

**BALANCEADOR DE CARGA DE LLAMADAS PARA SISTEMAS DE CALL CENTER
MULTICOLA IMPLEMENTADOS SOBRE LA PLATAFORMA DE SOFTWARE LIBRE
ASTERISK**

HENRY ARMANDO RIZO SANDOVAL

**PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
BOGOTÁ, 2015**

**BALANCEADOR DE CARGA DE LLAMADAS PARA SISTEMAS DE CALL CENTER
MULTICOLA IMPLEMENTADOS SOBRE LA PLATAFORMA DE SOFTWARE LIBRE
ASTERISK**

HENRY ARMANDO RIZO SANDOVAL

**Trabajo de Grado presentado como requisito para optar al título de Magíster en Ingeniería
Electrónica**

Directores:

**MSc. Gustavo Adolfo Ramírez Espinosa
MSc. Luis Carlos Trujillo Arboleda**

**PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA
BOGOTÁ, 2015**

AGRADECIMIENTOS

Quiero agradecerle en primer lugar a Dios, por permitirme culminar esta meta que estuvo llena de sacrificios y aprendizaje, por acompañarme y levantarme cuando sentía que ya no podía más.

A mis padres Luz Mary Sandoval y Henry Gonzalo Rizo por su apoyo incondicional y por haberme enseñado que las cosas hay que empezarlas y terminarlas con un alto grado de calidad, así cueste más.

A mis hermanas Maleja y Lore por su gran amor y entender mi falta de tiempo.

A mi novia Karen, por ser una parte importante en mi vida, pero sobre todo por su paciencia, amor incondicional y por hacerme reír así estuviera pasando por un momento difícil.

A mis directores Gustavo Ramírez y Luis Carlos Trujillo que se tomaron el tiempo de enseñarme y valoraron todo mi esfuerzo de trabajar y sacar adelante la maestría.

A cada una de las personas: jefes, compañeros de trabajo, amigos y familiares que contribuyeron con poco o mucho para poder culminar de manera satisfactoria este Trabajo de Grado.

TABLA DE CONTENIDO

1.	INTRODUCCIÓN	10
2.	OBJETIVO GENERAL Y OBJETIVOS ESPECÍFICOS.....	12
2.1	OBJETIVO GENERAL.....	12
2.2	OBJETIVOS ESPECÍFICOS	12
3.	MARCO TEÓRICO	13
3.1	DEFINICIONES Y CONCEPTOS BÁSICOS DE UN SISTEMA DE CALL CENTER .	13
3.1.1	Call Center.....	13
3.1.2	Componentes de un Call Center	13
3.1.3	Servicios de un Call Center	14
3.1.4	Modelos de Negocio de un Call Center.....	14
3.2	TEORÍA DE COLAS.....	15
3.3	MODELADO DE UN SISTEMA DE CALL CENTER	16
3.3.1	Métricas.....	17
3.3.2	Modelos Basados en Teoría de Colas	19
3.3.3	Tráfico de Llamadas.....	21
4.	BALANCEO DE CARGA EN SISTEMAS DE CALL CENTER.....	23
4.1	ALGORITMOS DE BALANCEO DE CARGA	23
4.1.1	<i>Generalized Round Robin (GRR)</i>	23
4.1.2	<i>Join Shortest Queue (JSQ)</i>	24
4.1.3	<i>Minimum Expected Delay (MED)</i>	24
4.1.4	<i>Minimum Average Speed of Answer (Min ASA)</i>	25
5.	ARQUITECTURA DE ASTERISK VERSIÓN 11.X.....	26
5.1	FUNCIONALIDADES.....	26
5.2	ARQUITECTURA BÁSICA	27
5.2.1	Módulos.....	27
5.2.2	Interfaces	29
5.2.3	Plan de Marcación.....	31
6.	MODELO CONCEPTUAL DEL SISTEMA	32
6.1	DESCRIPCIÓN DEL MODELO CONCEPTUAL	32
6.2	ARQUITECTURA PROPUESTA DEL SISTEMA	33
7.	SIMULACIÓN DE LOS ALGORITMOS DE BALANCEO DE CARGA GRR, JSQ, MED Y MIN ASA	35
7.1	MODELO DE SIMULACIÓN.....	35
7.1.1	Subsistema Generador Llamadas	36
7.1.2	Subsistema Balanceador Carga.....	36
7.1.3	Subsistema ACD.....	39
7.2	CONSIDERACIONES Y CARACTERÍSTICAS	41
7.3	PARÁMETROS DE MEDICIÓN	44
7.4	ANÁLISIS DE RESULTADOS.....	44

7.4.1	Tamaño Promedio de la Cola	44
7.4.2	Tiempo Promedio de Espera en Cola.....	49
7.4.3	Utilización de los Agentes.....	51
7.4.4	Algoritmo Propuesto: <i>Minimum Expected Delay Modified</i> (MEDM)	52
8.	ANÁLISIS Y ESPECIFICACIÓN DE LOS REQUERIMIENTOS DEL BALANCEADOR DE CARGA.....	55
8.1	CONFIGURAR BALANCEADOR	55
8.2	VISUALIZAR BALANCEADOR	56
8.3	DISTRIBUIR LLAMADAS	58
8.4	REGISTRAR DATOS	59
9.	DISEÑO E IMPLEMENTACIÓN DEL BALANCEADOR DE CARGA	60
9.1	ARQUITECTURA DEL BALANCEADOR DE CARGA	60
9.1.1	Generador de Conexiones AMI	60
9.1.2	Lector de Eventos AMI	61
9.1.3	Ejecutor de Acciones AMI	61
9.1.4	Selector de Servidor de ACD	61
9.1.5	Visualizador.....	61
9.1.6	Conector Base de Datos	61
9.2	LENGUAJE DE PROGRAMACIÓN.....	61
9.3	IMPLEMENTACIÓN DEL BALANCEADOR DE CARGA.....	62
9.3.1	Clases Desarrolladas	62
9.3.2	Funcionamiento del Balanceador de Carga	62
9.3.3	Diagramas de Flujo de los Algoritmos de Balanceo de Carga	65
10.	IMPLEMENTACIÓN DEL PROTOTIPO DEL SISTEMA	68
10.1	INSTALACIÓN DE ASTERISK	69
10.2	CONFIGURACIÓN DE LOS SERVIDORES	69
10.2.1	Servidor Generador de Llamadas.....	69
10.2.2	Servidores de Acceso	70
10.2.3	Servidores de ACD	71
10.2.4	Servidor de Base de Datos.....	73
11.	PRUEBAS DEL PROTOTIPO DEL SISTEMA	74
11.1	AMBIENTE DE PRUEBAS	74
11.2	ANÁLISIS DE RESULTADOS	75
11.2.1	Tamaño Promedio de la Cola	75
11.2.2	Tiempo Promedio de Espera en Cola.....	76
11.2.3	Utilización de los Agentes.....	80
11.3	ESCALABILIDAD DEL SISTEMA	81
12.	CONCLUSIONES	83
13.	BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN	84
	ANEXOS.....	86

ÍNDICE DE FIGURAS

Figura 1. Diagrama esquemático de la tecnología de un Call Center	14
Figura 2. Modelos de negocio de un Call Center.....	15
Figura 3. Modelo de una sola cola con tres servidores	15
Figura 4. Modelo general de un sistema de Call Center.....	16
Figura 5. Diagrama de flujo del cálculo del tamaño promedio de la cola	18
Figura 6. Modelo de Call Center Erlang C	20
Figura 7. Distribución de llegada de llamadas.....	22
Figura 8. Distribución de duración de llamadas	22
Figura 9. Arquitectura de un sistema de Call Center con múltiples servidores de ACD	23
Figura 10. Funcionamiento del algoritmo GRR	24
Figura 11. Arquitectura básica de Asterisk	27
Figura 12. Módulos de Asterisk.....	28
Figura 13. Interfaces de Asterisk	29
Figura 14. Diagrama de bloques funcional de un sistema de Call Center con balanceo de carga	32
Figura 15. Diagrama de flujo del proceso de balanceo de carga para llamadas de entrada.....	32
Figura 16. Arquitectura propuesta del sistema	33
Figura 17. Modelo de simulación de un de Call Center que incluye un平衡ador de carga	35
Figura 18. Subsistema Generador Llamadas	36
Figura 19. Subsistema Balanceador Carga – Algoritmo GRR	37
Figura 20. Subsistema Balanceador Carga – Algoritmo JSQ.....	37
Figura 21. Código en MATLAB del algoritmo JSQ.....	38
Figura 22. Subsistema Balanceador Carga – Algoritmo MED.....	38
Figura 23. Subsistema Balanceador Carga – Algoritmo Min ASA	39
Figura 24. Generación de múltiples operadores del Call Center	39
Figura 25. Asignación de un operador disponible a una llamada de un cliente.....	40
Figura 26. Generador del tiempo de servicio.....	40
Figura 27. Subsistema ACD completo.....	41
Figura 28. Variación del número de operadores disponibles.....	41
Figura 29. Variación del número de clientes en cola	41
Figura 30. Tráfico de entrada de un Call Center real	42
Figura 31. Tamaño promedio de la cola del servidor ACD1 bajo el escenario homogéneo – Modelo de simulación.....	45
Figura 32. Tamaño promedio de la cola del servidor ACD2 bajo el escenario homogéneo – Modelo de simulación.....	45
Figura 33. Tamaño promedio de la cola del servidor ACD3 bajo el escenario homogéneo – Modelo de simulación.....	45
Figura 34. Tamaño promedio de la cola utilizando el algoritmo GRR bajo el escenario homogéneo – Modelo de simulación.....	46
Figura 35. Tamaño promedio de la cola utilizando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación.....	46
Figura 36. Tamaño promedio de la cola utilizando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación	47
Figura 37. Tamaño promedio de la cola utilizando el algoritmo Min ASA bajo el escenario homogéneo – Modelo de simulación	47

Figura 38. Tamaño promedio de la cola utilizando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación	48
Figura 39. Tamaño promedio de la cola utilizando el algoritmo JSQ bajo el escenario heterogéneo – Modelo de simulación	48
Figura 40. Tamaño promedio de la cola utilizando el algoritmo GRR bajo el escenario heterogéneo – Modelo de simulación	49
Figura 41. Tamaño promedio de la cola utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Modelo de simulación.....	49
Figura 42. Tiempo promedio de espera en cola utilizando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación	50
Figura 43. Tiempo promedio de espera en cola utilizando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación	50
Figura 44. Tiempo promedio de espera en cola utilizando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación.....	51
Figura 45. Diagrama de flujo del algoritmo MEDM	53
Figura 46. Utilización de los agentes empleando el algoritmo MEDM bajo el escenario homogéneo – Modelo de simulación	54
Figura 47. Utilización de los agentes empleando el algoritmo MEDM bajo el escenario heterogéneo – Modelo de simulación	54
Figura 48. Caso de Uso Configurar Balanceador	55
Figura 49. Caso de Uso Visualizar Balanceador.....	57
Figura 50. Caso de Uso Distribuir Llamadas.....	58
Figura 51. Caso de Uso Registrar Datos	59
Figura 52. Arquitectura del balanceador de carga	60
Figura 53. Diagrama de flujo para ejecución de los algoritmos de balanceo de carga	66
Figura 54. Diagrama de flujo del algoritmo GRR	66
Figura 55. Diagrama de flujo del algoritmo JSQ.....	66
Figura 56. Diagrama de flujo del algoritmo MED.....	67
Figura 57. Diagrama de flujo del algoritmo Min ASA	67
Figura 58. Arquitectura del prototipo del sistema.....	68
Figura 59. Esquemas y tablas de la base de datos.....	73
Figura 60. Tamaño promedio de la cola del servidor ACD1 bajo el escenario homogéneo – Prototipo del sistema.....	75
Figura 61. Tamaño promedio de la cola del servidor ACD2 bajo el escenario homogéneo – Prototipo del sistema.....	76
Figura 62. Tamaño promedio de la cola del servidor ACD3 bajo el escenario homogéneo – Prototipo del sistema.....	76
Figura 63. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo GRR bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo	77
Figura 64. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo JSQ bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo	78
Figura 65. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo MED bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo	79
Figura 66. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo Min ASA bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo.....	80

Figura 67. Utilización promedio de los agentes en cada uno de los servidores de ACD bajo el escenario homogéneo – Prototipo del sistema.....	81
Figura 68. Utilización promedio de los agentes en cada uno de los servidores de ACD bajo el escenario heterogéneo – Prototipo del sistema	81
Figura 69. Utilización de los agentes empleando el algoritmo GRR bajo el escenario homogéneo – Modelo de simulación.....	86
Figura 70. Utilización de los agentes empleando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación.....	87
Figura 71. Utilización de los agentes empleando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación.....	87
Figura 72. Utilización de los agentes empleando el algoritmo Min ASA bajo el escenario homogéneo – Modelo de simulación	87
Figura 73. Utilización de los agentes empleando el algoritmo GRR bajo el escenario heterogéneo – Modelo de simulación.....	88
Figura 74. Utilización de los agentes empleando el algoritmo JSQ bajo el escenario heterogéneo – Modelo de simulación.....	88
Figura 75. Utilización de los agentes empleando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación.....	89
Figura 76. Utilización de los agentes empleando el algoritmo Min ASA bajo el escenario heterogéneo – Modelo de simulación.....	89
Figura 77. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo.....	93
Figura 78. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo.....	93
Figura 79. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo.....	94
Figura 80. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo.....	94
Figura 81. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo.....	94
Figura 82. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo.....	95
Figura 83. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo.....	95
Figura 84. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo.....	95
Figura 85. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo.....	96
Figura 86. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo	96
Figura 87. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo	97
Figura 88. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo	97

ÍNDICE DE TABLAS

Tabla 1. Funcionalidades básicas y avanzadas de Asterisk	26
Tabla 2. Función de densidad de probabilidad del tráfico de entrada.....	43
Tabla 3. Tiempo promedio de espera en cola en el servidor ACD1 durante la jornada laboral – Escenario Homogéneo	49
Tabla 4. Tiempo promedio de espera en cola en el servidor ACD2 durante la jornada laboral – Escenario Homogéneo	50
Tabla 5. Tiempo promedio de espera en cola en el servidor ACD3 durante la jornada laboral – Escenario Homogéneo	50
Tabla 6. Tiempo promedio de espera en cola en el servidor ACD1 durante la jornada laboral – Escenario Heterogéneo.....	51
Tabla 7. Tiempo promedio de espera en cola en el servidor ACD2 durante la jornada laboral – Escenario Heterogéneo.....	51
Tabla 8. Tiempo promedio de espera en cola en el servidor ACD3 durante la jornada laboral – Escenario Heterogéneo.....	51
Tabla 9. Tiempo promedio de espera en cola de los algoritmos GRR, MED y MEDM durante la jornada laboral – Escenario Homogéneo.....	54
Tabla 10. Tiempo promedio de espera en cola de los algoritmos GRR, MED y MEDM durante la jornada laboral – Escenario Heterogéneo.....	54
Tabla 11. Descripción Caso de Uso Configurar Balanceador	56
Tabla 12. Descripción Caso de Uso Visualizar Balanceador	57
Tabla 13. Descripción Caso de Uso Distribuir Llamadas	58
Tabla 14. Descripción Caso de Uso Registrar Datos	59
Tabla 15. Descripción de las tablas de la base de datos	74
Tabla 16. Escenarios implementados para la verificación de la escalabilidad del sistema	82
Tabla 17. Escalabilidad del sistema – Resultados del primer escenario	82
Tabla 18. Escalabilidad del sistema – Resultados del segundo escenario.....	82
Tabla 19. Escalabilidad del sistema – Resultados del tercer escenario.....	82
Tabla 20. Principales archivos de configuración de Asterisk.....	86

1. INTRODUCCIÓN

Históricamente, los Call Centers nacieron de la oportunidad de prestar un servicio inmediato al cliente a través del teléfono. Al principio era principalmente informativo y tenía un carácter de servicio adicional a la oferta del producto. Sin embargo, su utilización se expandió considerablemente, debido principalmente a dos factores:

- Fuerte competencia, que convirtió un servicio de lujo en un canal habitual y necesario de contacto con el cliente.
- Fuerte demanda del cliente, que cada vez goza de menos tiempo de ocio y por tanto le da más valor a su tiempo libre.

El mercado de los Call Centers es uno de los más dinámicos y avanzados tecnológicamente. Sin embargo, debido a su carácter competitivo ha enseñado a los usuarios a reclamar por todo tipo de servicios y a exigir la forma en la que quieren relacionarse con la empresa. Esto ha obligado a los Call Centers a integrar diversos canales de interacción con el cliente tales como: teléfono, e-mail, chat, entre otros, con la misma sencillez y eficacia que proporciona una solución a través de una llamada telefónica, ofreciendo a los clientes un único punto de contacto para resolver sus necesidades.

En respuesta al desafío de contar con una plataforma tecnológica que mejore la experiencia del usuario, y brinde mayor fiabilidad y escalabilidad dentro de los sistemas de Call Center, muchas empresas ofrecen soluciones que cuentan con funcionalidades como: enrutamiento multi-skill, alta disponibilidad y balanceo de carga. El *enrutamiento multi-skill* permite enviar la llamada al agente más apropiado y no simplemente al siguiente agente disponible. La *alta disponibilidad* consiste en una solución de hardware y/o software que garantiza la disponibilidad del servicio en todo momento. Mientras que el *balanceo de carga* por su parte, facilita la distribución equitativa de las llamadas entre múltiples servidores donde se registran los agentes, ayudando a maximizar la capacidad y el rendimiento de un sistema de Call Center.

Los desarrolladores de la comunidad de software libre han venido trabajado en el diseño y la implementación de hardware y software para este tipo de infraestructura. Asterisk constituye uno de los proyectos de software libre más representativos de una PBX¹ con funcionalidades de encolamiento de llamadas y distribución automática de llamadas (ACD²). Además, Asterisk ha incorporado la mayoría de estándares de telefonía del mercado, tanto los tradicionales como TDM (*Time Division Multiplexing*), como los de telefonía IP (*Internet Protocol*). Esto le permite conectarse a las redes públicas de telefonía tradicional e integrarse fácilmente con centrales tradicionales y otras centrales IP. Al contrario de otras plataformas, Asterisk permite la integración con otro tipo de aplicaciones o sistemas a través de *sockets TCP/IP*, permitiendo el desarrollo de aplicaciones en lenguaje de programación tales como: C++, Java, Perl, entre otros.

En este trabajo de grado se propone un平衡ador de carga para la plataforma de software libre Asterisk, que utilice las métricas fundamentales de un sistema Call Center para la distribución de

¹ *Private Branch Exchange* (PBX): Es un sistema telefónico no público encargado de manejar líneas telefónicas entre un grupo de usuarios, permitiendo el enrutamiento de llamadas desde el exterior hacia extensiones internas y viceversa.

² *Automatic Call Distributor* (ACD): Es una tecnología de Call Center (hardware o software) que permite distribuir eficientemente las llamadas a un grupo de agentes. El ACD permite poner en cola las llamadas cuando los agentes se encuentran ocupados.

llamadas entre múltiples servidores de ACD que se encuentren interconectados en una Red de Área Local (LAN: *Local Area Network*). Adicionalmente, este desarrollo permitirá que los sistemas de Call Center implementados sobre esta plataforma puedan tener un mayor crecimiento sin comprometer su funcionamiento y calidad del servicio.

2. OBJETIVO GENERAL Y OBJETIVOS ESPECÍFICOS

El desarrollo de este trabajo de grado tiene como fin alcanzar los siguientes objetivos:

2.1 OBJETIVO GENERAL

Diseñar e implementar un balanceador de carga de llamadas para sistemas de Call Center multicola implementados sobre la plataforma de software libre Asterisk, con el fin de brindar mayor escalabilidad al sistema.

2.2 OBJETIVOS ESPECÍFICOS

- Diseñar e implementar un modelo de simulación en la herramienta MATLAB, que permita validar el comportamiento de los algoritmos de balanceo de carga GRR, JSQ, MED y Min ASA.
- Diseñar e implementar un módulo de balanceo de carga para sistemas de Call Center multicola implementados sobre plataforma Asterisk, incluyendo los algoritmos GRR, JSQ, MED y Min ASA.
- Implementar un prototipo de un sistema de Call Center con múltiples servidores de ACD Asterisk interconectados en una Red de Área Local, que incluya el balanceador de carga.
- Validar el funcionamiento del balanceador de carga de llamadas mediante el prototipo.

3. MARCO TEÓRICO

3.1 DEFINICIONES Y CONCEPTOS BÁSICOS DE UN SISTEMA DE CALL CENTER

3.1.1 Call Center

Un Call Center es un conjunto tecnológico y administrativo que permite unificar la inteligencia y potencia de procesamiento de los sistemas informáticos y las facilidades de la conmutación de llamadas telefónicas, para suministrar información a quienes llaman. De esta forma, el Call Center es el punto de contacto entre un cliente y la empresa por medio de un enlace telefónico [1].

3.1.2 Componentes de un Call Center

Los siguientes son los componentes típicos que conforman un Centro de Llamadas o Call Center [1,2]:

- Central Telefónica o PBX (*Private Branch Exchange*): Es un sistema telefónico no público encargado de manejar líneas telefónicas entre un grupo de usuarios, permitiendo el enrutamiento de llamadas desde el exterior hacia extensiones internas y viceversa.
- Servidor CTI (*Computer Telephone Integration*): Es un *middleware*³ que hace las funciones de “director de orquesta” de todos los componentes hardware y software del Call Center. Este servidor define e inscribe a los agentes telefónicos al correspondiente ACD, imparte órdenes para el envío de información a los diferentes puestos de los agentes, y/o almacena y estructura la información para los diferentes reportes de operación que se requieran.
- Servidores de Bases de Datos: Son repositorios de información de los clientes de una organización.
- Sistema Interactivo de Respuesta de Voz (IVR: *Interactive Voice Response*): Es un conjunto de hardware y/o software que se encarga de la gestión de llamadas entrantes (*inbound*) en una organización. Este sistema permite y facilita la entrega de mensajes hablados a los usuarios que llaman, de tal forma que estos puedan acceder a la información residente en las bases de datos de la empresa. Al IVR se le asignan los trabajos de suministro de información rutinaria, y se deja para los agentes la atención especializada y específica de los requerimientos de los usuarios.
- *Distribuidor Automático de Llamadas* (ACD: *Automatic Call Distributor*): Es una tecnología de Call Center (hardware o software) que permite distribuir eficientemente las llamadas a un grupo de agentes. El ACD permite poner en cola las llamadas cuando los agentes se encuentran ocupados. Durante su funcionamiento permite el registro detallado de las llamadas recibidas, atendidas, abandonadas, etc., así como el registro de los tiempos de las mismas.
- Estación de Trabajo de los Agentes: Cada uno de los puestos de operación donde se ubican los agentes telefónicos para realizar su interacción con los usuarios que llaman al sistema.

³ *Middleware*: Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

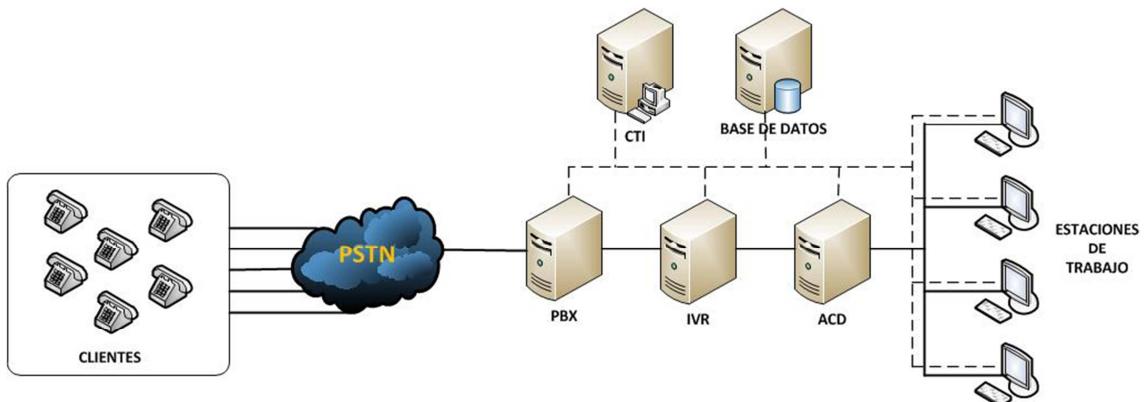


Figura 1. Diagrama esquemático de la tecnología de un Call Center [2]

3.1.3 Servicios de un Call Center

Los servicios que brinda un Call Center se pueden ubicar en dos grandes grupos: servicios *inbound* y servicios *outbound* [1,2]:

- Servicios *Inbound* (Servicios de Entrada): Son aquellos en los que el cliente se comunica con el Call Center y son catalogados como servicios de atención al cliente (SAC), en los que comúnmente se resuelven consultas de información general respecto a un producto o servicio. Desde un punto de vista funcional, el cliente realiza una llamada al Call Center de la empresa, la cual puede ser atendida en primera instancia por un IVR y posteriormente por un agente o asesor.
- Servicios *Outbound* (Servicios de Salida): En este tipo de servicios, el Call Center se comunica con el cliente a través de una llamada telefónica con el fin de realizar ventas, cobranzas, encuestas y/o actualización de datos. Visto desde un aspecto funcional, el agente o asesor del Call Center realiza una llamada al cliente o el Call Center utiliza un sistema automático de marcación para contactar al cliente.

3.1.4 Modelos de Negocio de un Call Center

A continuación se describen los modelos de negocio de un Call Center:

- *Inhouse*: La organización se responsabiliza por adquirir la tecnología, las telecomunicaciones y el personal.
- *Application Service Provider* (ASP): Las aplicaciones del Call Center son rentadas por un proveedor de servicios externo a la organización.
- *Outsourcing*: El Call Center es tercerizado a una empresa externa a la organización. El tercero es responsable de la infraestructura y el personal.

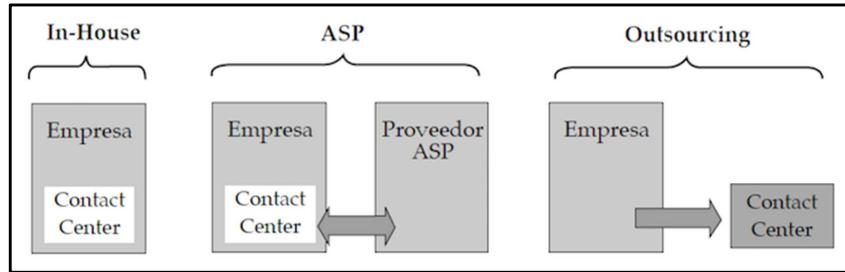


Figura 2. Modelos de negocio de un Call Center [1]

3.2 TEORÍA DE COLAS

Muchas industrias de servicios tienen un sistema de colas, en el que los productos o clientes llegan a una estación y esperan en una fila, luego obtienen algún tipo de servicio y finalmente salen del sistema. Este tipo de fenómeno se origina cuando los usuarios de un determinado servicio llegan con mayor rapidez a la que el sistema tiene capacidad de despachar, y por tal motivo se acumulan personas u objetos que deben esperar para ser atendidos [3].

La teoría de colas incluye el estudio matemático de las colas o líneas de espera y provee un gran número de modelos matemáticos para describirlas. Las fórmulas para cada modelo indican cual debería ser el desempeño del sistema correspondiente y señalan la cantidad esperada de tiempo y personas en una cola, en una gama de circunstancias.

Una línea de espera está constituida por un cliente que requiere de un servicio, el cual es proporcionado por un servidor en un determinado periodo de tiempo. Los clientes entran aleatoriamente al sistema y forman una o varias colas para ser atendidos. Si el servidor está desocupado, se proporciona el servicio a los elementos de la cola en un lapso de tiempo denominado tiempo de servicio y luego abandonan el sistema. A continuación se muestra un ejemplo de una cola con tres servidores:

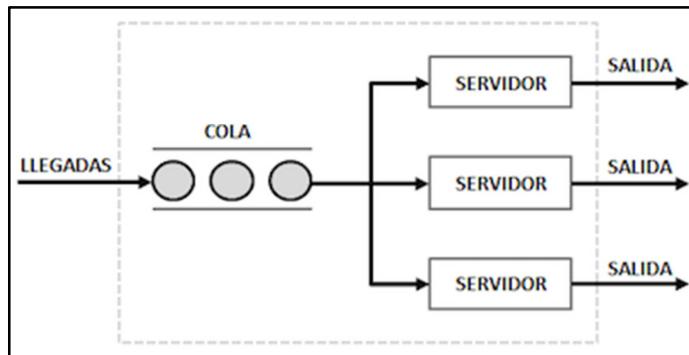


Figura 3. Modelo de una sola cola con tres servidores

Un sistema de colas pueden definirse mediante los siguientes componentes:

1. *Patrón de llegada de los clientes:* Se refiere al tiempo entre llegadas sucesivas de los clientes al sistema. Los clientes llegan en tiempos aleatorios e independientes entre sí (proceso estocástico). Para un sistema de Call Center, la probabilidad de que k llamadas ingresen en

un lapso de tiempo determinado, puede ser representada mediante una distribución de Poisson (ver sección 3.3.3.1 *Llegada de Llamadas*).

