de Paxos y Raft.

Actividades

7.1.- Ejercicio de relojes lógicos y vectoriales

Aplicar los algoritmos estudiados en clase para resolver un ejercicio de paso de mensajes usando relojes lógicos y vectoriales. Analizar las ventajas y desventajas de estos algoritmos y determinar cuándo es más conveniente usar el uno o el otro.

Recursos Bibliográficos adicionales

J. DESCRIPCIÓN DE UNIDADES

7.1.- (Artículo) Time, Clocks and the Ordering of Events in a Distributed System

Time, Clocks and the Ordering of Events in a Distributed System. Leslie Lamport. Communications of the ACM 21, 7 (July 1978), 558-565. Disponible en: http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf

7.2.- (Artículo) Spanner: Google's Globally-Distributed Database

Spanner: Google's Globally-Distributed Database. James C. Corbett, Jeffrey Dean, et al. Symposium on Operating System Design and Implementation. 2012. Disponible en: http://research.google.com/archive/spanner.html

7.3.- (Artículo) The Part-Time Parliament

The Part-Time Parliament. Leslie Lamport. ACM Transactions on Computer Systems 16, 2 (May 1998), 133-169. Disponible en: http://research.microsoft.com/en-us/um/people/lamport/pubs/lamport-paxos.pdf

Otros Recursos

7.1.- Servicios de red I (Laboratorio)

Configuración y levantamiento de servicios de red, y pruebas de seguridad y confiabilidad.

7.2.- Programación Distribuida I (Laboratorio)

Evaluación de rendimiento de un cluster según el número de nodos.

7.3.- Programación Distribuida II (Laboratorio)

Ejecución de códigos en paralelo.

K. RESPONSABLES DE LA ELABORACIÓN DEL CONTENIDO DE CURSO

Profesor	Correo	Participación
DOMINGUEZ BONINI FEDERICO XAVIER	fexadomi@espol.edu.ec	Colaborador
ABAD ROBALINO CRISTINA LUCIA	cabadr@espol.edu.ec	Coordinador de materia