2. *Patrón de servicio de los servidores*: Cada cliente requiere de cierta cantidad de tiempo proporcionado por el servidor, por lo que el tiempo de servicio requerido varía entre un cliente y otro. En un Call Center la duración de las llamadas puede ser modelada mediante una distribución de probabilidad exponencial, donde la mayoría de llamadas son más cortas que el promedio de duración (ver sección 3.3.3.2 *Tiempo de Servicio de las Llamadas*).
3. *Política de atención de las colas*: Describe el orden según el cual los clientes van siendo tomados de la cola de espera. Por ejemplo: FIFO (*First In First Out*), el primero en entrar es el primero en salir, y LIFO (*Last In First Out*), en la cual el último en llegar es el primero en salir.
4. *Tamaño máximo de las colas*: Es la cantidad de clientes permitidos en la cola dependiendo de las características del sistema.
5. *Número de servidores*: En un sistema de Call Center, el número de servidores se refiere a la cantidad de agentes u operadores que atienden las llamadas de los clientes.

Un Call Center está basado en un sistema de colas de disciplina FIFO, en el cual las llamadas de los clientes son enviadas a diferentes colas por medio de un IVR y atendidas por una serie de operadores (servidores) en un lapso de tiempo determinado.

3.3 MODELADO DE UN SISTEMA DE CALL CENTER

Un Call Center se constituye en un complejo sistema, tanto desde el punto de vista tecnológico, como en lo que refiere a la interacción humana. La Figura 4 muestra un modelo general de lo que puede ser un Call Center con N agentes, un solo tipo de llamada entrante y $N+K$ canales de voz que corresponden al número de agentes contestando llamadas más el número de llamadas que se encuentran en cola [7]:

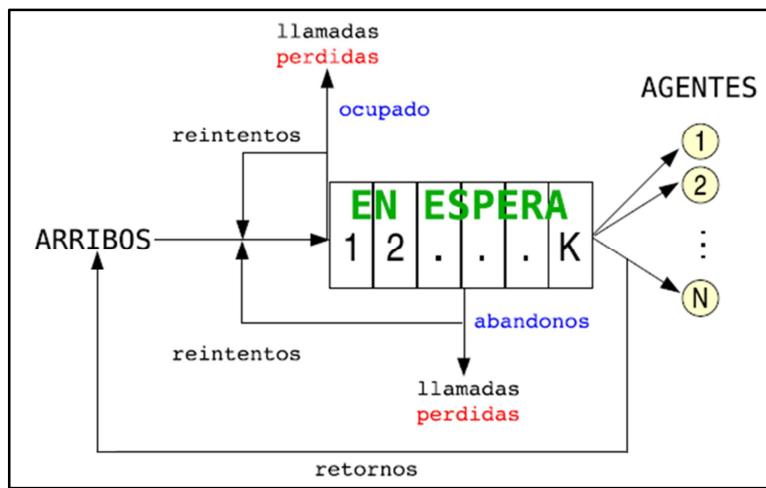


Figura 4. Modelo general de un sistema de Call Center [7]

Cuando un cliente llama tiene tres opciones:

1. Ser atendido inmediatamente si hay algún agente libre.
2. Esperar en cola, si todos los agentes están ocupados y hay canales disponibles.
3. No poder ingresar al sistema, por no haber canales disponibles.

Luego, para el caso del cliente que queda en cola existen dos posibilidades:

1. Esperar hasta ser atendido, cuando algún agente se desocupe.
2. Abandonar el sistema sin ser atendido.

Los retornos corresponden a clientes que fueron atendidos, pero por alguna razón vuelven a comunicarse y los reintentos, tanto para el caso de bloqueos como de abandonos, son los clientes que vuelven a llamar al Call Center.

3.3.1 Métricas

Las métricas para un Call Center, permiten de manera objetiva medir la utilización efectiva de los recursos con el fin de atender una muy buena porción de las peticiones de los clientes. Existen métricas de servicio que se calculan con el número de llamadas, abandonos, duración, etc. y otras métricas de calidad que miden el oportuno y correcto manejo de la información proporcionada al cliente.

El estándar de calidad actualmente utilizado a nivel mundial está basado en el modelo de gestión de rendimiento COPC que ha sido desarrollado por el *Customer Operations Performance Center* [8,9].

En la siguiente sección se exponen las métricas más utilizadas en los sistemas de Call Center actualmente. Ninguna de estas métricas tiene en cuenta el tiempo que el usuario o cliente ha estado en el IVR, debido a que este es un módulo opcional del sistema.

3.3.1.1 *Tiempo Promedio de Espera (ASA)*

El tiempo promedio de espera (*Average Speed of Answer*) es comúnmente usado en los Call Center para medir el tiempo promedio que espera un cliente antes de que su llamada sea atendida por un agente. En general, este promedio es aceptado para estimaciones y tendencias debido a la naturaleza de la distribución de la llegada de las llamadas y su duración. Se calcula como:

$$ASA = \text{Tiempo total de espera en cola de todas las llamadas} / \text{Total de llamadas contestadas}$$

Esta métrica no considera los tiempos de espera en cola de las llamadas abandonadas.

3.3.1.2 *Tiempo Promedio de Servicio (AHT)*

El tiempo promedio de servicio (*Average Handling Time*) define el tiempo que el agente está ocupado atendiendo un servicio. Este tiempo promedio también se denominada tiempo de trabajo. En el cálculo del AHT también se consideraran los tiempos de documentación o postproceso realizados por los agentes. En algunas ocasiones se realiza el cálculo de esta métrica por aparte para poder identificar la eficiencia de cada uno de los agentes por separado. Se calcula como:

$$AHT = (\text{Tiempo hablado} + \text{Tiempo de postproceso}) / \text{Número de llamadas contestadas}$$

3.3.1.3 Tamaño Promedio de la Cola

El tamaño de la cola indica cuantos clientes se encuentra esperando en una cola en un instante de tiempo determinado. Esta métrica es proporcionada por el distribuidor automático de llamadas ACD.

De acuerdo con [10], el tamaño promedio de la cola en el tiempo puede ser calculado como se muestra en el siguiente diagrama de flujo:

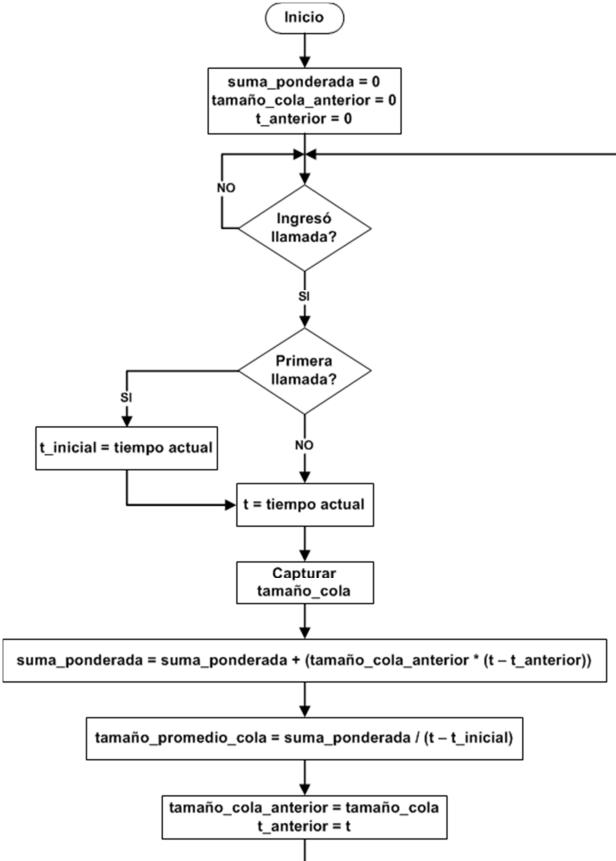


Figura 5. Diagrama de flujo del cálculo del tamaño promedio de la cola

3.3.1.4 Utilización de los Agentes

Corresponde a la fracción promedio en que los recursos (agentes) están ocupados atendiendo clientes. Esta métrica puede ser calculada de la siguiente manera:

$$\text{Utilización Agentes} = \text{Tasa Llegada} / \text{Capacidad Procesamiento}$$

Donde:

- *Tasa Llegada*: Tasa promedio de llegada de los clientes por unidad de tiempo.
- *Capacidad Procesamiento*: Tasa de procesamiento total en la cual los clientes son procesados por el número de agentes conectados al sistema.

3.3.1.5 Nivel de Servicio (SL)

El nivel de servicio (*Service Level*) se considera como una medida global de la operación. Define el porcentaje de llamadas que son contestadas antes de un umbral definido con respecto al total de llamadas entrantes en el sistema. El nivel más utilizado es el 80/20 que implica que el 80% de las llamadas deben ser contestadas antes de 20 segundos. Los recursos del Call Center (agentes y líneas telefónicas) deben estar dimensionados para atender este nivel de servicio. El nivel de servicio además de medir la eficiencia a nivel global, permite medir los abandonos prematuros de llamadas.

3.3.1.6 Porcentaje de Abandono

Corresponde a la fracción del total llamadas que ingresaron a la cola, y fueron colgadas por el cliente antes de ser atendidas:

$$\text{Porcentaje Abandono} = \frac{\text{Número de abandonos}}{\text{Número total de llamadas en cola}}$$

3.3.2 Modelos Basados en Teoría de Colas

El estudio de los Call Centers está basado principalmente en la teoría de colas. Los modelos de colas consideran una o más fuentes que realizan peticiones de servicio, las cuales son atendidas por servidores con el fin de satisfacer la demanda. En el caso de un sistema de Call Center, las fuentes son las llamadas y los servidores son los agentes del sistema. En los modelos generalmente se considera que el número de peticiones desde las fuentes es bastante grande comparado con el número de servidores.

Los diferentes modelos de teoría de colas pueden distinguirse por la manera como se tratan los clientes (llamadas telefónicas) y se identifican por la notación Kendall:

A/B/S/K

- **A:** Comportamiento de llegada de los clientes. Si A = M es un proceso de Poisson.
- **B:** Distribución de la duración del tiempo de servicio. M si es exponencial, D si es determinística y G para una distribución general.
- **S:** Número de servidores.
- **K:** Número de clientes potenciales en el sistema (Capacidad del sistema).

En caso de que la cola de espera sea infinita, se puede omitir el cuarto parámetro y se tendrá un modelo **A/B/S**.

A continuación se presentan algunos de los modelos más conocidos:

3.3.2.1 Modelo Erlang C

Es el modelo tradicional para el dimensionamiento de Call Centers, es del tipo M/M/N, donde N representa la cantidad de agentes y el número de líneas es ∞ . Este modelo considera llegadas y tiempos de servicio exponenciales de tasas λ y μ respectivamente. A continuación se muestra este modelo:

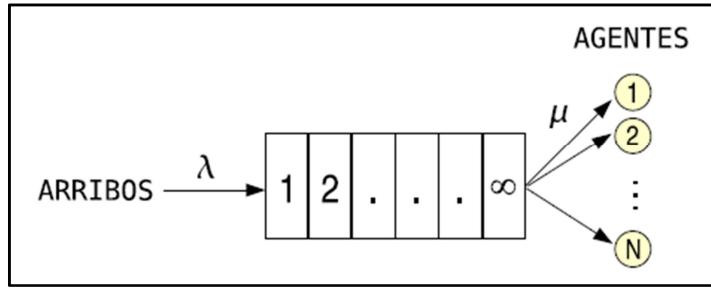


Figura 6. Modelo de Call Center Erlang C [7]

Los abandonos de llamadas, bloqueos y reintentos (mostrados en la Figura 4) no son tomados en cuenta en este modelo.

A partir de este modelo, es posible obtener analíticamente expresiones matemáticas para el nivel de servicio (SL) y el tiempo promedio de espera en cola (ASA), tal y como se muestra en [7]:

- Siendo $\rho = \frac{\lambda}{\mu}$ se calcula:

$$E_c(N, \rho) = \frac{\rho^N}{(N-1)!(N-\rho)} \cdot \frac{1}{\frac{\rho^N}{(N-1)!(N-\rho)} + \sum_{j=1}^{N-1} \frac{\rho^j}{j!}}$$

$$SL = 1 - E_c(N, \rho) \cdot (e^{-(N-\rho) \cdot AWT \cdot \mu})$$

$$ASA = \frac{E_c(N, \rho)}{(N-\rho) \cdot \mu}$$

Nota: El AWT (*Average Wait Time*) hace referencia al tiempo promedio de espera en la cola, en el cual se incluye el tiempo de las llamadas que ingresaron a la cola y fueron abandonadas.

Contrario a otros modelos donde los servicios bloqueados se consideran como perdidos, en el modelo Erlang C las peticiones que no se pueden atender inmediatamente son puestas en cola hasta que haya un servidor disponible.

Una de las limitaciones de este modelo es que asume que los llamantes esperan en la cola de manera indefinida hasta que la llamada es atendida por un agente. En la realidad algunos llamantes cuelgan apenas ingresan a la cola y otros duran un tiempo considerable hasta que son atendidos.

3.3.2.2 **Modelo Erlang B**

El modelo Erlang B tiene en cuenta el bloqueo de llamadas cuando no hay servidores disponibles presentándose una denegación de servicio, lo que implica que la petición debe realizarse de nuevo si quiere ser atendida. En el caso telefónico, cuando todas las líneas están ocupadas el usuario recibe un tono de ocupado y debe colgar e intentar marcar nuevamente hasta que haya un servidor disponible.

Conociendo el tráfico en Erlangs durante la hora pico (A) y la cantidad de líneas telefónicas disponibles (N), este modelo calcula la probabilidad $P_B(N, A)$ de que una llamada en su primer intento sea bloqueada [11]. La probabilidad se define como:

$$P_B(N, A) = \frac{\frac{A^N}{N!}}{\sum_{i=0}^N \frac{A^i}{i!}}$$

3.3.3 Tráfico de Llamadas

De una manera intuitiva podría considerarse que el número de agentes que se necesitan en un Call Center puede calcularse dividiendo el número esperado de llamadas por unidad de tiempo sobre el número esperado de llamadas atendidas por un agente en la misma unidad de tiempo. Si se tiene una tasa de 120 llamadas por hora y cada llamada tiene una duración promedio de 10 minutos, entonces cada agente podría contestar 6 llamadas por hora, por lo tanto se puede concluir que aparentemente con 20 agentes y 20 líneas telefónicas se atendería todo el tráfico de llamadas.

El error de esta apreciación, está en considerar que las llamadas llegan en orden (una tras otra), pues realmente las llamadas llegan en tiempos aleatorios e independientes entre sí (proceso estocástico). En el ejemplo anterior, el promedio de las llamadas es de 10 minutos, pero debe considerarse que el tiempo de llegada se distribuye aleatoriamente, ya que algunas llamadas llegan al mismo tiempo, otras llegan mientras unas son atendidas y en algunos periodos de tiempo no hay ninguna llamada en el sistema.

3.3.3.1 Llegada de Llamadas

La probabilidad de llegada de una llamada es modelada por medio de un proceso de Poisson. Esta distribución es discreta y se define sólo a través del parámetro λ que representa el promedio de llegadas al sistema (media), y es muy útil para caracterizar eventos que ocurren en el tiempo, tales como: solicitud de llamadas, cantidad de clientes que entran a un establecimiento, cantidad de tareas que requieren CPU, etc. La distribución de Poisson expresa la probabilidad de que ocurran un número de eventos en un periodo de tiempo determinado [11]:

$$p(k, \lambda) = \frac{\lambda^k \cdot e^{-\lambda}}{k!}$$

Donde:

- k : Número de llamadas.
- λ : Promedio de llamadas que ingresan al sistema por unidad de tiempo.

La Figura 7 muestra la distribución de probabilidad de Poisson:

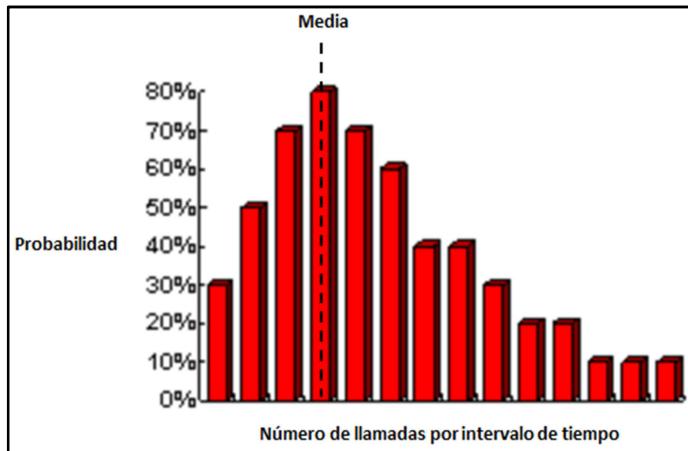


Figura 7. Distribución de llegada de llamadas

En la gráfica anterior se puede apreciar que la *media* representa la máxima probabilidad de que un número k de llamadas ingresen al sistema en un intervalo de tiempo determinado. Por lo tanto, la probabilidad de ocurrencia es menor cuando el número de llamadas que llegan al sistema se aleja de la media.

3.3.3.2 Tiempo de Servicio de las Llamadas

La duración de la atención de las llamadas tampoco es uniforme. La duración es modelada mediante una distribución exponencial (ver Figura 8). La mayoría de las llamadas son más cortas que el promedio de duración y algunas pocas más largas que dicho promedio.

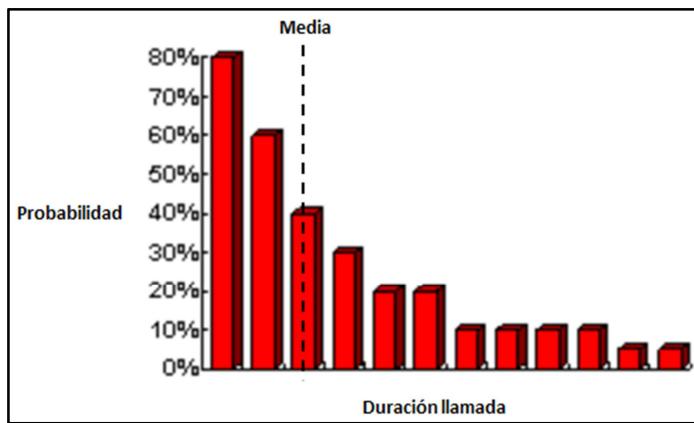


Figura 8. Distribución de duración de llamadas

4. BALANCEO DE CARGA EN SISTEMAS DE CALL CENTER

En los sistemas de Call Center, el balanceo de carga de llamadas es una característica que permite aumentar la capacidad y a su vez mejorar el rendimiento del sistema, pues las llamadas se distribuyen en múltiples servidores donde se conectan los agentes (servidores de ACD), tal y como se muestra en la Figura 9:

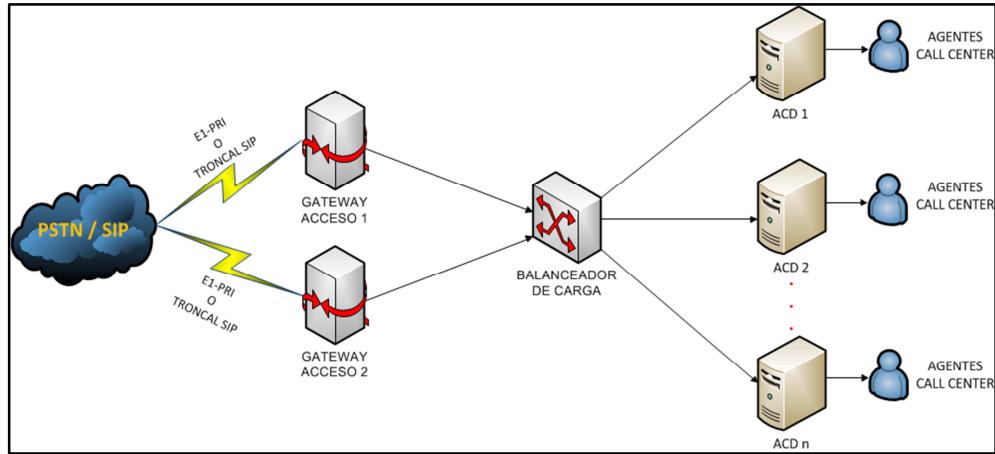


Figura 9. Arquitectura de un sistema de Call Center con múltiples servidores de ACD

En este tipo de arquitecturas, las llamadas ingresan desde la Red de Telefonía Pública Comutada (PSTN, en inglés *Public Switched Telephone Network*) o Troncal SIP hacia los *gateways* de acceso utilizando tecnologías como E1⁴ o SIP⁵. Luego estas llamadas son enviadas a un balanceador de carga, que es el encargado de distribuirlas (utilizando un algoritmo) entre múltiples servidores de ACD.

4.1 ALGORITMOS DE BALANCEO DE CARGA

Existen algoritmos de balanceo de carga tales como: *Generalized Round Robin* (GRR), *Join Shortest Queue* (JSQ), *Minimum Expected Delay* (MED) y *Minimum Average Speed Answer* (Min ASA) [4, 5], los cuales permiten la distribución de llamadas tanto de entrada como de salida entre múltiples servidores de ACD o entre un conjunto de Call Center Virtuales (VCC). A continuación se presenta una descripción detallada de estos algoritmos:

4.1.1 Generalized Round Robin (GRR)

Utiliza la técnica de turno rotativo (Round Robin) para asignar las llamadas a los diferentes servidores de ACD. Este algoritmo envía llamadas a todos los servidores de agentes de manera

⁴ La tecnología E1 es un servicio dedicado de telefonía digital que maneja un tráfico de 2 Mbps, soportando hasta 30 canales de voz simultáneos.

⁵ *Session Initiation Protocol* (SIP): Es un protocolo de control y señalización usado mayoritariamente en los sistemas de Telefonía IP, que fue desarrollado por el IETF (RFC 3261). Dicho protocolo permite crear, modificar y finalizar sesiones multimedia con uno o más participantes y sus mayores ventajas recaen en su simplicidad y consistencia.

equitativa, comenzando por el primer servidor de ACD hasta llegar al último y repitiendo el proceso de manera cíclica. En la siguiente figura se muestra el funcionamiento de este algoritmo:

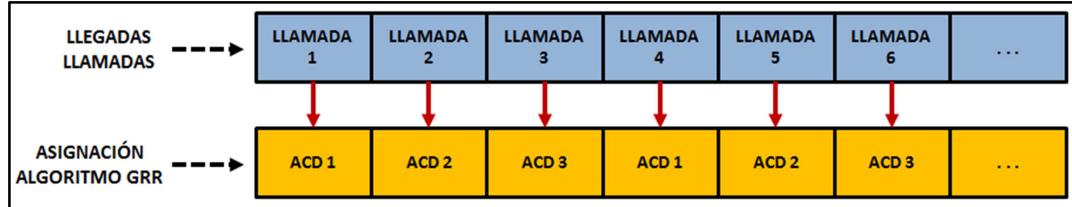


Figura 10. Funcionamiento del algoritmo GRR

4.1.2 *Join Shortest Queue (JSQ)*

Este algoritmo se encarga de enviar las llamadas al servidor de Call Center o servidor de ACD con menor número de llamadas en cola, es decir, que si se tienen 3 servidores de ACD y en el ACD1 hay dos llamadas en cola, en el ACD2 hay cuatro llamadas en espera y el ACD3 existe una sola llamada en cola, entonces el algoritmo dirigirá la llamada al ACD3.

De acuerdo con [4], JSQ es frecuentemente utilizado como un mecanismo de balanceo de llamadas y bajo ciertas circunstancias minimiza el tiempo individual de espera de cada cliente.

4.1.3 *Minimum Expected Delay (MED)*

El algoritmo MED calcula en tiempo real y sobre un histórico parcial, el tiempo que debe esperar un cliente en la cola para ser atendido y selecciona el servidor de ACD que tenga el menor valor. Este algoritmo determina el tiempo de retraso esperado utilizando la siguiente fórmula [6]:

$$ED = ((CallsQNow + 1) * AHT(n)) / \text{Max}(Agents Talking [OR] Ready)$$

Donde:

- ***CallsQNow***: Representa el número actual de llamadas en cola.
- **+ 1**: Es utilizado para indicar que la llamada actual ingresará potencialmente a la cola.
- ***AHT(n)***: Se define como el tiempo promedio de servicio en segundos durante el intervalo de los últimos *n* minutos. Esta variable se calcula de la siguiente manera:

$$AHT(n) = HandleTime(n) / CallsHandled(n)$$

- ***HandleTime***: Representa la duración total de un servicio. Incluye el tiempo de la llamada más cualquier tiempo de postproceso.
- ***Max(Agents Talking [OR] Ready)***: En el denominador se utiliza el valor del número de agentes con llamada o el número de agentes en el sistema, seleccionando el mayor valor entre estos dos.

Una variación de este algoritmo es introducir el concepto de promedio móvil exponencial (EMA: *Exponential Moving Average*⁶) para realizar el cálculo del tiempo promedio de servicio (AHT). De esta manera la fórmula anterior quedaría así:

$$ED = ((CallsQNow + 1) * AHT(EMA)) / Max(Agents Talking [OR] Ready)$$

El cálculo de ED (*Expected Delay*) es válido solo si no hay agentes disponibles para contestar las llamadas. Si hay agentes disponibles el valor de ED es igual a cero.

4.1.4 **Minimum Average Speed of Answer (Min ASA)**

Este algoritmo calcula el tiempo promedio de espera de cada uno de los servidores de ACD y envía la llamada al servidor que tenga el menor tiempo promedio. El cálculo del ASA puede realizarse sobre un intervalo de los últimos n minutos o haciendo uso del promedio móvil exponencial. La fórmula es la siguiente [5]:

$$\text{Min ASA} = \text{Min}(*.\text{AvgSpeedAnswer})$$

Donde:

- *: Hace referencia a todos los servidores de ACD.
- **AvgSpeedAnswer:** Tiempo promedio de espera en cola.

⁶ *Exponential Moving Average (EMA)*: Es una herramienta aritmética que permite calcular el promedio de una serie de valores, dando más peso a los datos obtenidos recientemente.

5. ARQUITECTURA DE ASTERISK VERSIÓN 11.X

Asterisk es una plataforma de software libre (bajo licencia GPL⁷) que proporciona funcionalidades de una central telefónica (PBX). Como cualquier PBX, se puede conectar un número determinado de teléfonos (*hardphones o softphones*) para hacer llamadas entre sí e incluso conectarse a un proveedor de VoIP (Voz sobre IP), a líneas analógicas convencionales o a una Red Digital de Servicios Integrados (RDSI⁸).

Asterisk puede ser instalado sobre múltiples sistemas operativos como Mac OX, Windows o FreeBSD, sin embargo está soportado principalmente para Linux. Fue desarrollado por Mark Spencer en el año de 1999 y actualmente cuenta con miles de usuarios a nivel mundial.

5.1 FUNCIONALIDADES

Asterisk proporciona una serie de funcionalidades básicas y avanzadas que la convierten en una de las soluciones de software libre más completas del mercado [13]:

Funcionalidades Básicas	Funcionalidades Avanzadas
<ul style="list-style-type: none">▪ Llamada en espera.▪ Transferencia.▪ Identificador de llamada.▪ Bloqueo de Caller ID.▪ Timbres distintivos.▪ Música en espera.▪ Música en transferencia.▪ Salas de conferencia.▪ Buzón de voz personal.▪ Colas de llamada.▪ Colas con prioridad.▪ Registro de llamadas en Base de Datos.▪ Buzón de voz por Mail.▪ Pickup de llamadas.▪ Desvío de llamadas si el usuario está ocupado.▪ Desvío de llamadas si el usuario no responde.	<ul style="list-style-type: none">▪ IVR (<i>Interactive Voice Response</i>): Gestión de llamadas a través menús interactivos.▪ LCR (<i>Least Cost Routing</i>): Direccionamiento de llamadas por el proveedor más económico.▪ ACD (<i>Automatic Call Distributor</i>): Es una tecnología de Call Center (hardware o software) que permite distribuir eficientemente las llamadas a un grupo de agentes. El ACD permite poner en cola las llamadas cuando los agentes se encuentran ocupados. Durante su funcionamiento permite el registro detallado de las de llamadas recibidas, atendidas, abandonadas, etc., así como el registro de los tiempos de las mismas.▪ AGI (<i>Asterisk Gateway Interface</i>): Integración con todo tipo de aplicaciones externas.▪ AMI (<i>Asterisk Manager Interface</i>): Gestión y control remoto de Asterisk.▪ Configuración en BD: Usuarios, extensiones, etc.

Tabla 1. Funcionalidades básicas y avanzadas de Asterisk

⁷ General Public License (GPL): Es la licencia de software más utilizada a nivel mundial. Garantiza a los usuarios finales la libertad de usar, estudiar, compartir y modificar el software. El software cubierto por esta licencia, es libre y está protegido de intentos de apropiación que restrinjan esas libertades a los usuarios.

⁸ Red Digital de Servicios Integrados (RDSI): Es una red que facilita conexiones digitales extremo a extremo para proporcionar una amplia gama de servicios, tanto de voz como de otros tipos, y a la que los usuarios acceden a través de un conjunto de interfaces normalizadas.

5.2 ARQUITECTURA BÁSICA

Asterisk presenta una arquitectura modular que depende de un núcleo principal denominado **PBX Core**, el cual permite la carga dinámica de los módulos, lee los archivos de configuración, ejecuta aplicaciones, procesa las peticiones del plan de marcación, crea instancias de los canales y realiza la conversión de formatos, códigos y protocolos. En la siguiente figura se puede observar la arquitectura básica de esta plataforma:

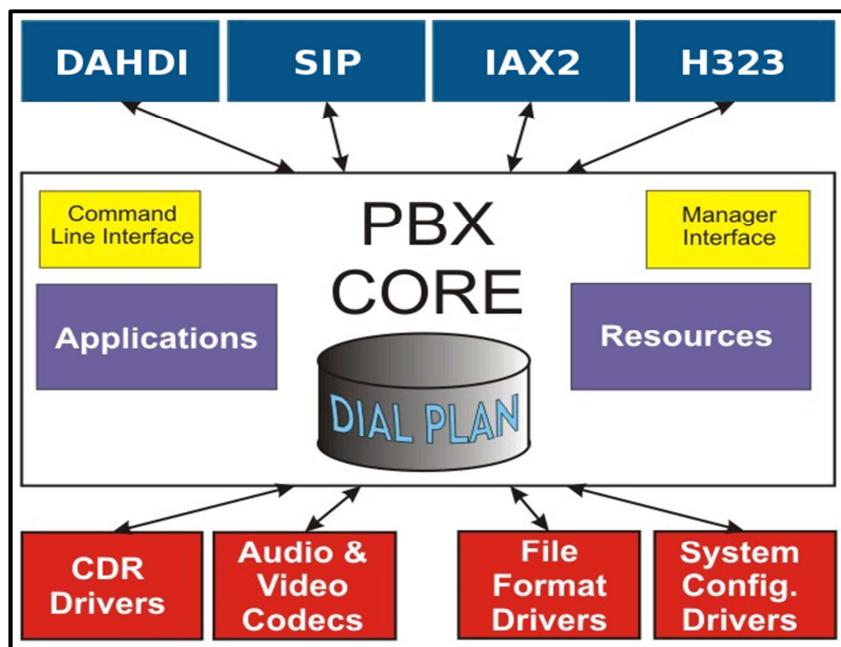


Figura 11. Arquitectura básica de Asterisk [14]

5.2.1 Módulos

Como se mencionó anteriormente, Asterisk está basado en módulos independientes, los cuales pueden ser cargados y descargados de acuerdo con las funcionalidades que el administrador desea proveer al sistema. Cada módulo posee una funcionalidad específica, de tal forma que pueden gestionarse todos los aspectos del sistema, pasando por los tipos de canales (SIP, IAX2⁹, DAHDI¹⁰), hasta las conexiones de otros sistemas que interactúan con Asterisk (mail, bases de datos, aplicaciones web, etc.).

Existe un archivo específico para la configuración de los módulos denominado *modules.conf* que se encuentra ubicado en el directorio */etc/asterisk/*.

⁹ *Inter-Asterisk Exchange Protocol* (IAX2): Es un protocolo utilizado para manejar conexiones de VoIP entre servidores Asterisk, y entre clientes y servidores que soportan este protocolo. Permite el manejo de múltiples códigos y gran número de streams. IAX2 utiliza un único puerto UDP (generalmente el 4569) tanto para la señalización como para los datos.

¹⁰ *Digium Asterisk Hardware Device Interface* (DAHDI): Es una tecnología de código abierto utilizada para controlar una serie de productos de Digium (Empresa patrocinadora de Asterisk), principalmente tarjetas de telefonía para enlaces digitales y líneas analógicas.

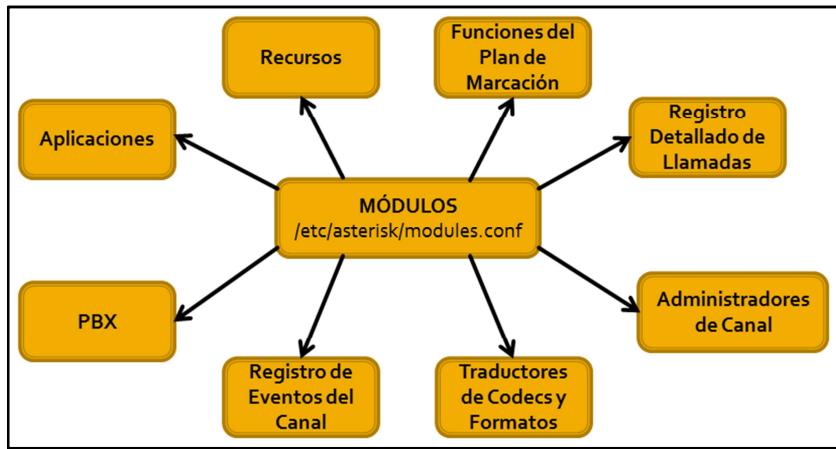


Figura 12. Módulos de Asterisk

5.2.1.1 *Aplicaciones*

Las aplicaciones del plan de marcación (*dialplan applications*) son utilizadas en el archivo *extensions.conf* para definir una serie de acciones que pueden ser aplicadas a una llamada. La aplicación *Dial()*, por ejemplo, es la encargada de realizar las conexiones con los recursos externos y es considerada una de las más importantes del plan de marcación.

Se han desarrollado aplicaciones básicas para el establecimiento de llamadas telefónicas, tales como: *Answer()*, *Dial()*, *Busy()*, *Hangup()*, *Ringing()* y otras de mayor complejidad como por ejemplo *Voicemail()* para el manejo de los buzones de voz, *Queue()* para el manejo de colas de espera, entre otras.

5.2.1.2 *Recursos*

La función específica de los recursos es la de integrar a Asterisk con otros sistemas externos entre los que encontramos: bases de datos, servidores web, calendarios, etc. Los recursos se cargan de manera estática y pueden operar de manera simultánea sobre múltiples canales, en lugar de crearse dinámicamente para cada canal en curso.

Algunos de los más comunes son: el recurso para ofrecer servicios de música de espera (*Music On Hold*) y el recurso para realizar interconexiones con bases de datos a través de ODBC (*Open Data Base Connectivity*).

5.2.1.3 *Funciones del Plan de Marcación*

La idea fundamental detrás de las funciones del plan de marcación (*dialplan functions*), es la capacidad de obtener o añadir determinada información específica a cada canal. Suelen ser complementarias a las aplicaciones y son capaces de ofrecer mejoras para determinados aspectos del sistema, que de por sí pudieran ser limitados.

5.2.1.4 *Registro Detallado de Llamadas (CDR)*

Los módulos de CDR (*Call Detail Record*) fueron diseñados para facilitar el almacenamiento detallado de las llamadas. Asterisk puede almacenar el detalle de las llamadas en archivos de texto plano y/o en una base de datos externa.

5.2.1.5 Administradores de Canal

Los administradores de canal (*channel drivers*), son los *drivers* específicos para cada tipo de canal disponible en la plataforma y son los que permiten realizar llamadas. Cada canal tiene asociado un protocolo específico como SIP, DAHDI, IAX2, entre otros. El módulo de canal actúa como una puerta (*gateway*) hacia el núcleo de Asterisk.

5.2.1.6 Traductores de Codecs y Formatos

Los traductores de codecs son los encargados de convertir (vía software) un tipo de códec a otro, de forma simultánea al curso de la llamada. Por ejemplo, si una llamada viene del canal DAHDI con el códec G.711 y quiere pasarse a una extensión SIP que únicamente tiene habilitado el códec G.729, el traductor de codecs será el encargado de realizar esta conversión en tiempo real. Este proceso recibe el nombre de *transcoding*.

Por otro lado, los traductores o intérpretes de formatos realizan la misma función de los traductores de codecs, pero hacen su trabajo sobre archivos. Por ejemplo, si el sistema tiene una grabación dentro de un menú en formato GSM, se utilizará un intérprete de formato para poder reproducir la grabación en cualquier canal que no soporte el códec GSM.

5.2.1.7 Registros de Eventos del Canal

Los registros de eventos de canal (*Channel Event Logging*) proporcionan un poderoso sistema de control de la actividad de las llamadas. Por ello es indispensable una cuidadosa configuración del plan de marcación.

5.2.1.8 PBX

Los módulos de PBX, son módulos periféricos que ofrecen mecanismos de control y configuración. Por ejemplo, *pbx_config* es el módulo que carga el plan de marcación tradicional de Asterisk.

5.2.2 Interfaces

Asterisk provee múltiples interfaces que permiten su configuración, administración y comunicación con otras aplicaciones y/o plataformas. A continuación se muestran estas interfaces:

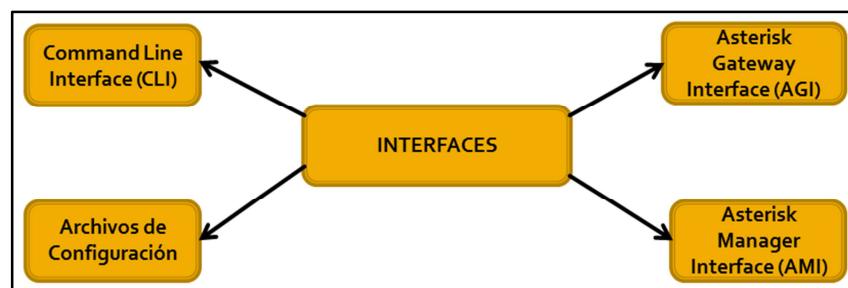


Figura 13. Interfaces de Asterisk

5.2.2.1 Command Line Interface (CLI)

La interfaz de línea de comandos es la consola que permite realizar una variedad de tareas de administración de la plataforma, tales como:

- Obtener información de los componentes del sistema.
- Afectar la configuración del sistema.
- Observar *logs*, errores y advertencias en tiempo real.
- Generar llamadas de prueba.
- Conocer documentación del sistema (APIs, aplicaciones, funciones, módulos, etc.).

5.2.2.2 Asterisk Gateway Interface (AGI)

Possiblemente una de las funcionalidades más interesantes es la capacidad de invocar aplicaciones externas al proceso de Asterisk a través de eventos telefónicos, con el fin de intercambiar datos entre ambos procesos. Esta interfaz se denomina AGI (*Asterisk Gateway Interface*) y su variante FASTAGI que ejecuta el proceso externo en otra máquina diferente a Asterisk por medio de un *socket* TCP/IP.

Para que Asterisk pueda lanzar la aplicación externa, es necesario definir en el plan de marcación (archivo *extensions.conf*) una extensión que invoque al AGI y le pase como parámetro el nombre del archivo que se desea ejecutar, el cual puede estar escrito en cualquier lenguaje de programación.

5.2.2.3 Asterisk Manager Interface (AMI)

Es una interfaz de administración que permite monitorear los eventos que ocurren en el sistema y realizar solicitudes a través de acciones. Existe una amplia gama de acciones que permiten obtener información del estado de la plataforma y también realizar tareas, como por ejemplo originar nuevas llamadas.

La interfaz AMI utiliza el modelo cliente – servidor y permite que aplicaciones externas se conecten a Asterisk mediante un *socket* haciendo uso del protocolo TCP/IP. Sistemas como marcadores predictivos, paneles de control y visualización, sistemas de facturación, etc., hacen uso de esta interfaz.

Existen dos tipos de mensajes en la interfaz AMI: eventos del *manager* (*manager events*) y acciones del *manager* (*manager actions*). Los eventos del *manager* son mensajes que van en una sola dirección y que son generados por Asterisk hacia la conexión AMI del cliente, con el fin de reportar lo que está ocurriendo en el sistema. Por otro lado, las acciones del *manager* son solicitudes realizadas por el cliente hacia el Asterisk. Dichas solicitudes tienen asociada una respuesta e implican que el sistema ejecute una acción.

Los mensajes procesados por la interfaz AMI tienen una estructura definida. Están basados en un protocolo línea a línea del tipo *llave: valor*. Cada línea es terminada por un salto de línea representado con *\r\n* y el bloque de datos completo finaliza con un doble salto de línea representado por *\r\n\r\n*. De esta manera es posible identificar un paquete completo que se lee desde el *socket*.

El archivo de configuración *manager.conf* está ubicado en el directorio */etc/asterisk/* y permite la creación de cuentas que harán uso de esta interfaz.

Existen tres tipos de paquetes que se identifican con una palabra reservada, estos pueden ser de tipo *Action*, *Response* y *Event*:

- *Action*: Esta palabra reservada indica que el paquete es una petición de un cliente externo con el propósito de que una acción en particular sea procesada por Asterisk. El paquete contiene el nombre de la acción que se desea ejecutar, seguido de los parámetros relacionados con dicha acción. Solo se puede ejecutar una acción a la vez y en caso de que existan varias, serán puestas en cola para su posterior ejecución.
- *Response*: Esta cabecera indica que el paquete es una respuesta emitida por Asterisk de acuerdo a una acción previamente enviada.
- *Event*: Esta cabecera indica que el paquete es un evento generado por el AMI hacia el cliente. Estos eventos son generados por las diferentes aplicaciones de Asterisk.

5.2.2.4 Archivos de Configuración

Toda la configuración de Asterisk se realiza a través de archivos de texto terminados con la extensión *.conf* que se encuentran localizados en el directorio */etc/asterisk/*. Cada uno de los archivos de texto permite al núcleo y a los diferentes módulos cambiar su configuración. Los archivos son utilizados para un propósito determinado y sirven para configurar cada uno de los módulos. Los archivos más usados y su descripción se muestran en Anexo A-1.

Todos los archivos de configuración tienen una estructura definida y se interpretan de arriba hacia abajo. Están organizados en unidades lógicas denominadas secciones o contextos que se representan así: *[nombre_sección]*. Una sección termina cuando otra es definida. Casi todos los archivos tienen una sección *[general]* donde se definen los parámetros generales de un módulo en particular. Para cada uno de los módulos, Asterisk implementa una función que se encarga de interpretar y verificar la sintaxis del archivo de configuración correspondiente.

5.2.3 Plan de Marcación

El plan de marcación (*dialplan*) se define en el archivo *extensions.conf* y constituye la columna vertebral de la arquitectura de Asterisk, pues indica cual es el flujo de las llamadas y que acciones deben realizar. Todas las llamadas que llegan al sistema pasan por el plan de marcación.

6. MODELO CONCEPTUAL DEL SISTEMA

6.1 DESCRIPCIÓN DEL MODELO CONCEPTUAL

Las arquitecturas tradicionales para Call Center están compuestas por dispositivos de hardware especializados en determinadas funciones (IVR, Grabaciones, ACD, Balanceador de Carga, etc.), que interactúan de manera conjunta para conformar el sistema. En este proyecto se propone una arquitectura separada lógicamente en etapas, cuyo objeto es distribuir en cada una de ellas tareas específicas complementarias. En la Figura 14 se muestra un diagrama de bloques funcionales de un sistema de Call Center con balanceo de carga y en la Figura 15 se ilustra un diagrama de flujo del proceso de balanceo de carga para llamadas de entrada:

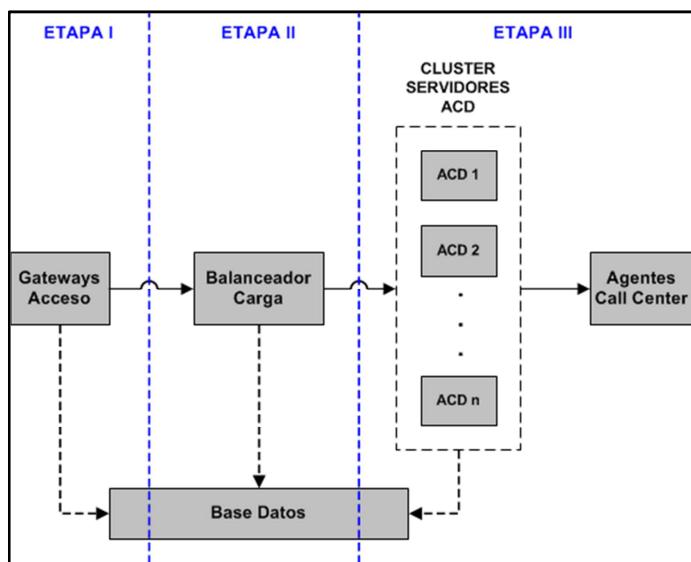


Figura 14. Diagrama de bloques funcionales de un sistema de Call Center con balanceo de carga

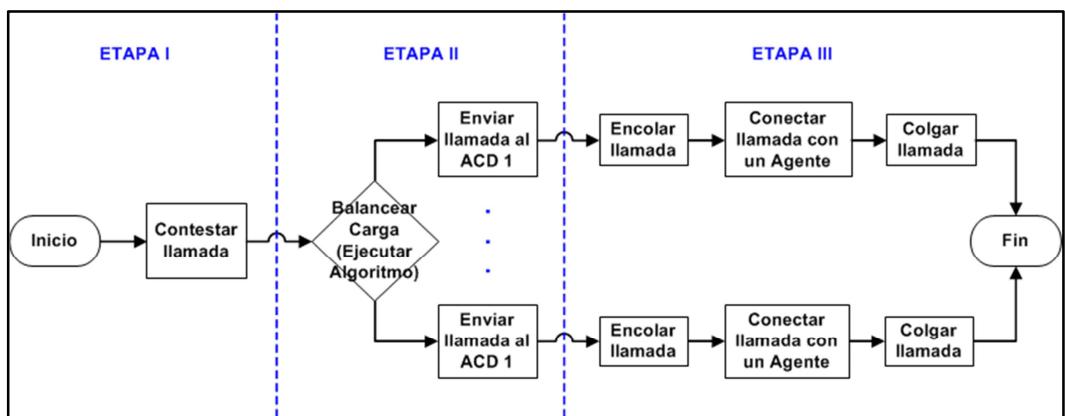


Figura 15. Diagrama de flujo del proceso de balanceo de carga para llamadas de entrada

En la primera etapa se realiza el proceso de ingreso de llamadas a través de *gateways* de acceso y también se hace el proceso de transcodificación de la llamada en caso de ser necesario.

La segunda etapa es la encargada del enruteamiento de las llamadas teniendo en cuenta los algoritmos de balanceo de carga GRR, JSQ, MED y Min ASA. Cada uno de los algoritmos fue implementado por separado, con el fin de evaluar su desempeño. En esta etapa se centra la mayor parte de este proyecto.

Finalmente, la tercera etapa es la de aplicaciones y/o funcionalidades. Aquí se encuentran los agentes, el ACD nativo de Asterisk, el monitoreo en línea de llamadas, la grabación digital y demás servicios que Asterisk pueda suministrar.

6.2 ARQUITECTURA PROPUESTA DEL SISTEMA

En la Figura 16 se muestra la arquitectura propuesta para un sistema de Call Center multicola implementado sobre la plataforma de software libre Asterisk, el cual incluye un平衡ador de carga de llamadas:

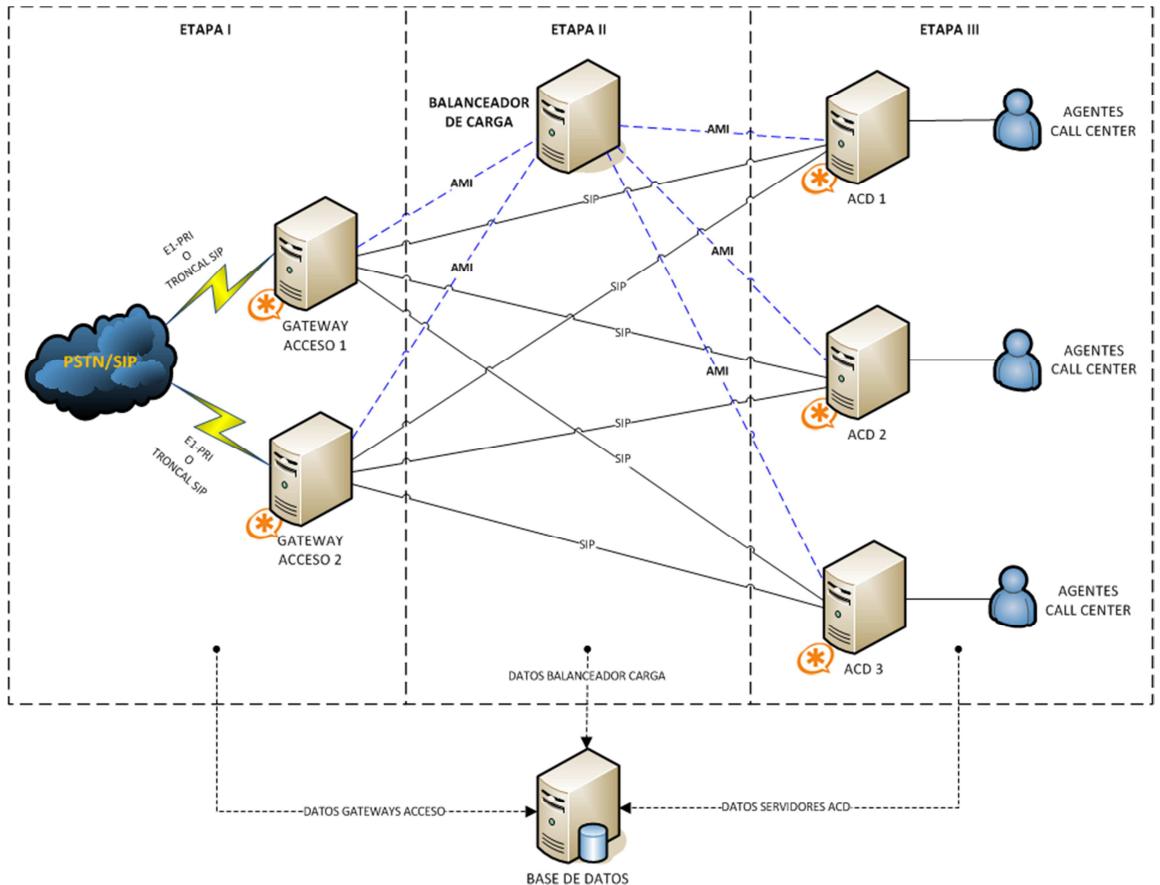


Figura 16. Arquitectura propuesta del sistema

El balanceador de carga crea una conexión vía **AMI (Asterisk Manager Interface)** con cada uno de los servidores Asterisk del sistema, tanto a los *gateways* de acceso como a los servidores de *ACD*.

Gracias a esta conexión es posible leer eventos del AMI y conocer en tiempo real las siguientes variables:

- Llamadas que ingresan al sistema.
- Agentes conectados.
- Agentes disponibles.
- Llamadas en cola.
- Tiempo promedio de espera (ASA).
- Tiempo promedio de servicio (AHT).
- Tiempo de espera de la primera llamada en cola.

Adicionalmente, el balanceador de carga recibe los eventos generados por la llegada de las llamadas a los servidores de acceso. Luego ejecuta el algoritmo de balanceo de carga y decide a qué servidor de ACD direcciona la llamada. La llamada es direccionada al servidor seleccionado y enviada al ACD local de dicha máquina.

Por otro lado, a cada servidor de ACD se conectan los agentes u operadores, que son los encargados de atender las llamadas que ingresan a la plataforma.

7. SIMULACIÓN DE LOS ALGORITMOS DE BALANCEO DE CARGA GRR, JSQ, MED Y MIN ASA

Con el objetivo de poder verificar el funcionamiento de los algoritmos, se construyó un modelo de simulación utilizando la herramienta **SimEvents** de MATLAB¹¹. Esta herramienta cuenta con una serie de bloques de construcción gráficos que permiten modelar un sistema de colas (p. ej. un sistema de Call Center) bajo el entorno de Simulink¹².

7.1 MODELO DE SIMULACIÓN

Para realizar la simulación de los algoritmos de balanceo de carga GRR, JSQ, MED y Min ASA se utilizó una arquitectura como la que se muestra en la sección 4 (Figura 9). En la Figura 17 se puede observar el modelo de simulación implementado:

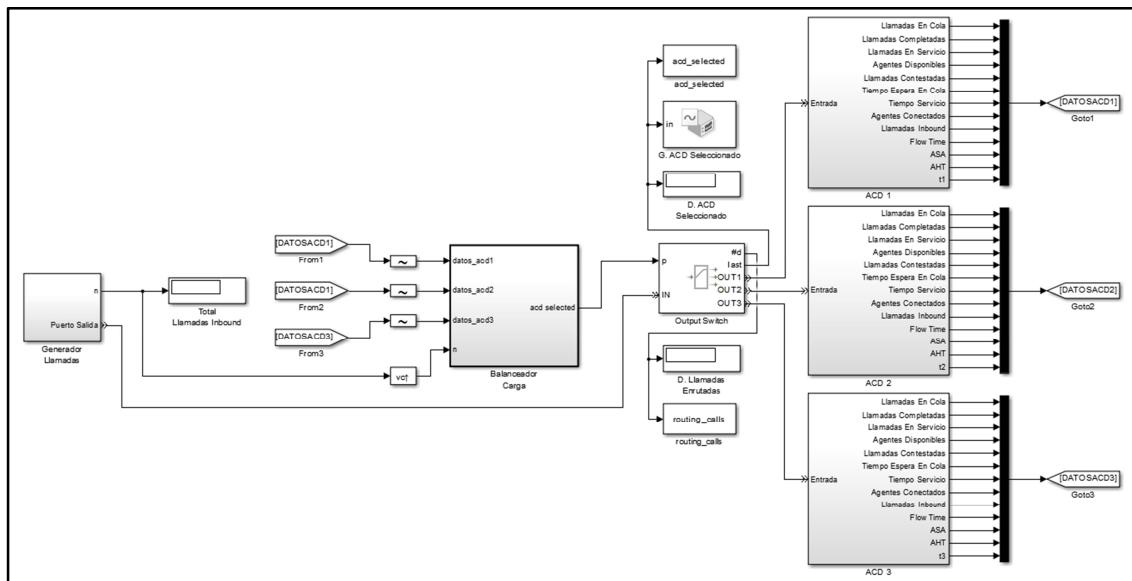


Figura 17. Modelo de simulación de un Call Center que incluye un balanceador de carga

Las llamadas de los clientes son generadas por un subsistema denominado *Generador Llamadas*. Cada llamada es enviada al bloque *Balanceador Carga* donde se realiza la configuración de los algoritmos que permiten la distribución de llamadas entre múltiples servidores de ACD, los cuales están representados por los subsistemas ACD1, ACD2 y ACD3.

En la siguiente sección se describe detalladamente cada uno de los bloques o subsistemas que conforman el modelo de simulación implementado.

¹¹ MATLAB es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio.

¹² Simulink es un entorno de programación visual de alto nivel de abstracción, que funciona sobre el entorno de programación MATLAB.

7.1.1 Subsistema Generador Llamadas

Este subsistema es el encargado de la generación de llamadas que ingresan a la plataforma de Call Center.

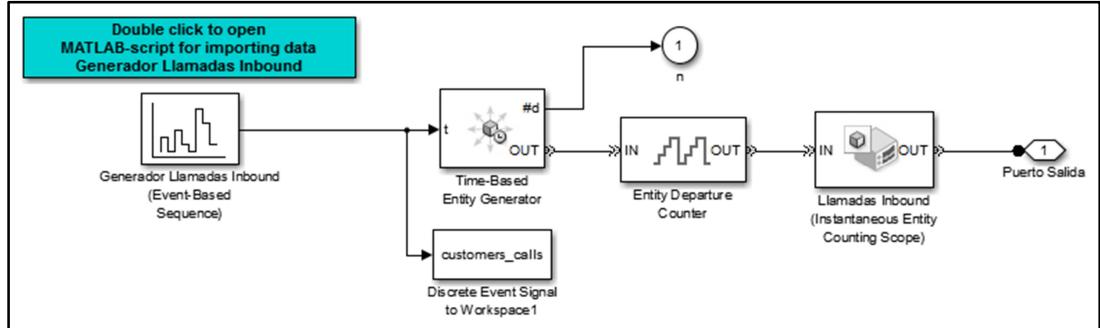


Figura 18. Subsistema Generador Llamadas

Las llamadas de entrada son generadas por un bloque *Event-Based Sequence* que es el encargado de leer el archivo que contiene los tiempos de llegada de las llamadas reales, combinado con un bloque generador de entidades (*Time-Based Entity Generator*).

7.1.2 Subsistema Balanceador Carga

Es el subsistema más importante del modelo de simulación, ya que aquí se configuran los algoritmos de balanceo de carga GRR, JSQ, MED y Min ASA, los cuales permiten la distribución de las llamadas entre varios servidores donde se conectan los agentes. Cada uno de estos algoritmos maneja una lógica, entradas y condiciones diferentes, que generan múltiples escenarios de simulación y permiten la validación de su funcionamiento.

7.1.2.1 Algoritmo GRR

Para simular este algoritmo se utiliza el bloque denominado *Output Switch* tal y como se muestra en la siguiente figura:

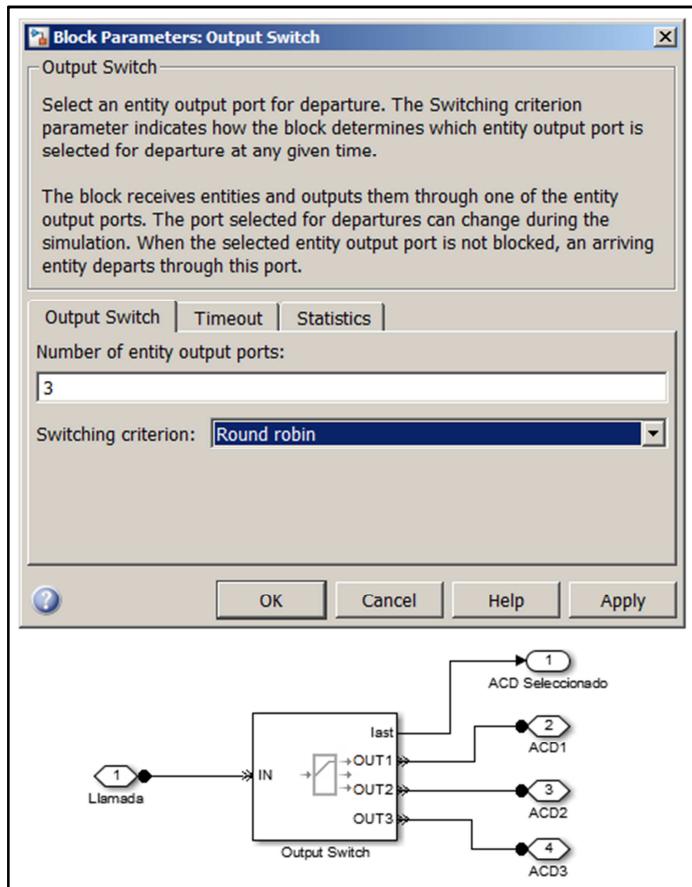


Figura 19. Subsistema Balanceador Carga – Algoritmo GRR

7.1.2.2 *Algoritmo JSQ*

Para la simulación de este algoritmo se utilizó un bloque *MATLAB Function* al cual le ingresa el número de llamadas en cola de cada uno de los servidores de ACD y determina cual es el menor.

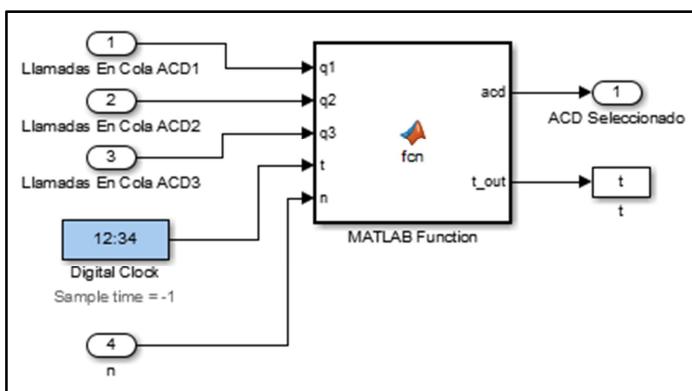


Figura 20. Subsistema Balanceador Carga – Algoritmo JSQ

La función que determina cual es el ACD que presenta el menor número de llamadas en cola es:

```

function [acd,t_out] = fcn(q1,q2,q3,t,n)

%Se declaran las variables persistentes
persistent last_n;

%Variables
acd = 0;

%Se inicializan las variables persistentes
if isempty(last_n)
    last_n = 0;
end

%Tiempo de simulación
t_out = t;

if(n > last_n)
    last_n = n;

    %Se determina cual es el ACD que tiene menor tamaño de cola
    [~, acd] = min([q1 q2 q3]);
end

```

Figura 21. Código en MATLAB del algoritmo JSQ

7.1.2.3 Algoritmo MED

Este algoritmo determina el retardo esperado (ED: *Expected Delay*) de un cliente en cola antes de ser atendido y selecciona el servidor de ACD que tenga el menor valor.

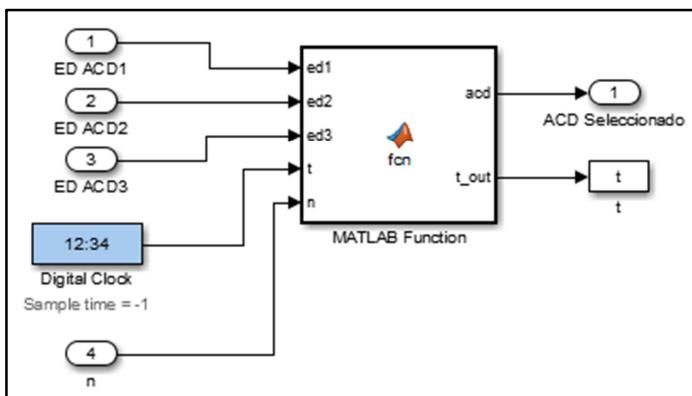


Figura 22. Subsistema Balanceador Carga – Algoritmo MED

El cálculo del retardo esperado se realizó teniendo en cuenta el promedio móvil exponencial (EMA) del tiempo de servicio, el número de llamadas en cola y el número de agentes registrados en el sistema. La fórmula aplicada fue la siguiente:

$$ED = ((CallsQNow + 1) * AHT(EMA)) / Agents\ Ready$$

También se tuvo en cuenta que el cálculo de ED es válido solo si no hay agentes disponibles para contestar las llamadas. Si hay agentes disponibles el valor de ED es igual a cero.

7.1.2.4 Algoritmo Min ASA

Para este algoritmo se calculó el promedio móvil exponencial (EMA) del tiempo de espera en cola, ya que de esta manera se dará más peso a los tiempos de espera más recientes. Una vez obtenido este valor para cada uno de los servidores de ACD, la llamada es direccionada al servidor que tenga el menor valor. Cuando hay agentes disponibles, el algoritmo envía la llamada al servidor de ACD con mayor número de agentes disponibles. En la Figura 23 se muestra el bloque *MATLAB Function* en donde se determina el menor tiempo promedio de espera:

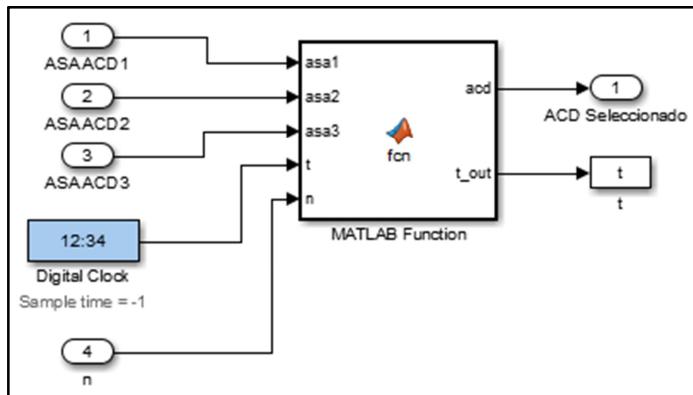


Figura 23. Subsistema Balanceador Carga – Algoritmo Min ASA

7.1.3 Subsistema ACD

Básicamente, este subsistema está conformado por un conjunto de agentes u operadores que están disponibles para contestar las llamadas de los clientes. Cuando la llamada es finalizada, el operador es devuelto al *pool* de operadores disponibles y es asignado a una nueva llamada. La duración de las llamadas de los clientes varía de manera aleatoria al igual que el tiempo que el cliente espera en cola.

Para generar múltiples entidades (operadores del Call Center) al iniciar la simulación, se utiliza un bloque *Event-Based Sequence* en combinación con un bloque *Time-Based Entity Generator*. Estas entidades son enviadas a un bloque *FIFO Queue* que representa la cola de operadores esperando para aceptar nuevas llamadas.

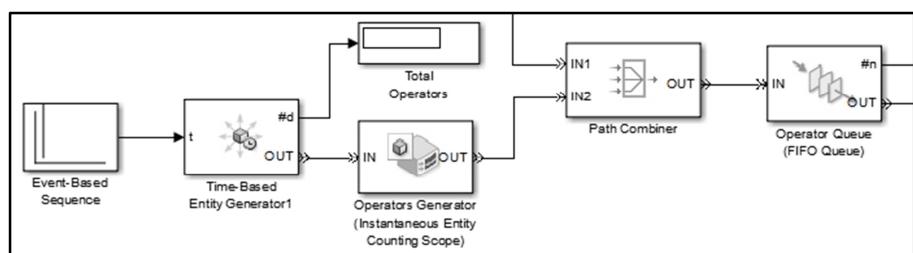


Figura 24. Generación de múltiples operadores del Call Center

El software generará un número n entidades en el tiempo 0. La función del bloque *Path Combiner* es la de retornar un operador a la cola cada vez que el operador complete una llamada.

Las llamadas que ingresan al ACD son enviadas inmediatamente a una cola. Un bloque *Entity Combiner* se encarga de combinar las entidades que provienen de la cola de clientes (*Customer Queue*) con la cola de operadores (*Operators Queue*). Esta acción representa la asignación de un operador disponible a una llamada de un cliente. En la Figura 25 se muestra lo descrito anteriormente:

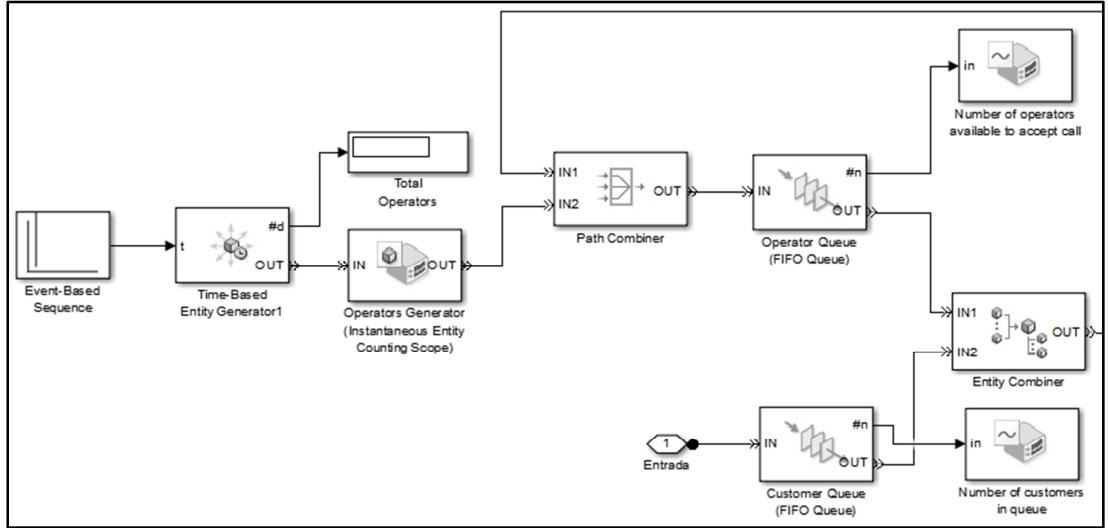


Figura 25. Asignación de un operador disponible a una llamada de un cliente

Las entidades combinadas avanzan hacia un bloque *Infinite Service* que tiene un generador de tiempo de servicio a través de un bloque *Event-Based Random Number*. Esta porción del modelo simula el tiempo de servicio de una llamada de un cliente utilizando una distribución exponencial.

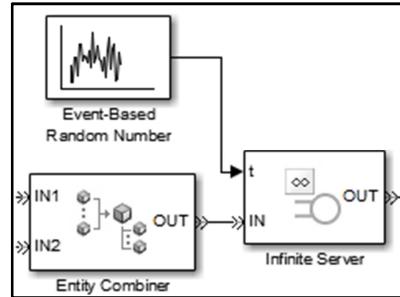


Figura 26. Generador del tiempo de servicio

Cuando un operador completa una llamada, el bloque *Entity Splitter* retorna la entidad que representa al operador a la cola para aceptar una nueva llamada. La entidad que representa al cliente es enviada a un *Entity Sink* para representar la finalización de la llamada. En el siguiente gráfico se muestra el subsistema completo:

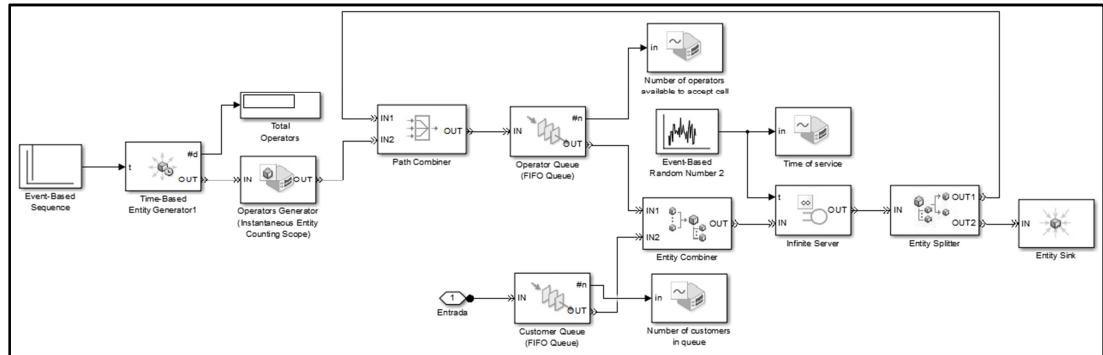


Figura 27. Subsistema ACD completo

En las Figuras 28 y 29 se puede observar el funcionamiento de los operadores y de las llamadas de los clientes a través de los bloques *Signal Scope* conectados en los bloques *Operator Queue* y *Customer Queue*.

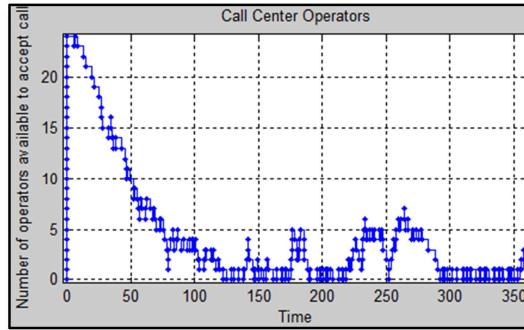


Figura 28. Variación del número de operadores disponibles

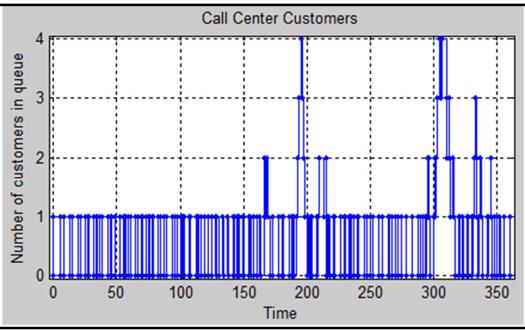


Figura 29. Variación del número de clientes en cola

En la Figura 28 se muestra la variación del número de entidades en la cola de operadores a medida que la simulación progresó. En el tiempo 0, se puede observar que el software genera 25 entidades. Estas entidades representan el número total de operadores que están inicialmente disponibles para aceptar las llamadas de los clientes. Cada operador es asignado a una llamada de un cliente y el número de operadores disponibles se reduce a cero. Debido a que la duración de cada llamada es aleatoria, el número de operadores disponibles aumenta y disminuye.

Por otro lado, en la Figura 29 se muestra como el número de entidades en la cola de clientes cambia en el transcurso de la simulación. Comparando las dos gráficas, se puede ver que en los tiempos cuando el número de operadores disponibles se ha reducido a cero, el número de clientes esperando en la cola se incrementa. Este número disminuye nuevamente cuando los operadores están disponibles.

7.2 CONSIDERACIONES Y CARACTERÍSTICAS

De acuerdo con [24], algunas consideraciones para la simulación de un sistema de Call Center son:

- Considerar varios tipos de llamadas en cuanto a su duración y tasa de llegada.

- El volumen de las llamadas puede calcularse mediante de datos históricos, modelos de series de tiempo o datos basados en experiencias. Generalmente el volumen de las llamadas se modela mediante un proceso de Poisson y el tiempo de servicio de las llamadas mediante una distribución exponencial.
- Considerar un número de agentes variable por cada periodo de simulación.
- Los tiempos de abandono se modelan mediante una variable aleatoria exponencial la cual se denomina el factor de paciencia. No se tiene en cuenta el retorno de llamadas por parte del cliente.

En [4] se consideran otros parámetros adicionales para la simulación de Call Centers Virtuales (VCC), tales como:

- Número de agentes homogéneos y heterogéneos en cada Call Center.
- Selección del tipo de algoritmo de balanceo de carga.

Con base en lo anterior, se fabricó un modelo de simulación que incluye las siguientes características:

- Para la generación de las llamadas se analizó el tráfico de entrada de una semana de un Call Center real y se determinó que presenta un comportamiento similar cualquier día de la semana. En la Figura 30 se puede observar el número de llamadas que ingresa por minuto (puntos de color azul) durante la jornada laboral y cuál es la tendencia de este tráfico (línea de color rojo):

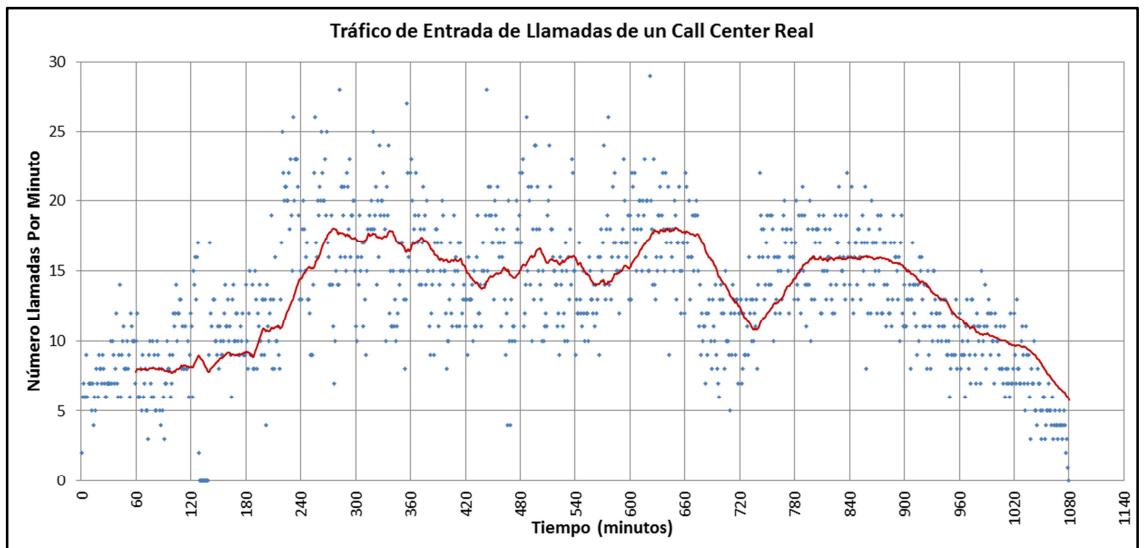


Figura 30. Tráfico de entrada de un Call Center real

También se halló la función de densidad de probabilidad (pdf), la cual describe la probabilidad de que en un minuto ingresen un determinado número de llamadas. En la siguiente tabla se muestra dicha probabilidad:

Llamadas Por Minuto	Ocurrencia	Probabilidad
0	11	1,02%
1	1	0,09%
2	3	0,28%
3	9	0,83%
4	16	1,48%
5	22	2,04%
6	31	2,87%
7	45	4,17%
8	54	5,00%
9	69	6,39%
10	72	6,67%
11	75	6,94%
12	84	7,78%
13	88	8,15%
14	70	6,48%
15	76	7,04%
16	58	5,37%
17	69	6,39%
18	53	4,91%
19	58	5,37%
20	37	3,43%
21	28	2,59%
22	20	1,85%
23	12	1,11%
24	6	0,56%
25	4	0,37%
26	4	0,37%
27	1	0,09%
28	2	0,19%
29	1	0,09%
30	0	0,00%
31	1	0,09%

Tabla 2. Función de densidad de probabilidad del tráfico de entrada

En la tabla anterior se puede observar que la probabilidad de que ingresen 13 llamadas en un intervalo de tiempo de un minuto tiene la mayor probabilidad (8,15%) y la probabilidad de que ingresen más de 25 llamadas es menor al 0,4%.

El tráfico seleccionado muestra claramente la naturaleza estocástica de las llamadas que ingresan al Call Center, convirtiéndolo en una fuente idónea para realizar las simulaciones del sistema y las pruebas del prototipo que incluye el balanceador de carga.

- Cada uno de los algoritmos de balanceo de carga fue simulado por separado bajo dos tipos de escenarios: homogéneo (igual número de agentes en cada servidor) y heterogéneo (diferente número de agentes en cada servidor), con el objetivo de verificar su comportamiento dinámico.
- El tiempo de simulación fue de 18 horas que corresponde a un día laboral del Call Center.
- Para el tiempo promedio de servicio se utilizó un generador aleatorio configurado bajo distribución exponencial con una media de 250 segundos, la cual fue calculada a partir del tiempo de servicio real de la jornada laboral del Call Center.
- En el modelo de simulación se incluyeron 3 servidores de ACD.
- El número de agentes registrados en cada uno de los servidores de ACD para el escenario homogéneo fue 23.
- El número de agentes registrados en cada uno de los servidores de ACD para el escenario heterogéneo fue de 22, 20 y 25 respectivamente.
- Todas las llamadas que ingresan a cada uno de los servidores de ACD fueron contestadas por los agentes (no se simuló el abandono de llamadas).

7.3 PARÁMETROS DE MEDICIÓN

Los parámetros más utilizados para la medición del rendimiento de un Call Center son comentados en [4] [24] y [27]. Con el fin de determinar cuál de los algoritmos funciona mejor en ambos escenarios, se seleccionaron los siguientes:

- Tamaño promedio de la cola.
- Tiempo promedio de espera en cola.
- Utilización de los agentes.

Estos tres parámetros son explicados detalladamente en la sección 3.3.1 de este documento.

7.4 ANÁLISIS DE RESULTADOS

A continuación se muestran los resultados de las simulaciones realizadas bajos los dos tipos de ambientes (homogéneo y heterogéneo), calculando las principales métricas ya mencionadas.

7.4.1 Tamaño Promedio de la Cola

Una métrica muy útil para medir la efectividad del algoritmo de balanceo de carga es el tamaño promedio de la cola en el tiempo. En las Figuras 31, 32 y 33 se puede observar que bajo un escenario homogéneo de 23 agentes conectados a cada uno de los servidores de ACD, los algoritmos JSQ y MED presentan el menor tamaño promedio de la cola.

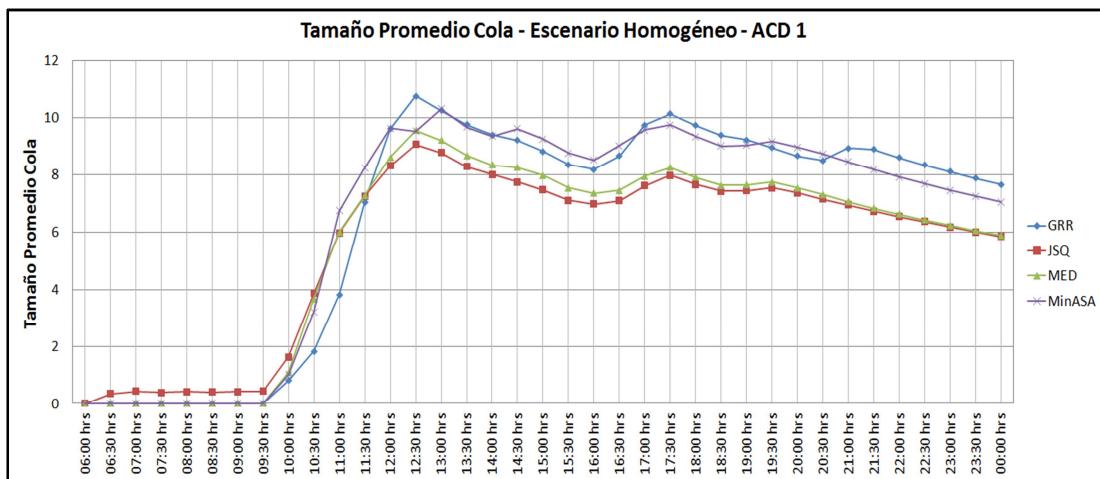


Figura 31. Tamaño promedio de la cola del servidor ACD1 bajo el escenario homogéneo – Modelo de simulación

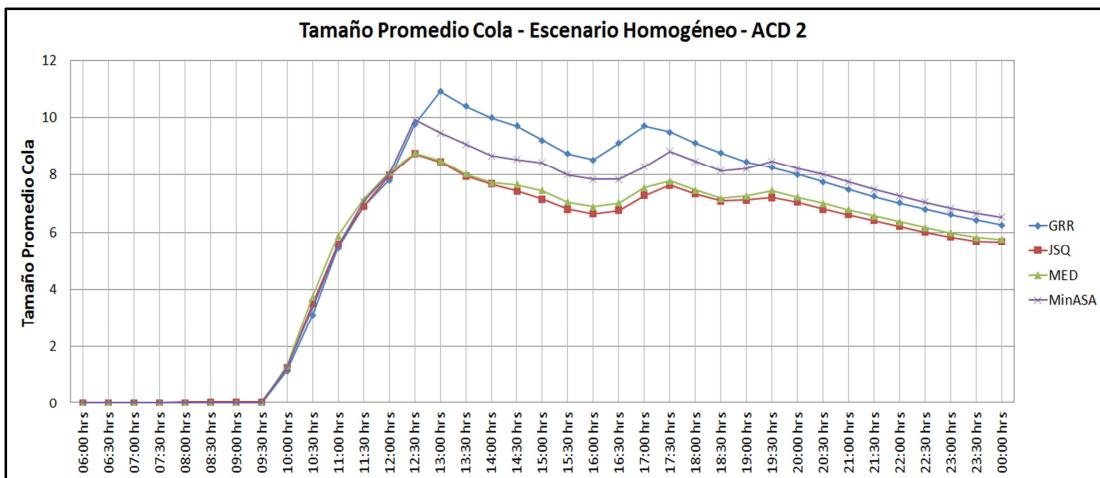


Figura 32. Tamaño promedio de la cola del servidor ACD2 bajo el escenario homogéneo – Modelo de simulación

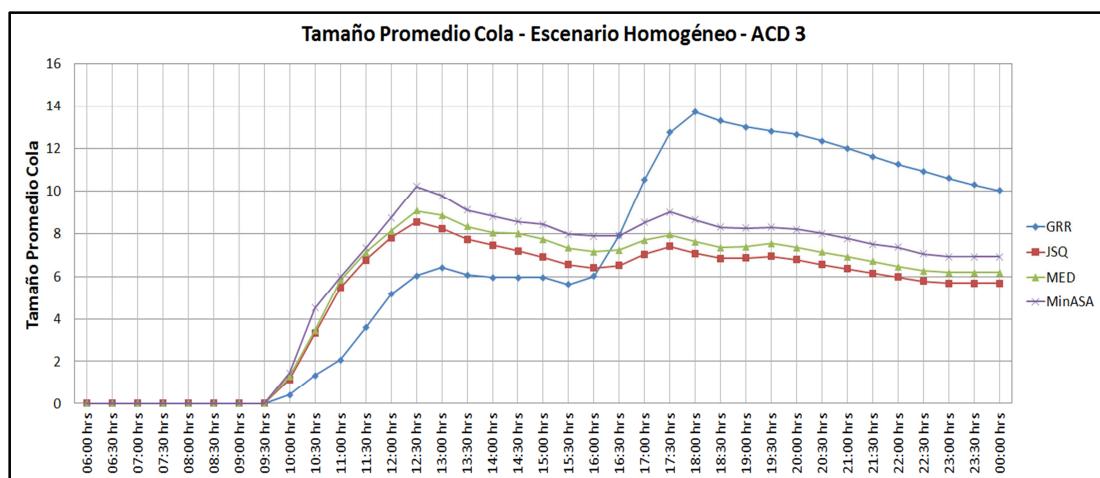


Figura 33. Tamaño promedio de la cola del servidor ACD3 bajo el escenario homogéneo – Modelo de simulación

Adicionalmente, se verificó el comportamiento del tamaño promedio de la cola comparando los resultados de los servidores de ACD en cada uno los algoritmos de balanceo de carga. Se encontró que el algoritmo GRR presenta grandes variaciones entre un servidor y otro, ya que envía la misma cantidad de llamadas a todos los servidores sin tener en cuenta parámetros como el número de agentes conectados en cada uno de los servidores de ACD y el tiempo de espera. Mientras que los algoritmos JSQ y MED mantienen de forma equilibrada el tamaño promedio de la cola en los 3 servidores. En cuanto al algoritmo Min ASA, mantiene el tamaño promedio de la cola equilibrado en los servidores de ACD2 y ACD3, sin embargo el ACD1 tiene un mayor promedio. A continuación se muestran los resultados obtenidos:

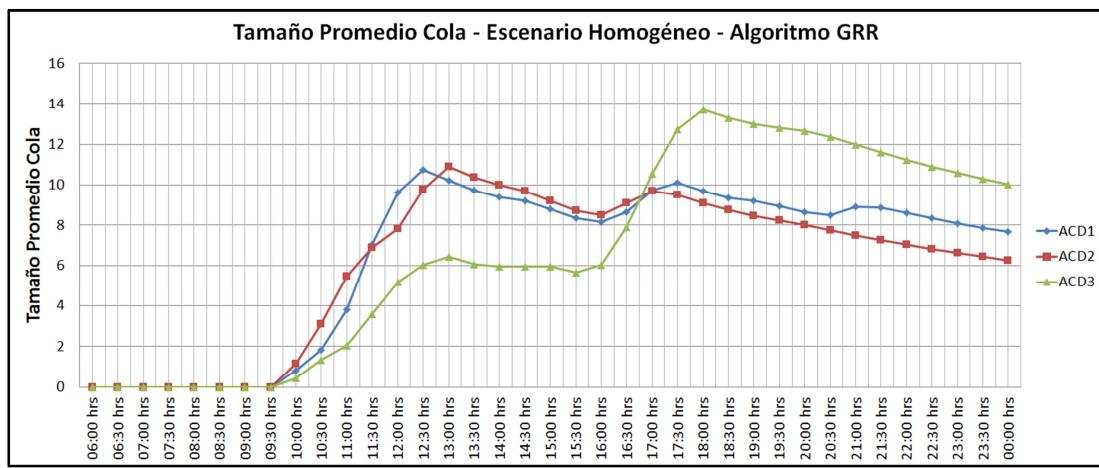


Figura 34. Tamaño promedio de la cola utilizando el algoritmo GRR bajo el escenario homogéneo – Modelo de simulación

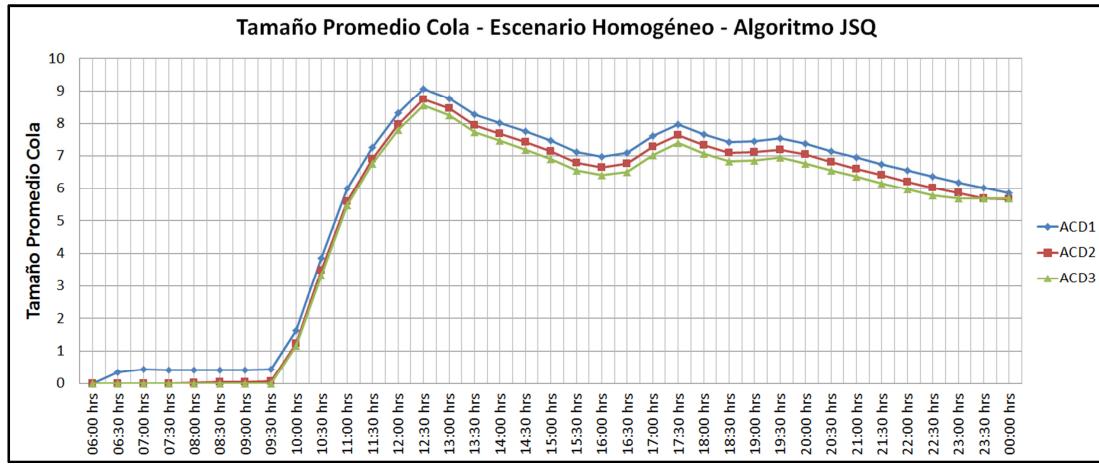


Figura 35. Tamaño promedio de la cola utilizando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación

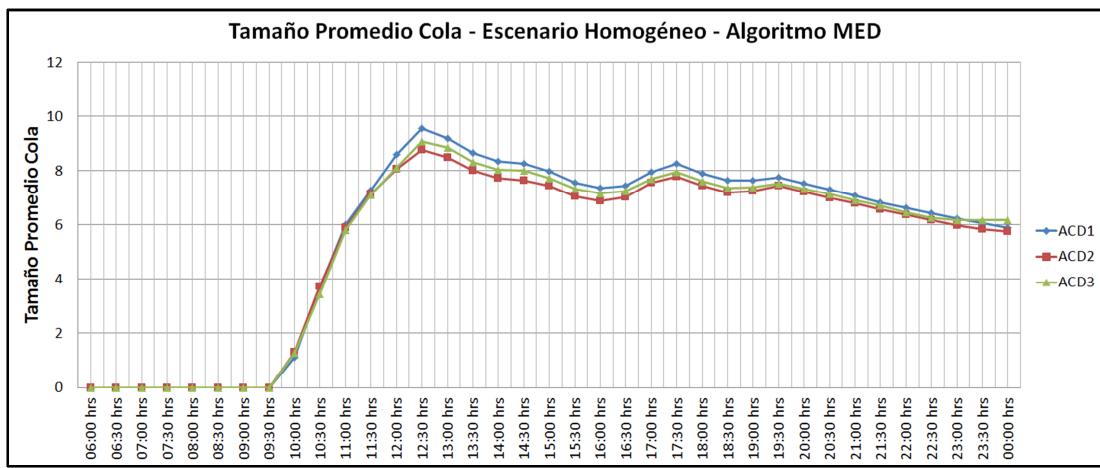


Figura 36. Tamaño promedio de la cola utilizando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación

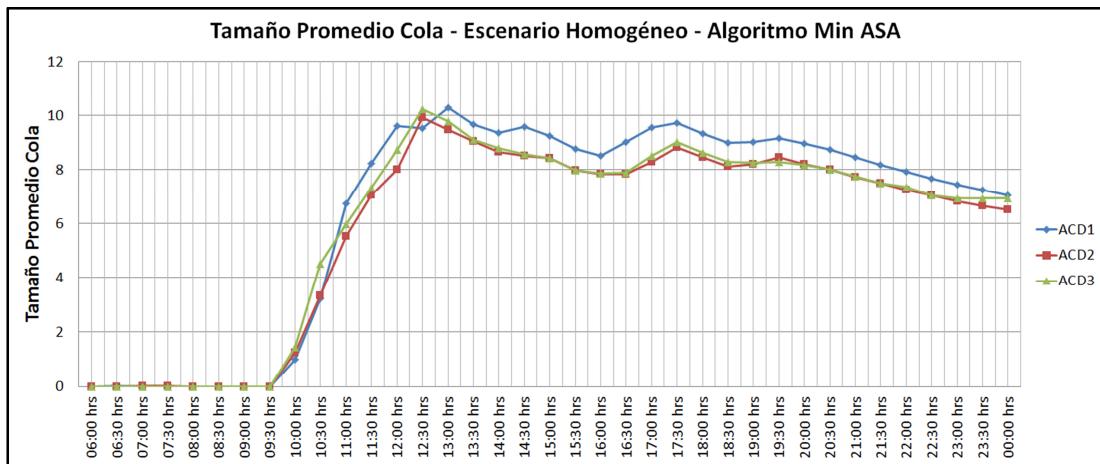


Figura 37. Tamaño promedio de la cola utilizando el algoritmo Min ASA bajo el escenario homogéneo – Modelo de simulación

En la Figura 38 se puede apreciar que el algoritmo MED dentro del escenario heterogéneo, es capaz de adaptarse y distribuir las llamadas de los clientes de acuerdo al número de agentes registrados en cada uno de los servidores de ACD. Es por ello que el tamaño promedio de la cola es mayor en el servidor ACD3 (25 agentes conectados) y menor en el servidor ACD2 (20 agentes conectados).

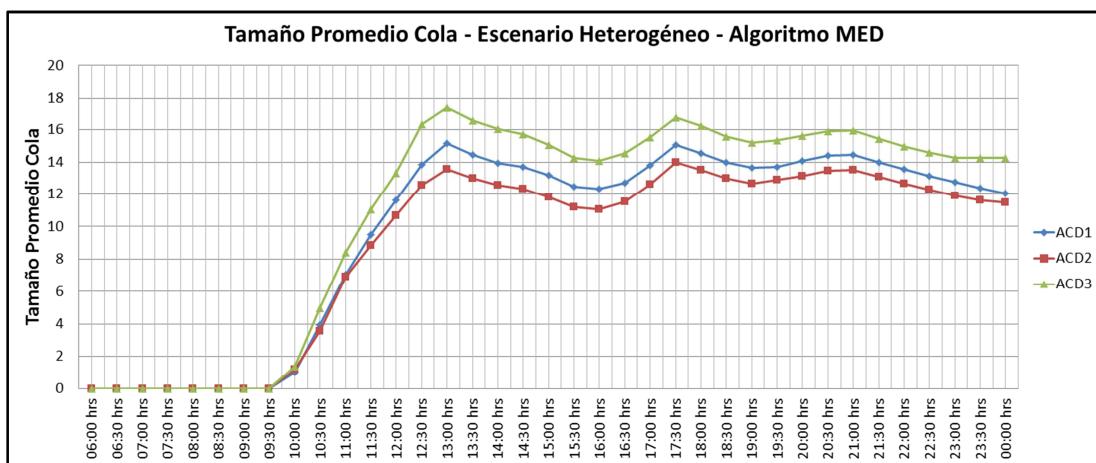


Figura 38. Tamaño promedio de la cola utilizando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación

El algoritmo JSQ presenta un comportamiento similar tanto en el escenario homogéneo como heterogéneo (ver Figuras 35 y 39), debido a que su métrica de decisión es el tamaño de la cola.

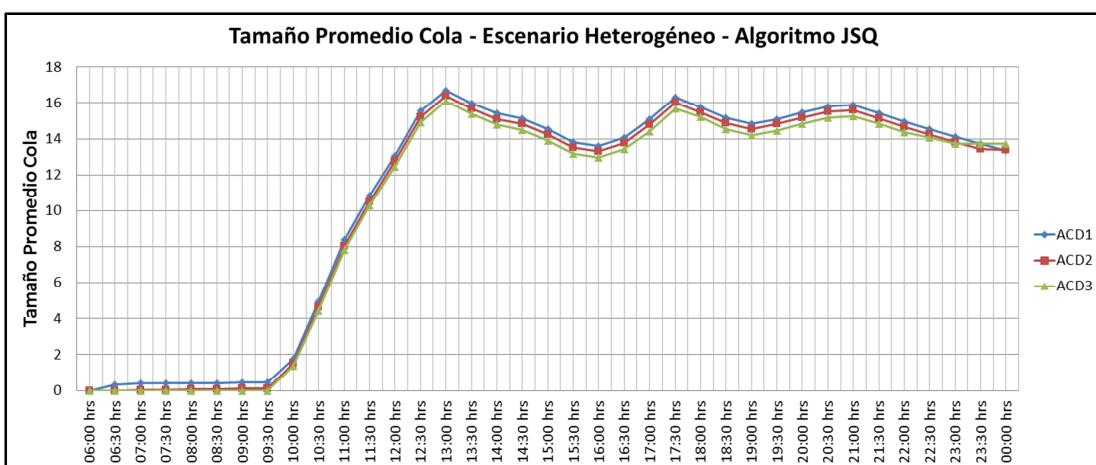


Figura 39. Tamaño promedio de la cola utilizando el algoritmo JSQ bajo el escenario heterogéneo – Modelo de simulación

El algoritmo de turno rotativo GRR, por su parte, presenta el peor resultado, ya que sobrecarga de llamadas al servidor con menor número de agentes conectados (ACD2) y subutiliza el servidor con mayor número de agentes conectados (ACD3). En la siguiente figura se puede observar dicho comportamiento:

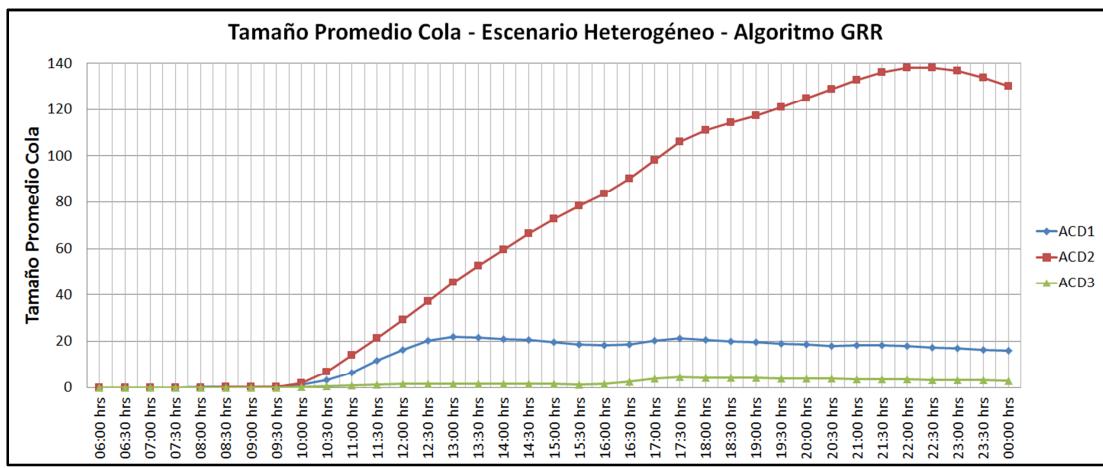


Figura 40. Tamaño promedio de la cola utilizando el algoritmo GRR bajo el escenario heterogéneo – Modelo de simulación

Finalmente, en la Figura 41 se puede apreciar que el algoritmo Min ASA tiene un comportamiento aceptable desde las 13:00 horas hasta las 17:00 horas, sin embargo durante el inicio y el final de la jornada su comportamiento es bastante variable.

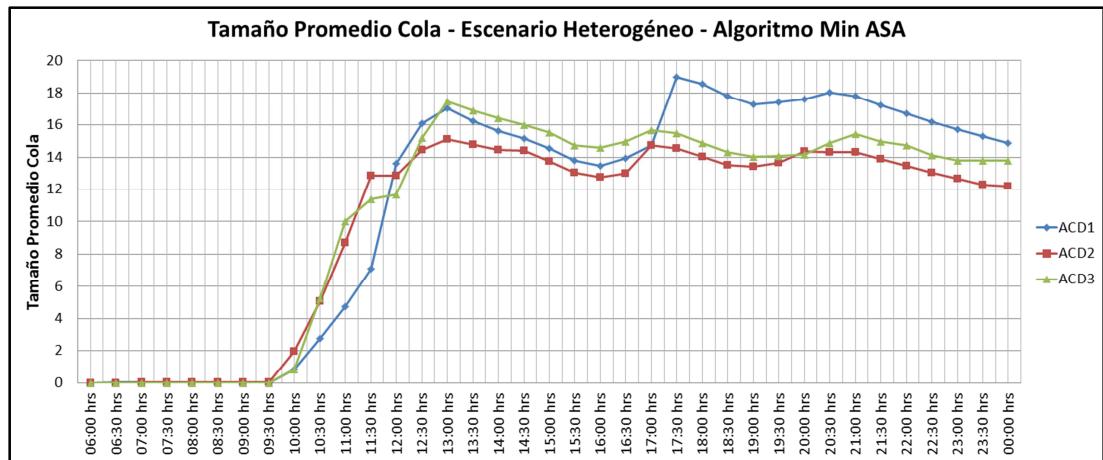


Figura 41. Tamaño promedio de la cola utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Modelo de simulación

7.4.2 Tiempo Promedio de Espera en Cola

Una de las métricas más importantes en un sistema de Call Center es el tiempo que un cliente espera en cola antes de ser atendido por un agente u operador. Bajo el escenario homogéneo, los algoritmos JSQ y MED presentan el mejor comportamiento, pues tienen un tiempo promedio de espera en cola en los 3 servidores de ACD un poco mayor a 60 segundos y una desviación estándar entre 79 y 88 segundos, como se muestra a continuación:

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	84,44	62,31	62,50	75,33
Desviación Estándar	122,20	79,56	87,57	100,86

Tabla 3. Tiempo promedio de espera en cola en el servidor ACD1 durante la jornada laboral – Escenario Homogéneo

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	68,75	60,50	61,56	70,03
Desviación Estándar	107,76	83,97	84,66	93,15

Tabla 4. Tiempo promedio de espera en cola en el servidor ACD2 durante la jornada laboral – Escenario Homogéneo

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	114,58	60,00	65,17	72,36
Desviación Estándar	191,92	81,61	83,93	105,19

Tabla 5. Tiempo promedio de espera en cola en el servidor ACD3 durante la jornada laboral – Escenario Homogéneo

En las Figuras 42 y 43, también se puede apreciar que los algoritmos JSQ y MED tienen un promedio de espera en cola similar en los 3 servidores de ACD a lo largo de la jornada laboral, lo que indica que sin importar a qué servidor sea direccionada la llamada de un cliente, este esperará en cola un tiempo similar. Este comportamiento no es el mismo para los algoritmos GRR y Min ASA.

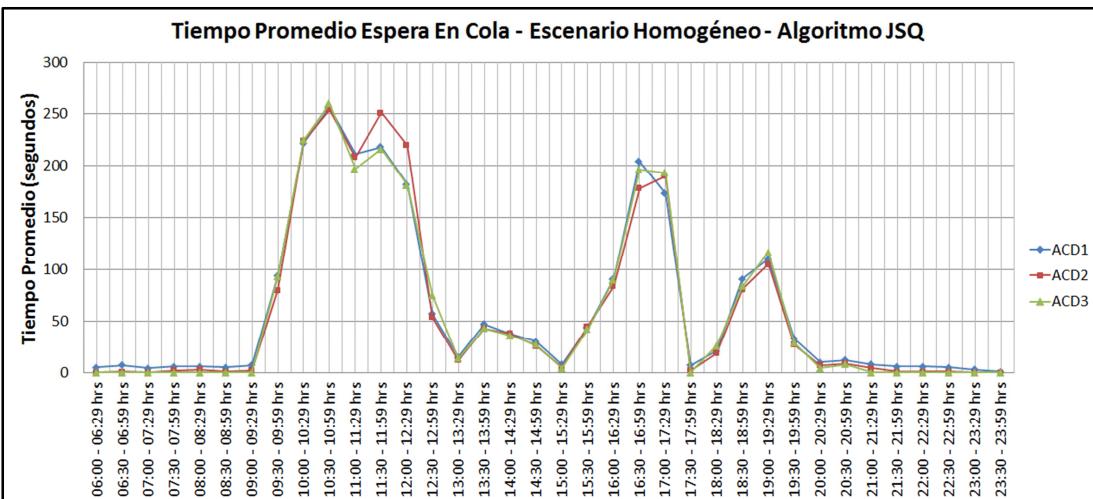


Figura 42. Tiempo promedio de espera en cola utilizando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación

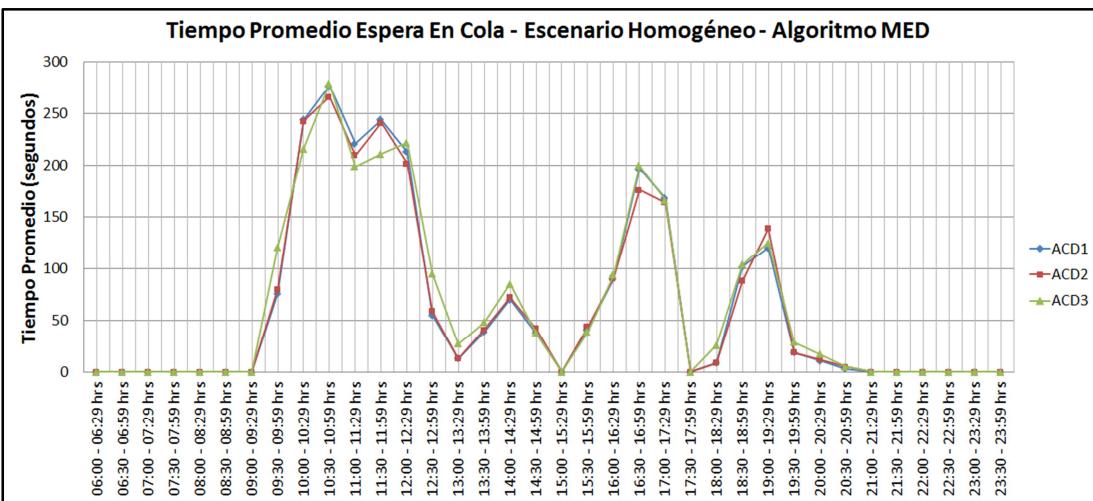


Figura 43. Tiempo promedio de espera en cola utilizando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación

Bajo el escenario heterogéneo, el algoritmo MED tiene el mejor resultado, pues presenta tiempos promedio de espera en cola entre 134 y 140 segundos, con una desviación estándar entre 159 y 166 segundos, como se muestra en las tablas siguientes:

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	179,92	148,69	134,14	166,72
Desviación Estándar	230,04	170,14	162,94	244,32

Tabla 6. Tiempo promedio de espera en cola en el servidor ACD1 durante la jornada laboral – Escenario Heterogéneo

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	1585,47	161,36	139,39	149,06
Desviación Estándar	1126,96	188,89	165,74	180,08

Tabla 7. Tiempo promedio de espera en cola en el servidor ACD2 durante la jornada laboral – Escenario Heterogéneo

Medida	GRR (seg)	JSQ (seg)	MED (seg)	Min ASA (seg)
Promedio	32,19	131,81	136,44	131,61
Desviación Estándar	69,10	152,37	159,16	163,94

Tabla 8. Tiempo promedio de espera en cola en el servidor ACD3 durante la jornada laboral – Escenario Heterogéneo

En la Figura 44 se puede observar el comportamiento del algoritmo MED en los 3 servidores de ACD:

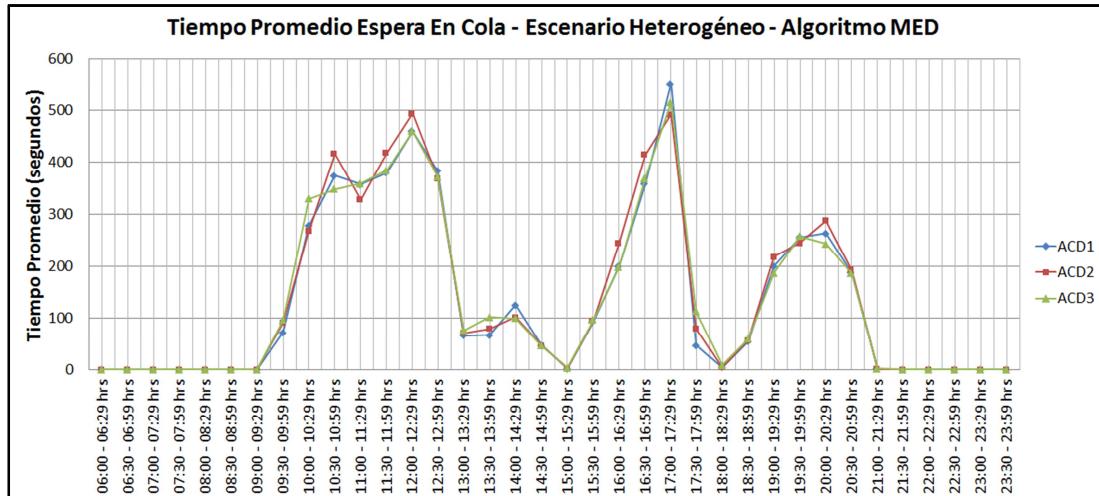


Figura 44. Tiempo promedio de espera en cola utilizando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación

7.4.3 Utilización de los Agentes

La utilización de los agentes hace referencia a la fracción promedio en que los agentes están ocupados atendiendo llamadas de los clientes. Esta métrica es de gran importancia desde el punto de vista operativo, ya que los recursos (agentes) deben ser utilizados el mayor tiempo posible y de forma homogénea en todos los servidores de ACD.

Bajo los dos escenarios de simulación, los algoritmos JSQ, MED y Min ASA muestran una desequilibrada utilización de agentes, ya que sobrecargan el servidor ACD1 y subutilizan el servidor ACD3 durante las primeras y últimas horas de la jornada. Esto significa que los agentes conectados al servidor ACD1 tienen una mayor carga laboral frente a los agentes conectados al servidor ACD3. Por otro lado, el algoritmo de turno rotativo GRR presenta una utilización más equitativa de los agentes. En las figuras que se muestran en los Anexos A-2 y A-3 se puede observar este fenómeno.

Como resultado general de las simulaciones realizadas, se obtuvo un mejor rendimiento utilizando el algoritmo MED, sin embargo el algoritmo JSQ presenta resultados bastante cercanos a los obtenidos con el algoritmo MED bajo los dos escenarios de simulación, convirtiéndolo en otra alternativa para el balanceo de carga de llamadas.

Por otro lado, es claro que bajo un escenario heterogéneo, el algoritmo de turno rotativo GRR obtiene los peores resultados, pero cuando se observa desde el punto de vista de la operación, este algoritmo utiliza de forma equitativa los agentes en los 3 servidores de ACD. Es por ello, que se propuso un algoritmo híbrido entre los algoritmos MED y GRR denominado *Minimum Expected Delay Modified* (MEDM), que además determina el menor retardo esperado entre los servidores de ACD, utiliza la técnica de turno rotativo cuando el menor retardo esperado es el mismo para dos o más ACDs. En la siguiente sección se describe el funcionamiento de este algoritmo:

7.4.4 Algoritmo Propuesto: *Minimum Expected Delay Modified* (MEDM)

El algoritmo MEDM es un híbrido entre los dos algoritmos MED y GRR. Además de calcular el retardo esperado en cada uno de los servidores de ACD y determinar cuál es el menor, utiliza la técnica de turno rotativo cuando el menor retardo esperado es el mismo para dos o más servidores de ACD, teniendo en cuenta a cuál de los ACDs se le envió la última llamada. De esta manera el algoritmo tomará la decisión de balanceo de carga de una forma más equilibrada. En la Figura 45 se presenta el diagrama de flujo de este algoritmo:

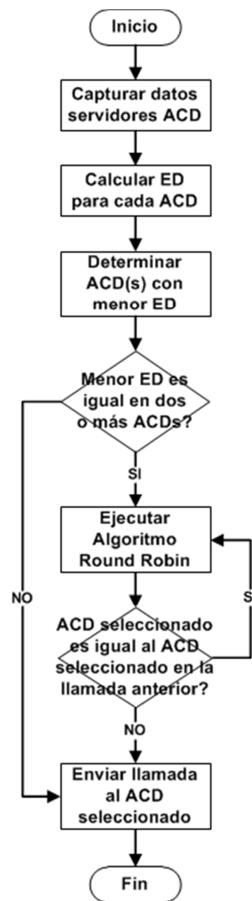


Figura 45. Diagrama de flujo del algoritmo MEDM

Al realizar las simulaciones con algoritmo propuesto, se logró constatar que es capaz de mantener equilibrada la utilización de los agentes durante toda la jornada laboral (ver Figuras 46 y 47), sin afectar drásticamente el tamaño promedio de la cola y los tiempos promedio de espera que se obtienen con el algoritmo MED (ver Tablas 9 y 10). Por lo tanto, se puede afirmar que el algoritmo MEDM se convierte en una buena alternativa para el balanceo de carga de llamadas en sistemas de Call Center implementados en una red LAN.

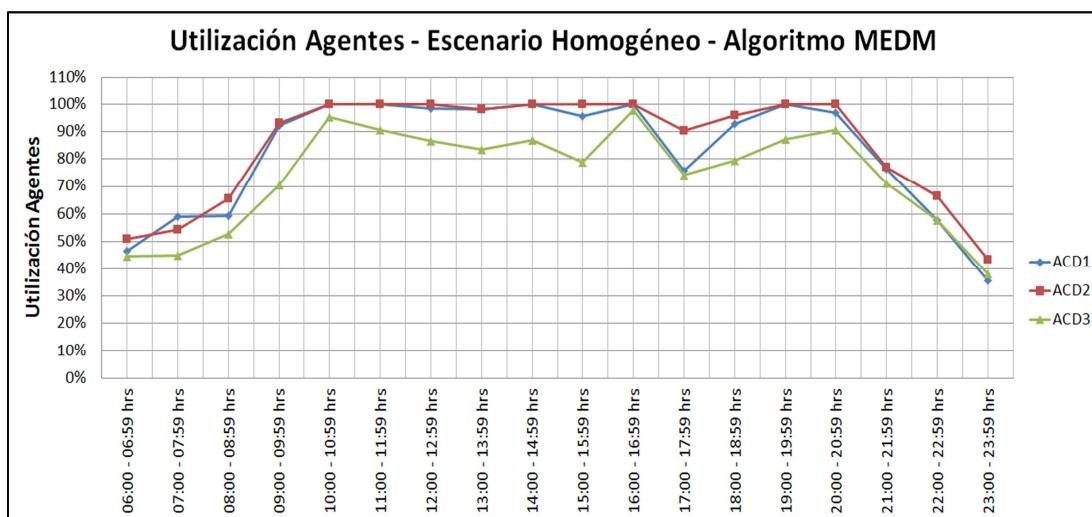


Figura 46. Utilización de los agentes empleando el algoritmo MEDM bajo el escenario homogéneo – Modelo de simulación

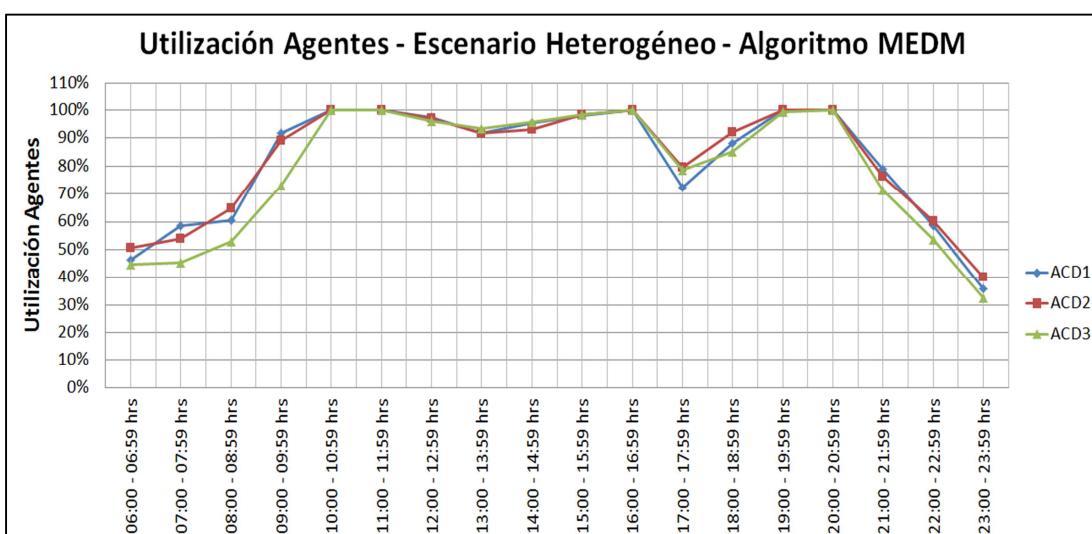


Figura 47. Utilización de los agentes empleando el algoritmo MEDM bajo el escenario heterogéneo – Modelo de simulación

Servidor	GRR (seg)	MED (seg)	MED Modified (seg)
ACD1	84,44	62,50	72,44
ACD2	68,75	61,56	69,03
ACD3	114,58	65,17	71,00

Tabla 9. Tiempo promedio de espera en cola de los algoritmos GRR, MED y MEDM durante la jornada laboral – Escenario Homogéneo

Servidor	GRR (seg)	MED (seg)	MED Modified(seg)
ACD1	179,92	134,14	145,06
ACD2	1585,47	139,39	147,97
ACD3	32,19	136,44	143,50

Tabla 10. Tiempo promedio de espera en cola de los algoritmos GRR, MED y MEDM durante la jornada laboral – Escenario Heterogéneo

8. ANÁLISIS Y ESPECIFICACIÓN DE LOS REQUERIMIENTOS DEL BALANCEADOR DE CARGA

Los requerimientos para la implementación del módulo para el balanceo de carga de llamadas fueron tomados teniendo en cuenta las necesidades de los Call Center en la actualidad. A continuación se describe de manera detallada cada uno de los casos de uso del balanceador de carga de llamadas utilizando el lenguaje gráfico UML (*Unified Modeling Language*¹³).

8.1 CONFIGURAR BALANCEADOR

Este caso de uso permite al administrador de la plataforma realizar la configuración del balanceador de llamadas de carga de llamadas. El administrador podrá agregar y/o quitar servidores de ACD y/o servidores de acceso (*gateways* de acceso), adicionar y/o eliminar colas, y seleccionar el algoritmo para la distribución de llamadas en múltiples servidores de ACD. En la Figura 48 se muestra el diagrama UML de este caso de uso:

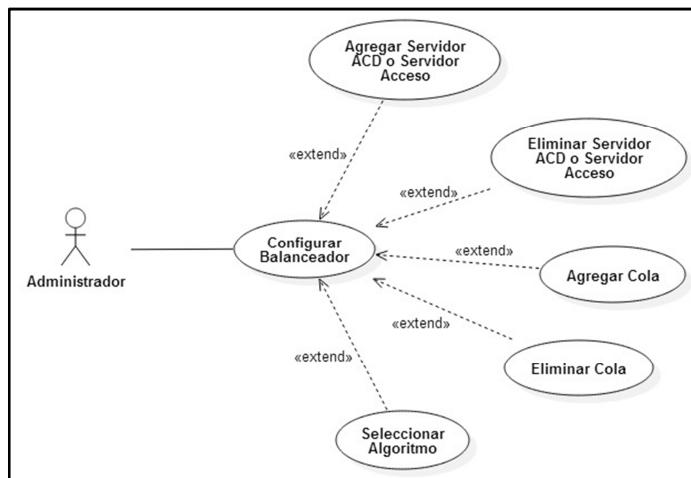


Figura 48. Caso de Uso Configurar Balanceador

En la Tabla 11 se describe este caso de uso:

Sistema:	Balanceador de carga de llamadas.
Nombre:	Configurar Balanceador.
Descripción:	Permite al usuario configurar el balanceador de carga de llamadas teniendo en cuenta los servidores de ACD, servidores de acceso y nombres de las colas. Además se puede definir el algoritmo para distribuir las llamadas.
Actores:	Administrador.
Precondiciones:	<ol style="list-style-type: none">1. El usuario debe estar autenticado en el sistema.2. Los servidores que hagan parte de la plataforma deben tener configurada una dirección IP dentro de la misma red.3. Las colas del Call Center deben estar definidas en cada uno de los

¹³ *Unified Modeling Language* (UML): Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.

	<ul style="list-style-type: none"> servidores de ACD (archivo <code>/etc/asterisk/queues.conf</code>). 4. Los agentes del Call Center deben estar configurados en cada uno de los servidores de ACD (archivo <code>/etc/asterisk/agents.conf</code>). 5. El usuario para realizar la conexión al <i>manager</i> de Asterisk (AMI) de cada uno de los servidores debe estar creado en el archivo <code>/etc/asterisk/manager.conf</code>.
Flujo Normal:	<ul style="list-style-type: none"> 1. El usuario ejecuta el comando para desplegar la interfaz gráfica de configuración. 2. El sistema despliega la interfaz gráfica de configuración. 3. El usuario ingresa el número de servidores de acceso y de agentes con sus respectivas direcciones IP. 4. El usuario ingresa el número de colas y sus respectivos nombres. 5. El usuario selecciona el algoritmo de distribución de llamadas a ejecutar. 6. El sistema crea un archivo de configuración con extensión <code>.conf</code>.
Flujo Alternativo:	<ul style="list-style-type: none"> 1. El usuario ejecuta el comando para desplegar la interfaz gráfica de configuración. 2. El sistema despliega la interfaz gráfica de configuración. 3. El usuario ingresa incorrectamente los datos al sistema. 4. El sistema no realiza la creación del archivo de configuración.
Postcondiciones:	<ul style="list-style-type: none"> 1. El administrador de la plataforma configura los servidores de acceso, los servidores de agentes, los nombres colas y selecciona un algoritmo de distribución de llamadas.

Tabla 11. Descripción Caso de Uso Configurar Balanceador

8.2 VISUALIZAR BALANCEADOR

Este caso de uso tiene dos tipos de actores: el supervisor del Call Center y el administrador de la plataforma. Es muy importante para el supervisor del Call Center o para la persona encargada de la operación poder observar en tiempo real el estado y las estadísticas de los servidores de ACD de la operación, pues le permite realizar una adecuada distribución del personal en las horas de alto o bajo tráfico de llamadas. Por otro lado, el administrador de la plataforma, además de poder visualizar el estado y las estadísticas de los servidores de ACD puede verificar el correcto funcionamiento de los servidores de acceso. En la Figura 49 se muestra el diagrama UML de este caso de uso:

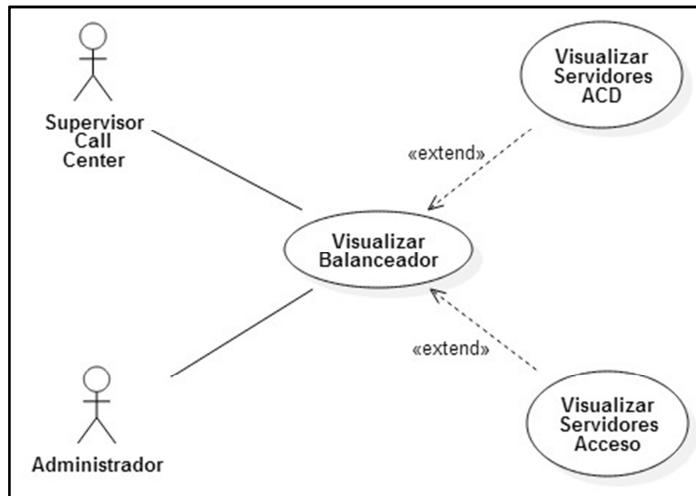


Figura 49. Caso de Uso Visualizar Balanceador

En la Tabla 12 se describe este caso de uso:

Sistema:	Balanceador de carga de llamadas.
Nombre:	Visualizar Balanceador.
Descripción:	Permite al usuario visualizar el estado y las estadísticas tanto de los servidores de ACD como de los servidores de acceso.
Actores:	Administrador y Supervisor.
Precondiciones:	<ol style="list-style-type: none"> El usuario debe estar autenticado en el sistema. El sistema debe estar conectado a cada uno de los servidores de acceso y de agentes. El sistema verifica periódicamente el estado de los servidores y solicita los siguientes datos: <ol style="list-style-type: none"> 1. Servidores de ACD: <ul style="list-style-type: none"> • Agentes conectados. • Agentes disponibles. • Llamadas en cola. • Tiempo promedio de espera (ASA). • Tiempo promedio de servicio (AHT). • Tiempo de espera de la primera llamada en cola. 2. Servidores de Acceso: <ul style="list-style-type: none"> • Llamadas activas.
Flujo Normal:	<ol style="list-style-type: none"> El usuario selecciona la opción de visualizar el estado y las estadísticas de los servidores utilizando el comando “screen -r”. El sistema despliega el panel de visualización.
Flujo Alternativo:	N.A.
Postcondiciones:	<ol style="list-style-type: none"> El usuario observó en tiempo real el estado y las estadísticas de los servidores.

Tabla 12. Descripción Caso de Uso Visualizar Balanceador

8.3 DISTRIBUIR LLAMADAS

El siguiente caso de uso se denomina distribuir llamadas. Las llamadas ingresan desde la Red de Telefonía Pública Comutada (PSTN) o desde la plataforma de VoIP (SIP), a los servidores o *gateways* de acceso. Luego estas llamadas son distribuidas, utilizando un algoritmo de balanceo de carga, entre múltiples servidores de ACD donde se registran los agentes. A continuación se muestra este caso de uso:

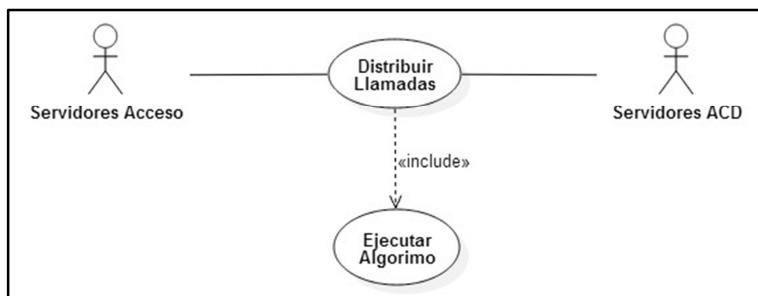


Figura 50. Caso de Uso Distribuir Llamadas

En la Tabla 13 se describe este caso de uso:

Sistema:	Balanceador de carga de llamadas.
Nombre:	Distribuir Llamadas.
Descripción:	Permite al sistema distribuir (utilizando un algoritmo) las llamadas que ingresan al sistema entre múltiples servidores de ACD.
Actores:	Servidores de Acceso y Servidores de ACD.
Precondiciones:	<ol style="list-style-type: none"> El sistema debe estar conectado a cada uno de los servidores de acceso y servidores de ACD a través de la interfaz AMI. El plan de marcación (<i>dialplan</i>) debe estar configurado tanto en los servidores de acceso como en los servidores de ACD (archivo <i>/etc/asterisk/extensions.conf</i>).
Flujo Normal:	<ol style="list-style-type: none"> El sistema recibe los eventos generados por la llegada de las llamadas a los servidores de acceso. El sistema aplica el algoritmo de distribución y decide a qué servidor de ACD direcciona la llamada. La llamada es direccionada a uno de los servidores donde se encuentran conectados los agentes y pasada al ACD local de dicha máquina.
Flujo Alternativo:	<ol style="list-style-type: none"> Si la llamada no se pudo dirigir por algún motivo, se envía a una extensión en el mismo servidor de acceso donde se ejecuta el plan de contingencia.
Postcondiciones:	<ol style="list-style-type: none"> Las llamadas de entrada son distribuidas entre múltiples servidores de ACD donde se conectan los agentes del Call Center.

Tabla 13. Descripción Caso de Uso Distribuir Llamadas

8.4 REGISTRAR DATOS

Este caso de uso se compone de otros tres casos de uso: registrar estadísticas de los servidores de acceso, registrar estadísticas de los servidores de ACD y registrar el ACD seleccionado. El balanceador de carga de llamadas registra en una base de datos las estadísticas de los servidores de acceso y de los servidores de ACD. También registra el servidor de ACD seleccionado.

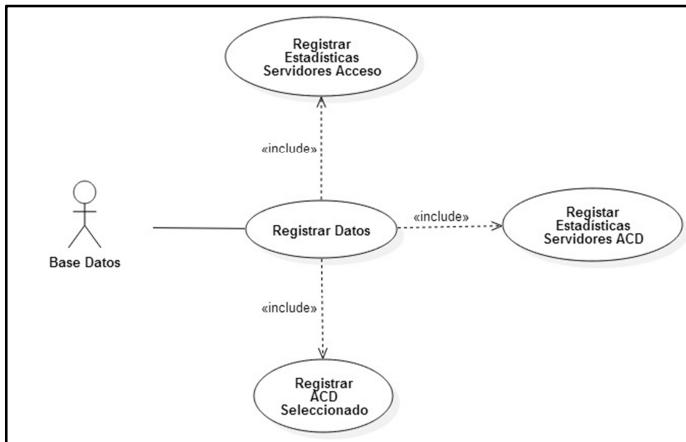


Figura 51. Caso de Uso Registrar Datos

En la Tabla 14 se describe este caso de uso:

Sistema:	Balanceador de carga de llamadas.
Nombre:	Registrar Datos.
Descripción:	Permite al sistema registrar en una base de datos las estadísticas de los servidores de acceso y de los servidores de ACD, al igual que el servidor de ACD seleccionado mediante alguno de los algoritmos de distribución.
Actores:	Base de Datos.
Precondiciones:	<ol style="list-style-type: none"> El sistema debe estar conectado a cada uno de los servidores de acceso y servidores de ACD. El sistema debe estar conectado a la base de datos.
Flujo Normal:	<ol style="list-style-type: none"> El sistema recibe los eventos generados por la llegada de las llamadas entrantes a los servidores de acceso. El sistema aplica el algoritmo de distribución y decide a qué servidor de ACD direcciona la llamada. El sistema registra los datos de los servidores de accesos y de los servidores de ACD en la base de datos. De igual manera se registra el servidor de ACD seleccionado. La llamada es direccionada a uno de los servidores donde se encuentran conectados los agentes y pasada al ACD local de dicha máquina.
Flujo Alternativo:	<ol style="list-style-type: none"> Si la conexión a la base de datos falla, el sistema escribe los datos en un archivo de log.
Postcondiciones:	<ol style="list-style-type: none"> Base de datos actualizada.

Tabla 14. Descripción Caso de Uso Registrar Datos

9. DISEÑO E IMPLEMENTACIÓN DEL BALANCEADOR DE CARGA

Como se comentó en el capítulo 6, el balanceador de carga establece una única conexión a la interfaz AMI de cada uno de los servidores Asterisk tanto de Acceso como de ACD. La aplicación crea tantas conexiones (hilos) como servidores Asterisk existan. En el núcleo, cada hilo lee constantemente el *socket* de cada conexión e interpreta los eventos generados por el AMI, capturando la información necesaria para distribuir las llamadas.

Cuando una llamada ingresa a la plataforma, el balanceador recibe un evento generado por la llegada de la llamada a uno de los servidores de acceso. Luego ejecuta el algoritmo de distribución y decide a que servidor de ACD se debe direccionar la llamada. El balanceador le informa al servidor de acceso a que servidor de ACD debe enviar la llamada. Finalmente, la llamada es direccionada al servidor seleccionado y enviada al ACD local de dicha máquina, especificándole la cola elegida por el cliente en las opciones del IVR.

9.1 ARQUITECTURA DEL BALANCEADOR DE CARGA

Al igual que Asterisk, el balanceador de carga de llamadas está conformado por una serie de módulos que se relacionan entre sí. En la figura siguiente se muestra la arquitectura del balanceador de carga:

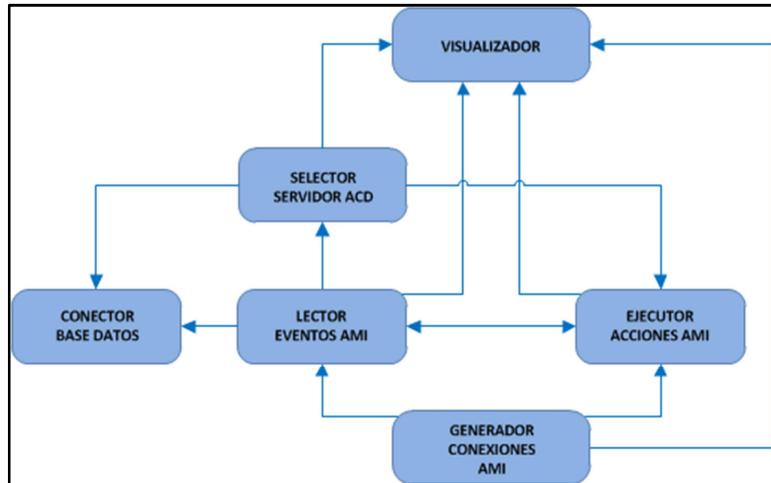


Figura 52. Arquitectura del balanceador de carga

9.1.1 Generador de Conexiones AMI

Este módulo es el encargado generar y establecer las conexiones a la interfaz AMI de cada uno de los servidores de Acceso y de ACD. Gracias a estas conexiones es posible leer eventos de los servidores y ejecutar acciones (comandos) a cada servidor de la plataforma. Cada conexión se hace a través de un hilo diferente, convirtiendo al balanceador de carga en una aplicación multihilo.

9.1.2 Lector de Eventos AMI

Una vez se hayan establecido las conexiones con cada uno de los servidores de la plataforma, el lector de eventos captura los datos generados por la aplicación Asterisk. De esta manera el balanceador de carga puede determinar si una llamada ingresó a la plataforma y obtener la respuesta de los comandos ejecutados por la aplicación.

9.1.3 Ejecutor de Acciones AMI

Este módulo es el encargado ejecutar acciones o comandos que permitan conocer el estado y las estadísticas de los servidores de acceso y de agentes. Además, permite direccionar las llamadas hacia los servidores de ACD.

9.1.4 Selector de Servidor de ACD

En este módulo se implementan los diferentes algoritmos de distribución de llamadas.

9.1.5 Visualizador

Este módulo permite al usuario (administrador y supervisor del Call Center) ver el estado y estadísticas de los servidores Asterisk que hacen parte de la plataforma.

9.1.6 Conector Base de Datos

Permite registrar en una Base de Datos las estadísticas de los servidores de acceso y de los servidores de ACD, al igual que el servidor seleccionado mediante alguno de los algoritmos de balanceo de carga.

9.2 LENGUAJE DE PROGRAMACIÓN

Para que la aplicación del balanceador de carga fuera lo más liviana y rápida posible, se utilizó el lenguaje de programación C++ y la programación orientada a objetos (POO). Este lenguaje de programación consume pocos recursos del sistema, maneja una gran variedad de bibliotecas y es utilizado por la solución de software libre Asterisk.

Se utilizó la clase *CThread*, desarrollada por Walter E. Capers [25] para el manejo de los hilos. Dicha clase se caracteriza por:

- Soporta dos modelos de hilos: hilos orientados a eventos e hilos asíncronos.
- Soporta hilos homogéneos e hilos especializados.
- Provee un manejo FCFS (*First Come First Serve*) para el manejo de tareas.

En el caso de los hilos orientados a eventos, el hilo se encuentra todo el tiempo en un estado de pausa, esperando a que la función miembro *Event* sea llamada. Cuando esto sucede el hilo despierta y ejecuta las instrucciones de la función virtual *OnTask*. Bajo el modelo de hilos asíncronos cada hilo despierta cada t milisegundos y ejecuta las instrucciones de la función *OnTask*. Para la implementación del balanceador de carga, se escogió el modelo de hilos asíncronos, ya que cada hilo está constantemente procesando los eventos leídos desde el *socket* de cada máquina.

Para la creación y conexión del socket se utilizó la clase *ClientSocket* desarrollada por Rob Tougher [26]. Esta clase se fue empleada para la comunicación con la interfaz AMI de cada servidor Asterisk, pues permite de manera sencilla la lectura y escritura de datos en el *socket* correspondiente.

Además, la aplicación fue implementada para que funcione sobre sistemas operativos Linux como CentOS y Ubuntu.

9.3 IMPLEMENTACIÓN DEL BALANCEADOR DE CARGA

9.3.1 Clases Desarrolladas

Para la implementación del balanceador de carga se definieron las clases *AsteriskManagerIVR* y *AsteriskManagerAgent*, las cuales heredan de la clase *CThread*. Como variables miembro cuentan con un objeto de la clase *ClientSocket* para la lectura y escritura en el *socket* de la interfaz AMI, y los datos necesarios para la conexión a dicha interfaz: dirección IP del servidor, usuario y clave del *manager* de Asterisk, puerto TCP/IP, entre otras.

También se definió la clase *LeerArchivoConfiguracion* que se utiliza para leer el archivo de configuración del balanceador de carga.

En el Anexo A-4 se muestra la definición de las clases mencionadas anteriormente.

9.3.2 Funcionamiento del Balanceador de Carga

En esta sección se comentará el funcionamiento del módulo desarrollado haciendo referencia a partes del código.

Una vez el programa es iniciado, la función principal *main()*, crea un objeto *LeerArchivoConfiguracion* que permite leer el archivo de configuración para que el balanceador de carga inicie su operación. El archivo de configuración contiene los datos de los servidores de acceso (servidores de IVR), servidores de ACD, nombre de las colas, el algoritmo a utilizar y el servidor de base de datos donde se registran los eventos, tal y como se muestra a continuación:

```

numasteriskivr:2
ipasteriskivr1:192.168.72.242
ipasteriskivr2:192.168.72.243

numasteriskag:3
ipasteriskag1:192.168.72.244
ipasteriskag2:192.168.72.245
ipasteriskag3:192.168.72.246

numcolas:3
cola1:cola1
cola2:cola2
cola3:cola3

algoritmo:jsq

bdenable:on
bdnombre:bd_tesis
bdip:192.168.72.250
bdusuario:postgres
bdclave:postgres
bdpuerto:5432
bdesquema:jsq_heterogeneo

```

Con el archivo de configuración cargado en memoria, el programa continúa con la creación de dos tipos de objetos, uno de la clase *AsteriskManagerAgent* y otro de la clase *AsteriskManagerIVR*. Estas clases son las encargadas de generar las conexiones al *manager* de Asterisk (AMI) de cada uno de los servidores de la plataforma, definiendo un hilo de tipo asíncrono con un tiempo muerto de 100 milisegundos por cada servidor. El código asociado es el siguiente:

```

//Se realiza la conexión a los Asterisk de ACD
for(n=0;n<serversAgent;n++)
{
    thrAsteriskAgent[n] = new AsteriskManagerAgent(ipServerAgent[n],Port,UserAMI,PassAMI);
    thrAsteriskAgent[n]->IniciarMapaGlobal();
    thrAsteriskAgent[n]->SetThreadType(ThreadTypeIntervalDriven,100);
    usleep(1000);
}

//Se realiza la conexión a los Asterisk de Acceso
for(n=0;n<serversIVR;n++)
{
    thrAsteriskIVR[n] = new AsteriskManagerIVR(ipServerIVR[n],Port,UserAMI,PassAMI,n);
    thrAsteriskIVR[n]->Iniciarizar();
    thrAsteriskIVR[n]->SetThreadType(ThreadTypeIntervalDriven,100);
    usleep(1000);
}

```

Una vez la conexión al AMI de cada uno de los servidores esté establecida, se procede a leer los datos enviados por Asterisk para ser evaluados. Cada paquete del *manager* inicia por las palabras reservadas *Action*, *Response* o *Event* y cada línea del paquete termina por un salto de línea representado por la cadena `\r\n`. Finalmente, cada paquete se diferencia de otro por un doble salto de línea representado por la cadena `\r\n\r\n`. El paquete inicial de autenticación es:

```
Action: Login\r\n
Username: balanceador\r\n
Password: pwdbalanceador\r\n
Events: on\r\n\r\n
```

Cada uno de los objetos *AsteriskManagerAgent* está encargado de mantener actualizados los datos de los servidores de ACD, ejecutando la acción *QueueSummary* para cada una de las colas configuradas inicialmente (característica multicola del balanceador), la cual entrega los siguientes datos por cada cola:

- Agentes conectados (*LoggedIn*).
- Agentes disponibles (*Available*).
- Llamadas en cola (*Callers*).
- Tiempo promedio de espera (*HoldTime*).
- Tiempo promedio de servicio (*TalkTime*).
- Tiempo de espera de la primera llamada en cola (*LongestHoldTime*).

Los objetos de la clase *ClientSocket* sobrecargan los operadores `>>` y `<<` para la escritura y lectura de datos en el *socket*. El envío del paquete de autenticación se muestra a continuación:

```
//Se realiza la conexión al manager de Asterisk
AsteriskManagerWR << "Action: login\r\nUsername: " + UserManager + "\r\nSecret: " + PassManager + "\r\nEvents:
on\r\n\r\n"

//Se capturan los datos enviados por el manager de Asterisk
AsteriskManagerWR >> Respuesta;
```

En el proceso de lectura de datos del *socket* es muy posible leer uno o más paquetes del AMI o en algunos casos pueden estar incompletos (no se leen la misma cantidad de bytes en cada ciclo), por lo tanto es necesario separar y evaluar cada paquete por separado. Esta tarea se realiza con las funciones *SeparaPaqueteEvento()* y *SeparaPaqueteResponse()*. Dependiendo del tipo de paquete se realizan las acciones pertinentes.

9.3.2.1 Recepción y Direcccionamiento de las Llamadas

Cuando una llamada es recibida por un servidor de acceso, se genera un evento de usuario *UserEvent* (definido en el archivo `/etc/asterisk/extensions.conf`) cuya cabecera es la palabra reservada *Event*:

```

Event: UserEvent\r\n
Privilege: user,all\r\n
UserEvent: InboundCall\r\n
Channel: SIP/t_generador_to_gw2-0000268c\r\n
Exten: 1410\r\n
Context: bugarouter\r\n
Queue: cola1\r\n
UniqueID: 2-1431100479.9868\r\n\r\n

```

En ese momento el objeto *AsteriskManagerIVR* asociado a ese servidor, lee el evento y extrae del paquete el evento (*UserEvent*), el canal por donde ingreso la llamada (*Channel*) y la cola (*Queue*), haciendo uso de la función *SepararPaqueteEvento()*. Estos datos son enviados a la función *DeterminarServidor()*, que como su nombre lo indica, es la encargada de determinar a cual servidor de ACD se debe direccionar la llamada cuando se haya aplicado el algoritmo de distribución configurado previamente. Esta función también utiliza los datos que constantemente se están solicitando a los servidores de ACD por medio de los objetos *AsteriskManagerAgent*.

Una vez se haya determinado el servidor de ACD al cual hay que direccionar la llamada, el balanceador de carga envía una acción de redirección (*Action: Redirect*) con el siguiente fragmento de código:

```

//La variable sAsteriskAgentes contiene el servidor de ACD seleccionado
AsteriskManagerWR << "Action: Redirect\r\n";
AsteriskManagerWR << "ActionID: " + ActionID + "\r\n";
AsteriskManagerWR << "Channel: SIP/t_generador_to_gw2-0000268c \r\n";
AsteriskManagerWR << "Exten: " + sAsteriskAgentes + "\r\n";
AsteriskManagerWR << "Context: bugarouter\r\n";
AsteriskManagerWR << "Priority: 1\r\n\r\n";

```

De esta manera la llamada es direccionada al servidor de ACD seleccionado por el algoritmo.

9.3.3 Diagramas de Flujo de los Algoritmos de Balanceo de Carga

Como se comentó en la sección anterior, los algoritmos de balanceo de carga son ejecutados cada vez que ingresa una llamada al sistema, como se muestra en el diagrama de flujo siguiente:

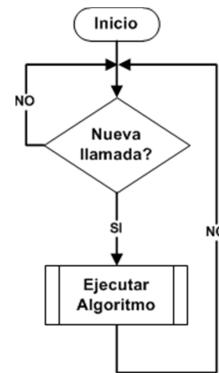


Figura 53. Diagrama de flujo para ejecución de los algoritmos de balanceo de carga

Cada uno de los algoritmos de balanceo de carga fue implementado en el lenguaje de programación C++, teniendo en cuenta los diagramas de flujo que se presentan a continuación:

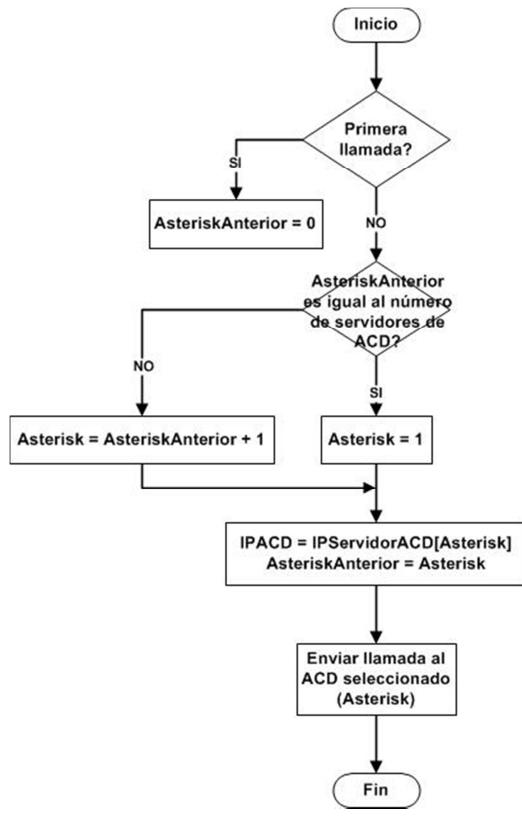


Figura 54. Diagrama de flujo del algoritmo GRR

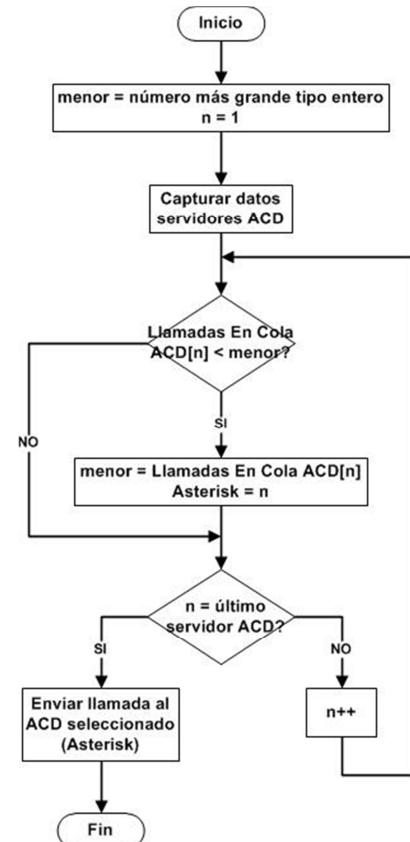


Figura 55. Diagrama de flujo del algoritmo JSQ

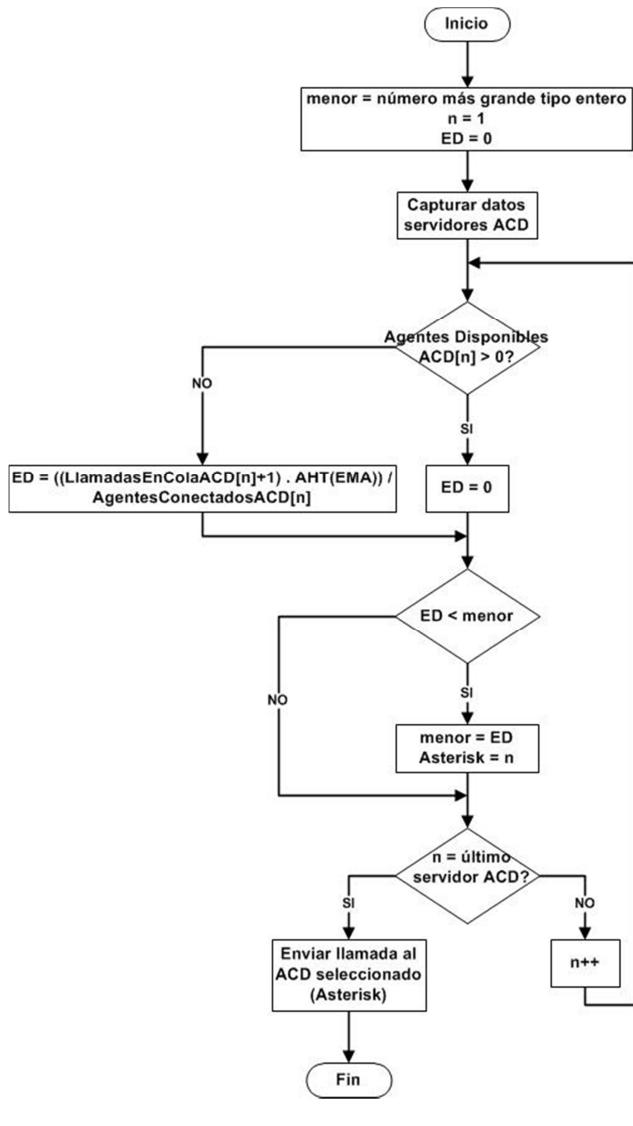


Figura 56. Diagrama de flujo del algoritmo MED

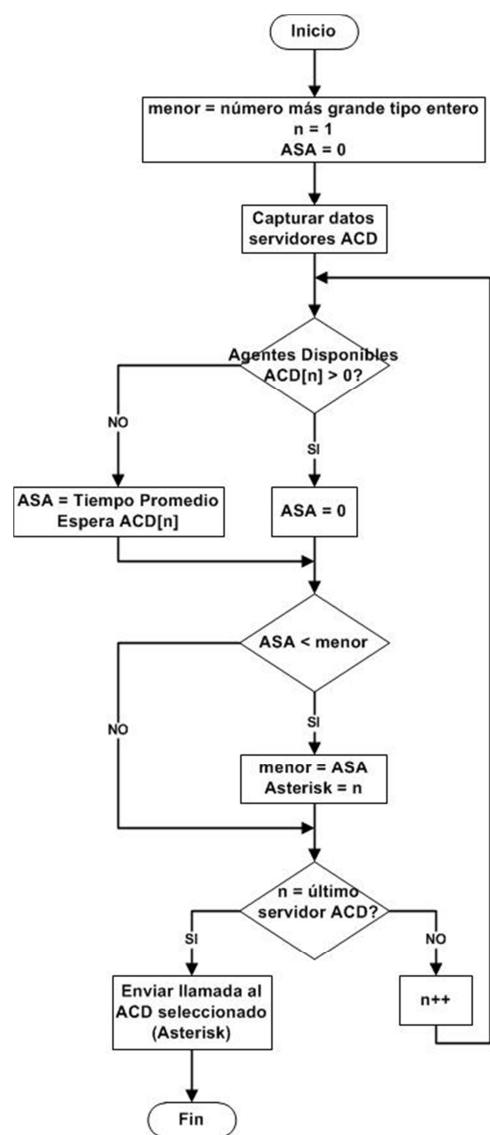


Figura 57. Diagrama de flujo del algoritmo Min ASA

10. IMPLEMENTACIÓN DEL PROTOTIPO DEL SISTEMA

La Figura 58 muestra la arquitectura del prototipo del sistema. Este tipo de arquitectura es similar a la de un Call Center Virtual (VCC: *Virtual Call Center*) configurado con balanceo de carga [2], sin embargo la arquitectura propuesta no maneja un *switch* telefónico centralizado, sino un conjunto de *gateways* o servidores de acceso, los cuales notifican al balanceador de carga el ingreso de llamadas.

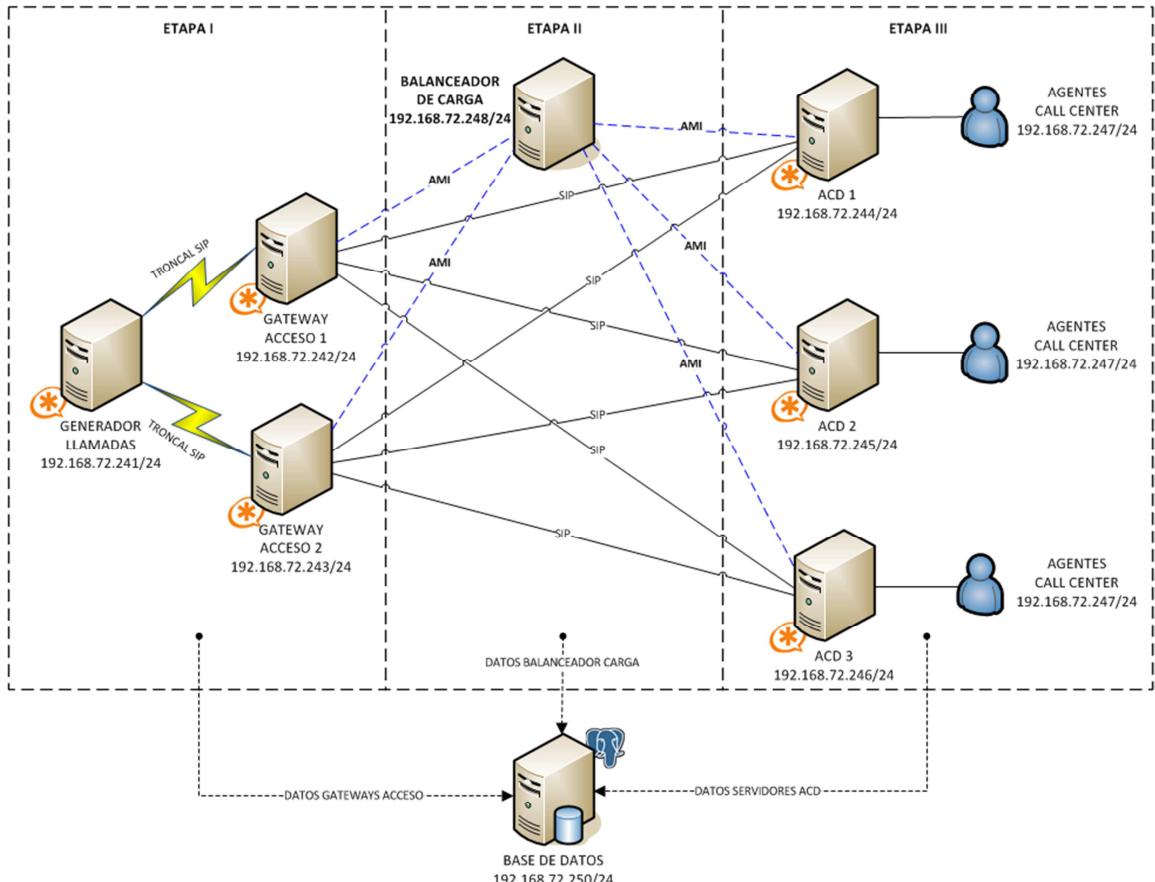


Figura 58. Arquitectura del prototipo del sistema

Con base en lo anterior, se construyó un prototipo que refleja la operación de un Call Center real bajo un ambiente virtualizado que permitió: una fácil y rápida implementación de las máquinas que componen el sistema, administración centralizada, configuración de un único servidor físico, etc.

Para la implementación de los servidores que conforman el sistema de Call Center, se utilizó la plataforma de virtualización VMware ESXi y a cada servidor se le instaló el sistema operativo CentOS, con su respectiva aplicación y configuración de red.

10.1 INSTALACIÓN DE ASTERISK

La versión de Asterisk escogida para este proyecto fue la 11.16.0 que tiene soporte a largo plazo (LTS: *Long Term Support*) por la empresa Digium, creadora de este software telefónico.

El proceso de instalación que se describe a continuación se realizó para cada uno de los servidores Asterisk que se muestran en la arquitectura del prototipo del sistema.

En primer lugar se descargó y descomprimió el software necesario utilizando los siguientes comandos:

```
# cd /usr/src/  
# wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz  
# tar -zvxf asterisk-11-current.tar.gz
```

La instalación de Asterisk requiere de ciertas dependencias que fueron instaladas previamente empleando el comando *yum install*:

```
yum install -y make openssl-devel ncurses-devel newt-devel libxml2-devel kernel-devel gcc gcc-c++ sqlite-devel
```

Una vez instaladas las dependencias, se procedió a compilar Asterisk de la siguiente manera:

```
./configure  
make menuconfig  
make  
make install  
make samples (crea los archivos de configuración de ejemplo)  
make config
```

Con los pasos anteriores, Asterisk ya estaría completamente instalado, sin embargo es indispensable configurar aspectos tan importantes como la definición de canales, el plan de marcación, el sistema de colas, entre otros. La configuración realizada se comenta en la siguiente sección.

10.2 CONFIGURACIÓN DE LOS SERVIDORES

10.2.1 Servidor Generador de Llamadas

Las llamadas de entrada son originadas por el servidor *Generador de Llamadas* hacia los *Gateways de Acceso*, utilizando un generador de tráfico desarrollado en el lenguaje de programación C++ y aplicando el mismo tráfico que se utilizó en el modelo de simulación. El programa desarrollado (*generador.cpp*) lee el archivo que contiene los tiempos en que se deben lanzar cada una de las llamadas, realiza una conexión a la interfaz AMI de Asterisk y envía la acción encargada de generar la llamada como se muestra en el siguiente fragmento de código:

```

AsteriskManagerWR<< "Action: Originate\r\n";
AsteriskManagerWR<< "ActionID: " + ActionID + "\r\n";
AsteriskManagerWR<< "Channel: " + Channel + "\r\n";
AsteriskManagerWR<< "Exten: s\r\n";
AsteriskManagerWR<< "Context: playback\r\n";
AsteriskManagerWR<< "Priority: 1\r\n";
AsteriskManagerWR<< "Timeout: 30000\r\n";
AsteriskManagerWR<< "CallerID: 1410\r\n\r\n";

```

Todas las llamadas son enviadas a los servidores de acceso a través de troncales SIP que son configuradas en el archivo *sip.conf* del servidor de la siguiente manera:

<i>[t_generador_to_gw1]</i>
<i>type=friend</i>
<i>username=t_generador_to_gw1</i>
<i>qualify=yes</i>
<i>host=192.168.72.242</i>
<i>context=no_entrante</i>
<i>disallow=all</i>
<i>allow=alaw</i>
<i>allow=ulaw</i>
<i>allow=gsm</i>
<i>dtmfmode=rfc2833</i>
<i>canreinvite=no</i>

<i>[t_generador_to_gw2]</i>
<i>type=friend</i>
<i>username=t_generador_to_gw2</i>
<i>qualify=yes</i>
<i>host=192.168.72.243</i>
<i>context=no_entrante</i>
<i>disallow=all</i>
<i>allow=alaw</i>
<i>allow=ulaw</i>
<i>allow=gsm</i>
<i>dtmfmode=rfc2833</i>
<i>canreinvite=no</i>

10.2.2 Servidores de Acceso

En el archivo *manager.conf* de cada servidor se creó el usuario *balanceador*, con contraseña *pwdbalanceador*. Este usuario es utilizado por el balanceador de carga para conectarse a la interfaz AMI. Únicamente se permite la lectura de eventos definidos por el usuario (*UserEvent*) y el envío de cualquier comando hacia Asterisk. La configuración es la siguiente:

<i>[balanceador]</i>
<i>secret=pwdbalanceador</i>
<i>read = user</i>
<i>write = system,call,agent,user,config,command,reporting,originate,message</i>
<i>permit = 0.0.0.0/0.0.0.0</i>

Por otro lado, en el archivo del plan de marcación *extensions.conf* de cada una de las máquinas, se configuró un evento de usuario *UserEvent* que indica al balanceador de carga que una llamada ha ingresado al sistema y que debe ser direccionada a una cola específica. Además, se definió el contexto donde se realiza la marcación (utilizando la aplicación *Dial* y una troncal SIP) al servidor de ACD seleccionado por el balanceador. La configuración realizada se muestra en el Anexo A-5.

Finalmente, a cada servidor de acceso se le configuró una troncal SIP hacia cada uno de los servidores de ACD, con el fin poder establecer un canal de comunicación entre ambos servidores. La configuración se hizo en el archivo *sip.conf* como se presenta a continuación:

```
[t_gwN_to_acdM]
type=friend
username=t_gwN_to_acdM
qualify=yes
host=192.168.72.X
context=no_entrante
disallow=all
allow=alaw
allow=ulaw
allow=gsm
dtmfmode=rfc2833
canreinvite=no
```

Donde:

- N: Número del servidor de acceso.
- M: Número del servidor de ACD.
- X: Octeto final de la IP del servidor de ACD.

10.2.3 Servidores de ACD

En el archivo *manager.conf* de cada servidor se creó el usuario *balanceador*, con contraseña *pwdbalanceador*, que tiene permisos para leer los eventos generados por la aplicación *app_queue* y los eventos generados por el usuario. Además, se puede enviar cualquier tipo de comando hacia Asterisk. La configuración se muestra a continuación:

```
[balanceador]
secret=pwdbalanceador
read = agent,user
write = system,call,agent,user,config,command,reporting,originate,message
permit = 0.0.0.0/0.0.0.0
```

La configuración de los agentes y colas en cada uno de los servidores se realizó en los archivos *agents.conf* y *queues.conf*. En el archivo *agents.conf* se definió un máximo de 25 agentes, que son adicionados a las colas dinámicamente ejecutando el comando “*queue add member Agent/XXXX to colaX*”. La configuración realizada fue la siguiente:

```
;Archivo agents.conf
[agents]
agent => 1000001,,Agente 1000001
agent => 1000002,,Agente 1000002
agent => 1000003,,Agente 1000003
agent => 1000004,,Agente 1000004
agent => 1000005,,Agente 1000005
agent => 1000006,,Agente 1000006
agent => 1000007,,Agente 1000007
agent => 1000008,,Agente 1000008
agent => 1000009,,Agente 1000009
agent => 1000010,,Agente 1000010
...
agent => 1000025,,Agente 1000025
```

En el archivo *queues.conf* se crearon 3 colas denominadas *cola1*, *cola2* y *cola3*, a las cuales se les habilitó el parámetro *eventwhencalled= yes* para que permitiera la generación de eventos al AMI. También se configuró la estrategia de cola como *leastrecent* de manera que las llamadas en el ACD local fueran enviadas al agente con mayor tiempo disponible.

Finalmente, se configuró el plan de marcación en cada uno de los servidores de ACD de manera que cuando una llamada es recibida pueda ser procesada por el módulo *app_queue* de Asterisk.

```
[from_gateway]
exten => _X,1,Answer()
same => n,NoOp(Llamada direccionada a la cola ${EXTEN})
same => n,Set(CDR(userfield)=${SIP_HEADER(X-Asterisk-UniqueID)});Se captura el UniqueID del servidor de acceso
same => n,Gotofff(${QueueID}" = "1"?id1)
same => n,Gotofff(${QueueID}" = "2"?id2)
same => n,Gotofff(${QueueID}" = "3"?id3)
same => n,Log(ERROR, QUEUEID ${QueueID} NO ENCONTRADO)
same => n,Hangup()
same => n(id1),Set(Queue=cola1)
same => n,Goto(from_gateway, ${EXTEN},encolar)
same => n(id2),Set(Queue=cola2)
same => n,Goto(from_gateway, ${EXTEN}, encolar)
same => n(id3),Set(Queue=cola3)
same => n,Goto(from_gateway, ${EXTEN}, encolar)
same => n(encolar),NoOp(Encolar llamada)
same => n,Set(Queuelength=${QUEUE_WAITING_COUNT(${Queue})})
same => n,UserEvent(AverageQueueLength, Channel: ${CHANNEL}, Queue: ${Queue}, QueueLength: ${QueueLength}, UniqueID: ${UNIQUEID})
same => n,AGI(agi-tiempo-servicio)
same => n,NoOp(ServiceTime: ${ServiceTime})
same => n,Queue(${Queue},cht,,,set_service_time.agi)
same => n,Hangup()
```

Con el objetivo de fijar el tiempo de servicio para cada llamada, se crearon dos programas en C++ que hacen uso de la interfaz *Asterisk Gateway Interface* (AGI). El primer programa se denominada *agi-tiempo-servicio.cpp* y es el encargado de fijar, como variable de canal, el tiempo de servicio de la llamada que se obtiene de la base de datos. Mientras el segundo programa *set-service-time.agi* es

ejecutado por medio de la aplicación *Queue* para que la llamada sea colgada t segundos después de ser contestada por un agente conectado a alguno de los servidores de ACD.

10.2.4 Servidor de Base de Datos

Con el objetivo de registrar el detalle de las llamadas en cada una de las etapas del sistema, se instaló y configuró una base de datos PostgreSQL, debido a sus características técnicas entre las cuales se encuentran: estabilidad, potencia, robustez, facilidad de administración e implementación de estándares, manejo de grandes cantidades de datos y una alta concurrencia de usuarios accediendo simultáneamente. Además, PostgreSQL es una de las bases de datos soportada por la plataforma de software libre Asterisk.

Una vez instalada y configurada la versión 9.4 de PostgreSQL, se procedió a crear una base de datos denominada *bd_tesis*, sobre la cual se configuraron los esquemas y tablas para almacenar los datos de las pruebas de cada uno los algoritmos de balanceo de carga bajo los escenarios homogéneo y heterogéneo. En la siguiente figura se muestra la configuración realizada:

Schema	Owner	Comment
grr_heterogeneo	postgres	
grr_homogeneo	postgres	
jsq_heterogeneo	postgres	
jsq_homogeneo	postgres	
med_heterogeneo	postgres	
med_homogeneo	postgres	
minasa_heterogeneo	postgres	
minasa_homogeneo	postgres	
public	postgres	
servidores_acd	postgres	

Figura 59. Esquemas y tablas de la base de datos

La segmentación por esquemas facilitó el análisis de los datos, ya que se logró obtener el tamaño promedio de la cola, el tiempo promedio de espera y la utilización de los agentes, de cada uno de los servidores de ACD. Adicionalmente, permitió registrar el comportamiento de los algoritmos de balanceo de carga, al igual que el detalle de las llamadas que cursan por los servidores Asterisk. A continuación se muestra la descripción de cada una de las tablas:

Nombre Tabla	Descripción
avg_queue_length	Almacena el tamaño de la cola en un momento

	determinado, al igual que tamaño promedio en el tiempo.
balanceador_log	Registra en tiempo real los siguientes datos de cada uno de los servidores de ACD: <ul style="list-style-type: none"> • Agentes conectados. • Agentes disponibles. • Llamadas en cola. • Tiempo promedio de espera (ASA). • Tiempo promedio de servicio (AHT). • Tiempo de espera de la primera llamada en cola. También almacena el algoritmo configurado, el número de llamadas que hay en el sistema y el ACD seleccionado.
cdr_acd	Registra el detalle de las llamadas que ingresan a los servidores de ACD.
cdr_generador	Registra el detalle de las llamadas del generador de tráfico.
cdr_gw	Registra el detalle de las llamadas de los servidores o <i>gateways</i> de acceso.
cdr_login	Registra el detalle de las llamadas del servidor utilizado para emular las conexiones de los agentes del Call Center.
contador_llamadas	Esta tabla es empleada para almacenar el número de llamadas que ingresan a cada uno de los servidores de ACD.
queue_log	Almacena en tiempo real los datos del sistema de colas de cada uno de los servidores de ACD.

Tabla 15. Descripción de las tablas de la base de datos

11. PRUEBAS DEL PROTOTIPO DEL SISTEMA

11.1 AMBIENTE DE PRUEBAS

El propósito de las pruebas es mostrar el funcionamiento del balanceador de carga desarrollado en un ambiente real controlado, utilizando tráfico real (señalización y voz) mediante el uso de agentes de Call Center simulados con un servidor Asterisk. Las características bajo las cuales se realizaron las pruebas del prototipo, son las mismas que se mencionaron en la sección 7.2 de este documento, a excepción de la duración de la jornada laboral que para el prototipo fue de 12 horas.

Como se muestra en la Figura 58, las llamadas entrantes se generan desde el servidor *Generador de Llamadas* hacia los *Gateways de Acceso*. Cada una de las llamadas que fluyen a través de la plataforma son distribuidas por el balanceador de carga entre los 3 servidores de ACD al ejecutar el algoritmo seleccionado.

Mediante la conexión persistente al AMI de cada uno de los servidores (línea punteada en azul) se mantiene la comunicación entre los servidores y el balanceador. Una vez el balanceador de carga decide a qué servidor de ACD pasar la llamada, esta última es enviada al servidor de ACD seleccionado a través de una troncal SIP que se programa estáticamente en cada uno de los

servidores o *gateways* de acceso. De esta manera la voz y la señalización de la llamada se establecen entre el par de servidores.

Cuando la llamada es direccionada a uno de los servidores de ACD, esta es enviada a la cola y el ACD de cada máquina es el encargado de distribuir el tráfico entre los agentes conectados a dicho servidor. Los agentes se emularon a través de una llamada persistente entre un servidor Asterisk (en la Figura 58 se muestra como la IP 192.168.72.247) y cada uno de los servidores de ACD. Por lo tanto si se necesitan conectar 23 agentes a un servidor de ACD, se genera ese mismo número de llamadas desde el servidor Asterisk configurado para tal fin.

Todos los eventos del sistema son registrados en tiempo real en una base de datos PostgreSQL.

11.2 ANÁLISIS DE RESULTADOS

Los parámetros de medición utilizados para el análisis de los datos obtenidos mediante el prototipo, son los mismos que para el modelo de simulación (ver sección 7.3). A continuación se muestran los resultados de las pruebas realizadas en el escenario real durante períodos de 12 horas:

11.2.1 Tamaño Promedio de la Cola

En las Figuras 60, 61 y 62 se puede observar que bajo un escenario homogéneo de 23 agentes conectados a cada uno de los servidores de ACD, los algoritmos JSQ y MED presentan el menor tamaño promedio de la cola.

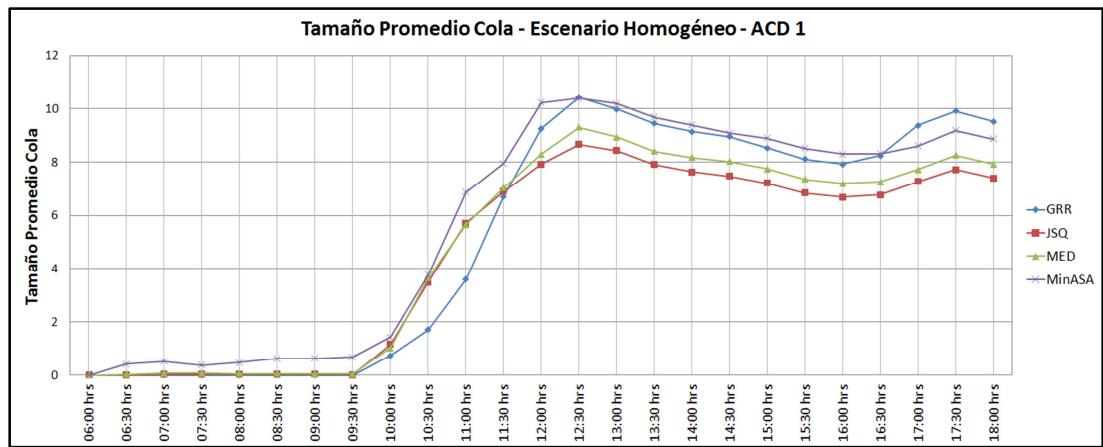


Figura 60. Tamaño promedio de la cola del servidor ACD1 bajo el escenario homogéneo – Prototipo del sistema

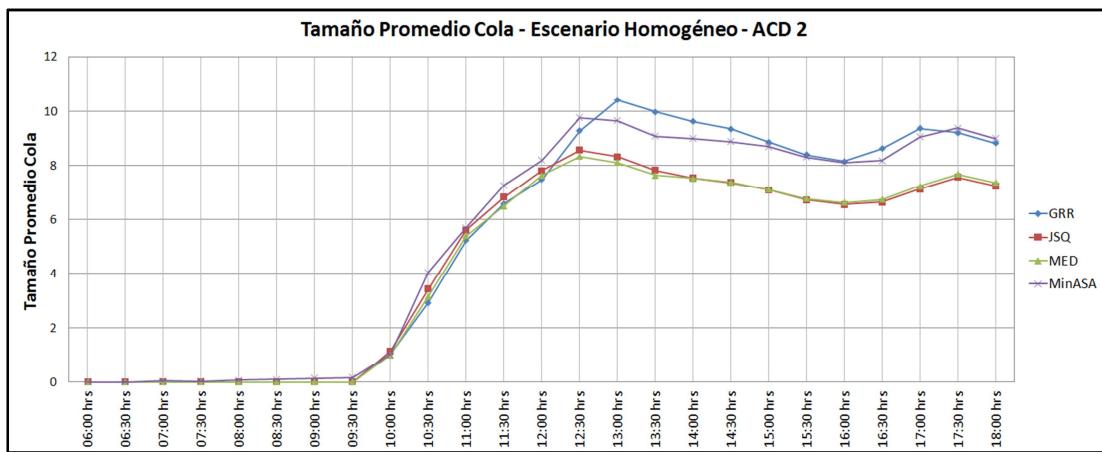


Figura 61. Tamaño promedio de la cola del servidor ACD2 bajo el escenario homogéneo – Prototipo del sistema

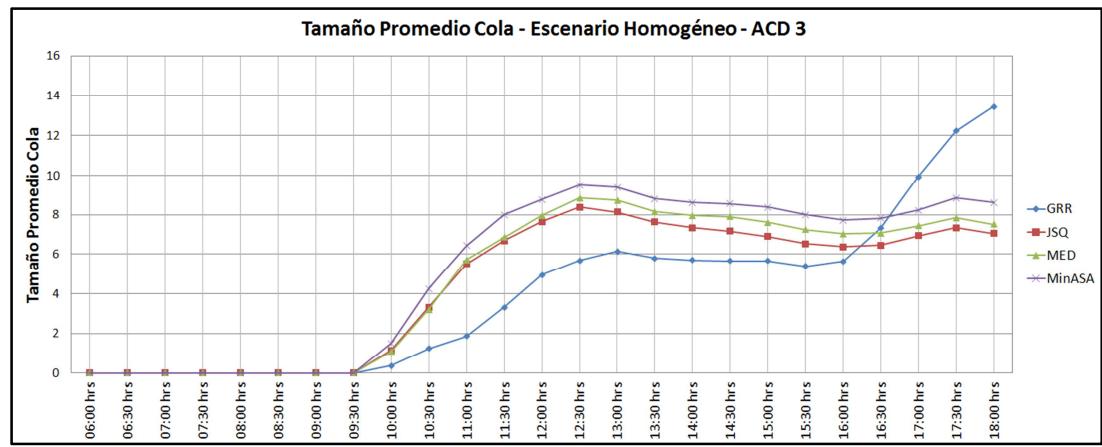


Figura 62. Tamaño promedio de la cola del servidor ACD3 bajo el escenario homogéneo – Prototipo del sistema

Al comparar los resultados obtenidos bajo un escenario heterogéneo tanto en el modelo de simulación como en el prototipo, se evidenció una gran similitud en el tamaño promedio de cola de cada uno de los servidores de ACD, lo que indica el buen funcionamiento que tiene el balanceador de carga desarrollado y la confiabilidad de la implementación de Call Centers sobre una plataforma de software libre como Asterisk. En el Anexo A-6 se muestran los resultados del tamaño promedio de cola en cada uno de los algoritmos de balanceo de carga.

11.2.2 Tiempo Promedio de Espera en Cola

Desde el punto de vista del usuario que llama a un Call Center, el tiempo promedio de espera en cola es quizás la métrica más importante, es por ello que se realizó la comparación entre el modelo simulación y el prototipo implementado. Los resultados obtenidos, ratifican nuevamente el buen desempeño del balanceador de carga al aplicar los 4 algoritmos. Aunque las pruebas se llevaron a cabo en escenarios tanto homogéneos como heterogéneos, únicamente se muestran los resultados obtenidos bajo el escenario heterogéneo, ya que este evidencia el carácter dinámico de un sistema de Call Center.

11.2.2.1 Algoritmo GRR

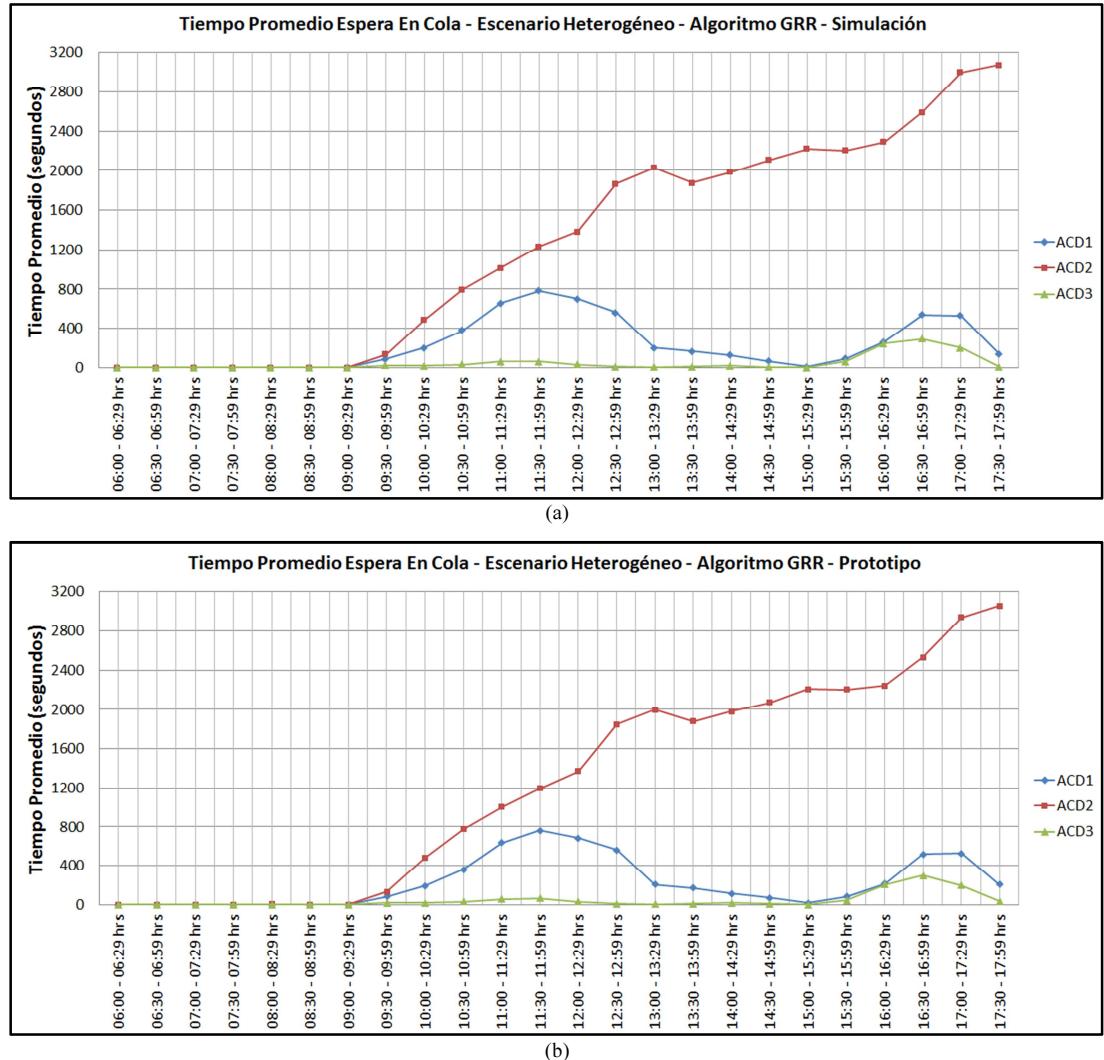


Figura 63. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo GRR bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo

Tanto en el prototipo del sistema como en el modelo de simulación se puede apreciar que los clientes que son direccionados al servidor de ACD2 esperan más tiempo en cola que aquellos que son atendidos en los otros dos servidores de ACD. Este comportamiento se debe a que el segundo servidor tiene menos agentes atendiendo llamadas y el algoritmo GRR envía las llamadas que ingresan al sistema a los diferentes servidores de ACD de manera rotativa.

11.2.2.2 Algoritmo JSQ

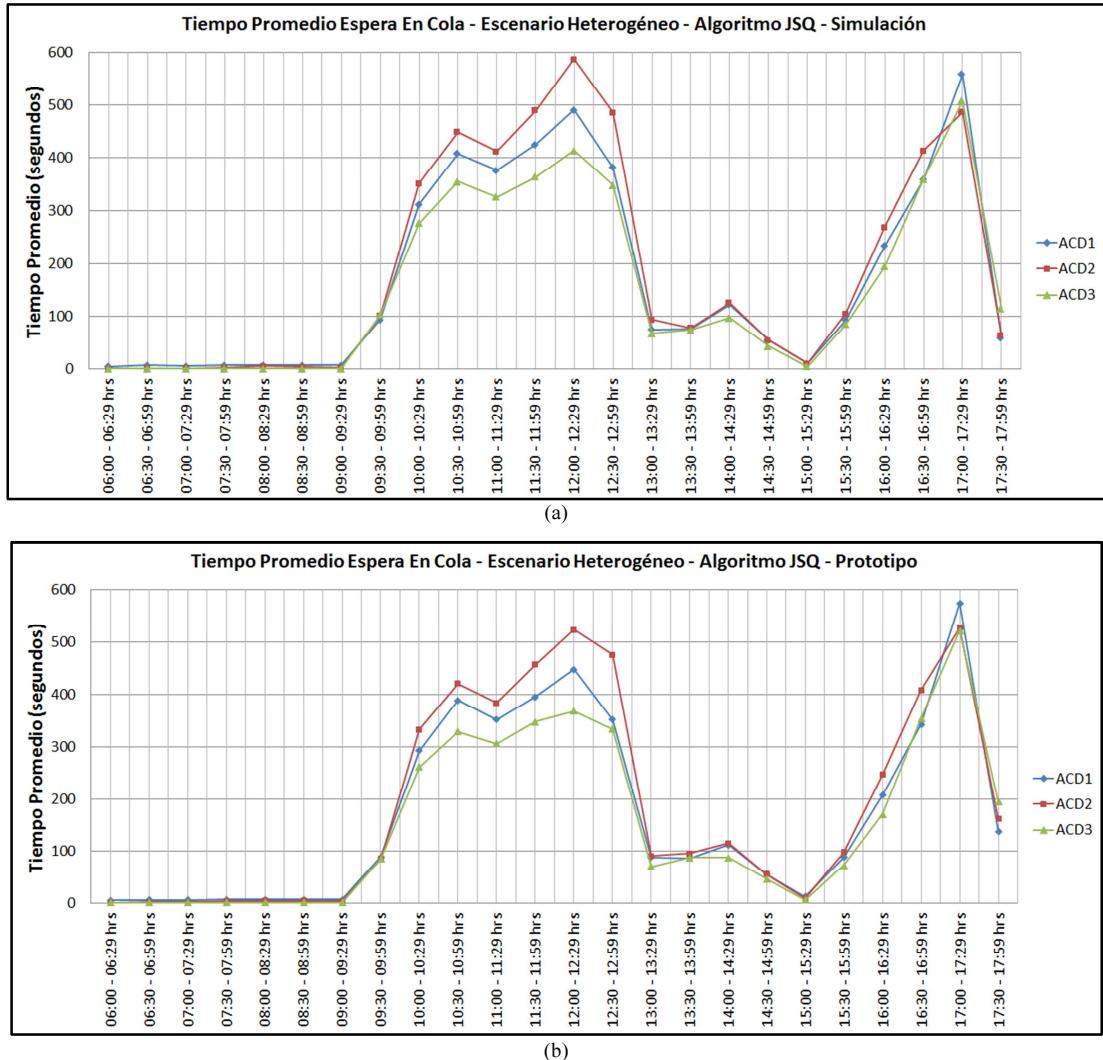


Figura 64. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo JSQ bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo

El tiempo promedio que esperan los clientes en cola bajo el modelo real (prototipo) es un poco menor que en el modelo de simulación entre las 10:30 am y 1:00 pm. Además, durante estas horas se puede observar un comportamiento similar al encontrado en el algoritmo GRR, ya que los clientes que son atendidos en el ACD 2 esperan mayor tiempo en cola.

11.2.2.3 Algoritmo MED

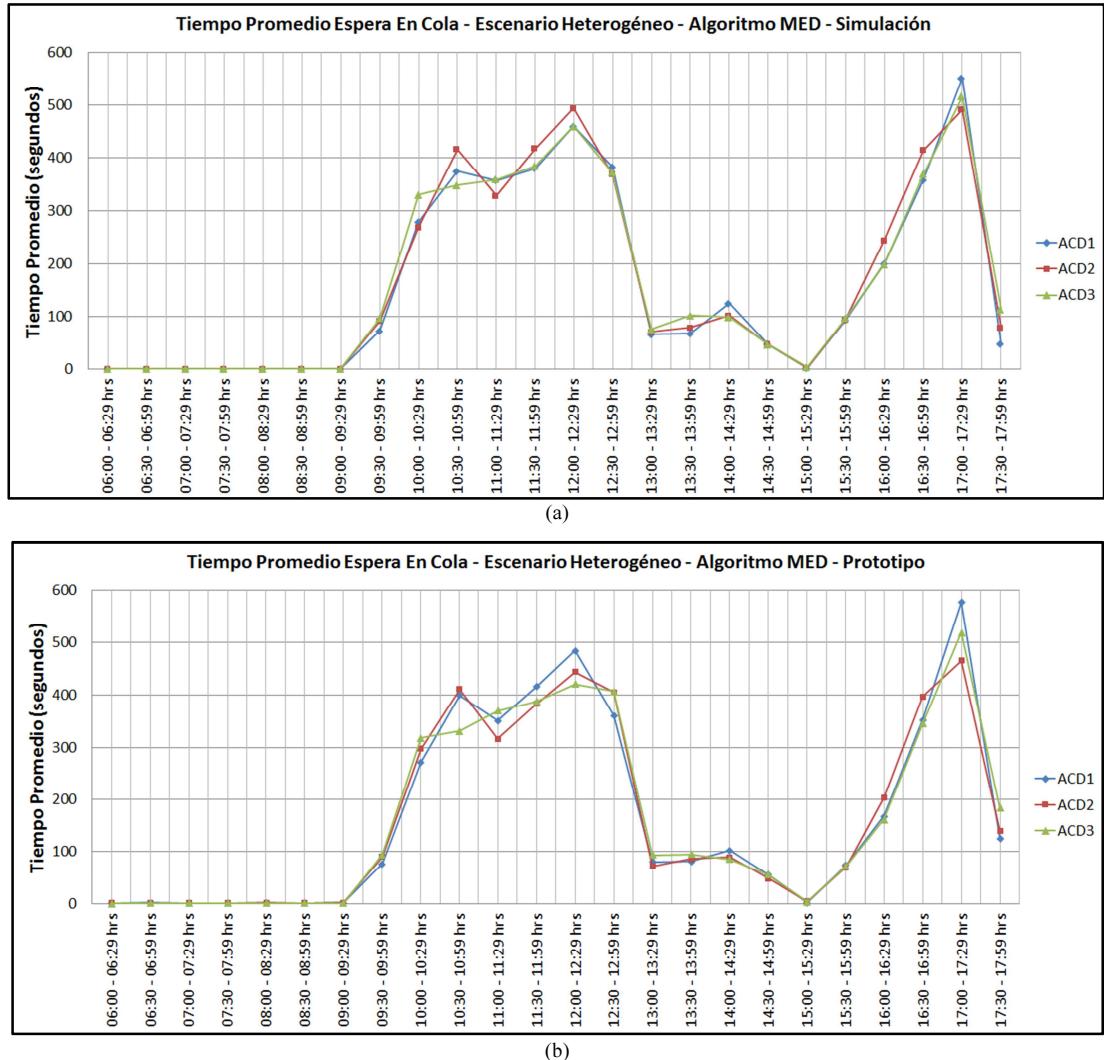


Figura 65. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo MED bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo

De acuerdo con las gráficas anteriores, bajo el algoritmo del mínimo retardo esperado se garantiza que los clientes que llaman al Call Center van a esperar en cola tiempos muy parecidos sin importar a que ACD sean direccionados. Este comportamiento refleja el buen funcionamiento del algoritmo y del balanceador de carga desarrollado.

11.2.2.4 Algoritmo Min ASA

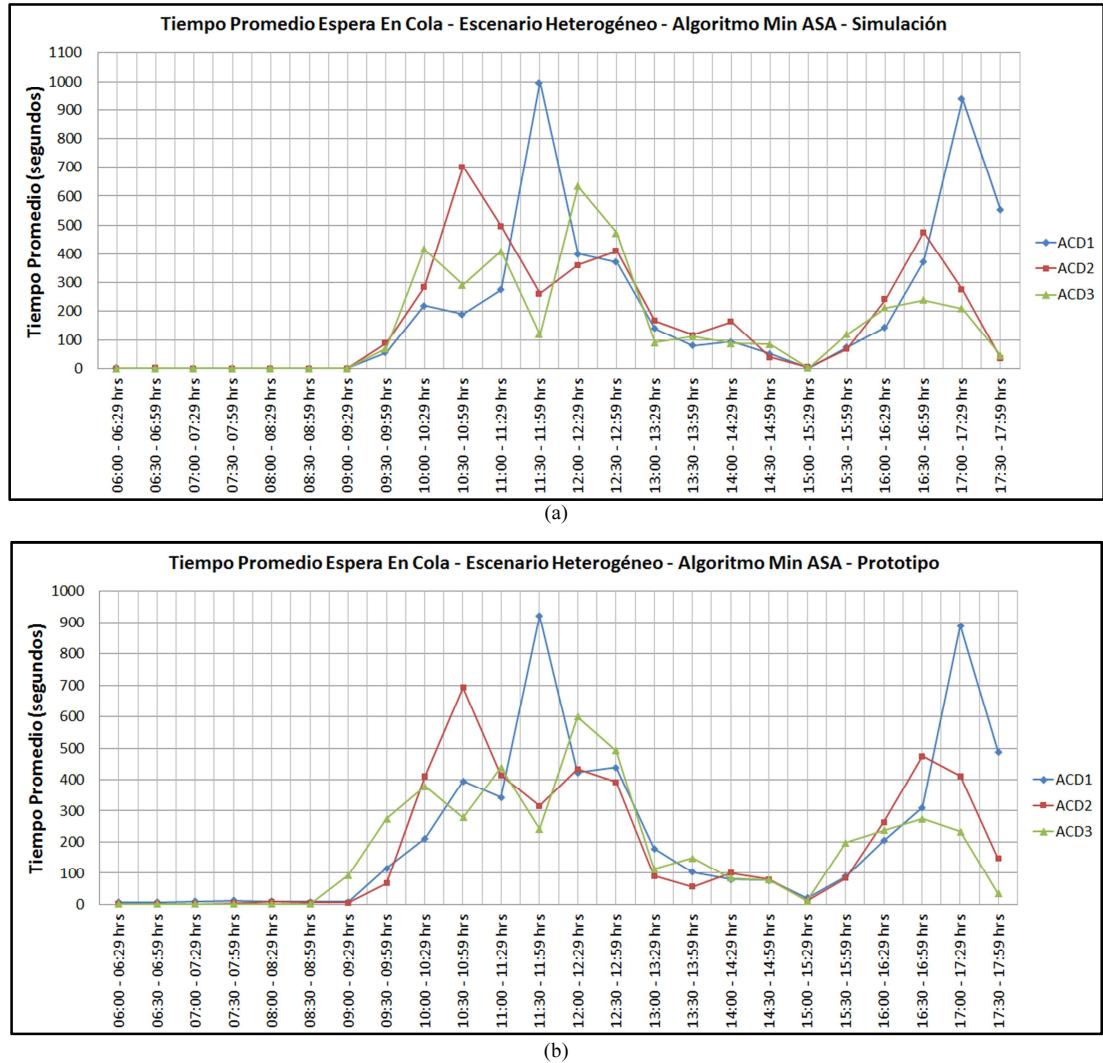


Figura 66. Tiempo promedio de espera en los servidores de ACD utilizando el algoritmo Min ASA bajo el escenario heterogéneo, (a) representa los resultados de la simulación, mientras (b) representa los resultados del prototipo

A diferencia del algoritmo MED, el algoritmo Min ASA no garantiza tiempos de espera homogéneos, ya en algún momento de la jornada cualquiera de los servidores de ACD puede tener el menor tiempo promedio de espera, sin embargo en un lapso de tiempo posterior un cliente direccionado al mismo ACD puede esperar un mayor tiempo.

11.2.3 Utilización de los Agentes

Como se mencionó anteriormente, esta métrica es de gran importancia, ya que los recursos deben ser utilizados el mayor tiempo posible y de forma homogénea en todos los servidores de ACD.

Durante las pruebas realizadas tanto en el modelo de simulación como en el modelo real (prototipo), se encontró una gran falencia de los algoritmos JSQ, MED y Min ASA, debido a que presentan una

mala administración de los recursos asignados a cada uno de los servidores de ACD, pues en las primeras horas y al finalizar la jornada laboral subutilizan los agentes que se encuentra conectados en el servidor ACD3 y sobrecargan los agentes del servidor ACD1. En los diagramas de barras siguientes se presenta el promedio de utilización durante la jornada laboral tanto en el escenario homogéneo como en el heterogéneo:

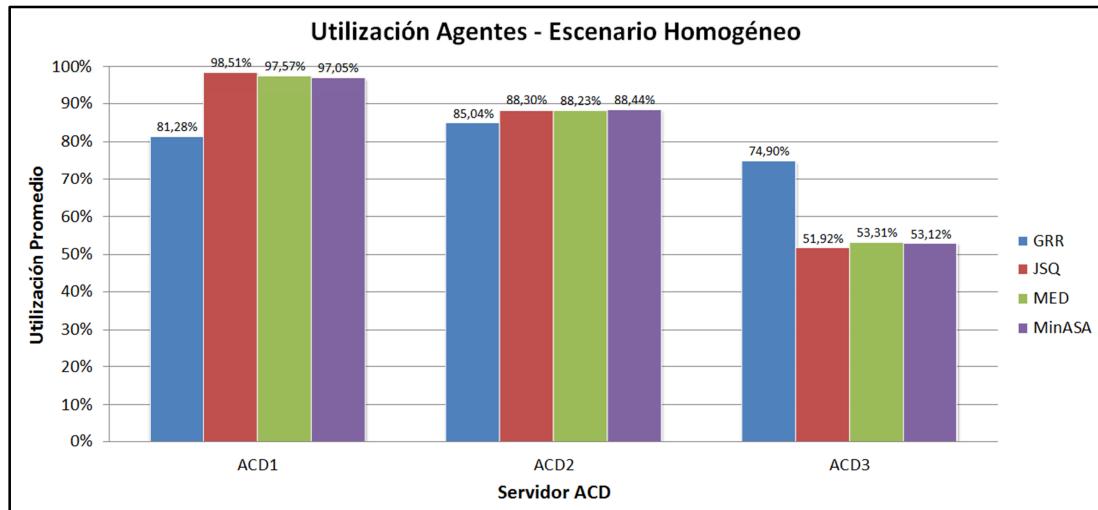


Figura 67. Utilización promedio de los agentes en cada uno de los servidores de ACD bajo el escenario homogéneo – Prototipo del sistema

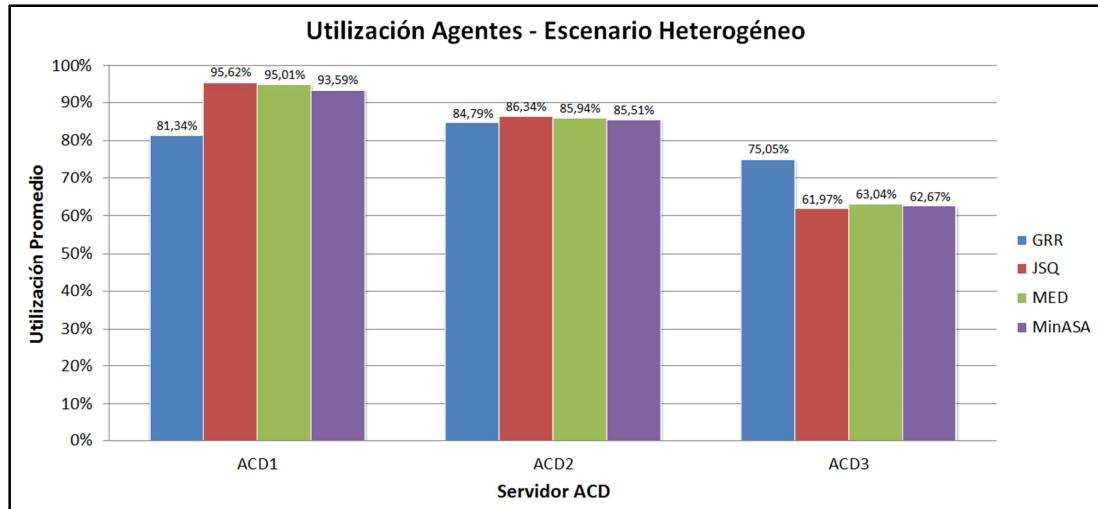


Figura 68. Utilización promedio de los agentes en cada uno de los servidores de ACD bajo el escenario heterogéneo – Prototipo del sistema

11.3 ESCALABILIDAD DEL SISTEMA

Una de las características principales de la solución, es la escalabilidad. Es por ello que para verificar dicha característica se implementaron 3 tipos de escenarios:

Escenario	Número de Servidores ACD	Número de Agentes por Servidor	Total de Agentes Conectados
1	2	10	20
2	3	10	30
3	4	10	40

Tabla 16. Escenarios implementados para la verificación de la escalabilidad del sistema

A cada escenario se le envió un tráfico de 80 llamadas (aproximadamente una llamada cada segundo) y se configuró el balanceador de carga con el algoritmo de turno rotativo GRR para que se hiciera una distribución equitativa de las llamadas. Además, a cada llamada se le estableció un tiempo de servicio de 60 segundos. Los resultados obtenidos se muestran a continuación:

- **Escenario 1**

Servidor ACD	Número de Llamadas Contestadas	Tiempo Promedio de Espera (segundos)
1	40	80,65
2	40	81,40

Tabla 17. Escalabilidad del sistema – Resultados del primer escenario

- **Escenario 2**

Servidor ACD	Número de Llamadas Contestadas	Tiempo Promedio de Espera (segundos)
1	27	43,56
2	27	43,56
3	26	41,35

Tabla 18. Escalabilidad del sistema – Resultados del segundo escenario

- **Escenario 3**

Servidor ACD	Número de Llamadas Contestadas	Tiempo Promedio de Espera (segundos)
1	20	22,40
2	20	22,35
3	20	22,40
4	20	22,40

Tabla 19. Escalabilidad del sistema – Resultados del tercer escenario

La implementación de este tipo de escenarios demuestra que el sistema es escalable, pues el balanceador de carga desarrollado puede estar conectado a múltiples servidores de ACD y permitir la conexión de un mayor número de agentes, mejorando los tiempos de espera de los clientes.

La característica clave del balanceador es que la carga adicional sólo requiere de más recursos de hardware en lugar de una modificación extensiva de la aplicación.

12. CONCLUSIONES

Como resultado general de las simulaciones realizadas, se obtuvo un mejor rendimiento utilizando el algoritmo MED, sin embargo el algoritmo JSQ presenta resultados bastante cercanos bajo los dos escenarios de simulación, convirtiéndolo en otra alternativa para el balanceo de carga de llamadas.

Al comparar los resultados obtenidos bajo un escenario heterogéneo tanto en el modelo de simulación como en el prototipo, se evidenció una gran similitud, lo que indica el buen funcionamiento que tiene el balanceador de carga desarrollado y la confiabilidad de la implementación de Call Centers sobre una plataforma de software libre como Asterisk.

A partir de los algoritmos *Minimum Expected Delay* (MED) y *Generalized Round Robin* (GRR), se propuso un algoritmo híbrido denominado *Minimum Expected Delay Modified* (MEDM), que mejora la capacidad de utilización de los agentes en cada uno de los servidores de ACD, sin afectar drásticamente el tiempo promedio que deben esperar los clientes y el tamaño promedio de la cola de atención.

Se demostró que el balanceador de carga de llamadas, permite que sistemas de Call Center implementados bajo Asterisk, sean escalables, ya que puede conectarse a múltiples servidores de ACD sin afectar su funcionamiento. La característica clave del balanceador es que la carga adicional sólo requiere de más recursos de hardware en lugar de una modificación extensiva de la aplicación.

El contar con un balanceador de carga de llamadas para sistemas de Call Center multicola implementados sobre la plataforma de software libre Asterisk, abre nuevas posibilidades para el mejoramiento de la atención al cliente. Los resultados obtenidos tanto en las simulaciones como en el prototipo son satisfactorios, y establecen un punto de partida para la búsqueda y desarrollo de nuevos algoritmos que involucren mayor nivel de inteligencia y mejoren el desempeño del balanceador de carga.

13. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN

- [1] Morris, Eddie; Ancajima, Alfredo; Chiri, Carlos; Galindo, Juan; Guido, Carlos; Mejía, Enrique. *Servicios de contact center basados en offshore outsourcing*. Cap. 1. Universidad ESAN – Lima, 2009.
- [2] Gans, Noah; Koole, Ger; Mandelbaum, Avishai. *Telephone Call Centers: Tutorial, Review, and Research Prospects*. Marzo 2003.
- [3] Martínez Eraso, Camilo Ernesto. *Análisis de redes de colas modeladas con tiempos entre llegadas exponenciales e hiper erlang para la asignación eficiente de los recursos*. Trabajo de Grado. Pontificia Universidad Javeriana, Bogotá. 2009.
- [4] Adetunji, A.; Shahrabi, A.; Larijani, H.; Mannion, M. *Performance Comparison Of Call Routing Algorithms Over Virtual Call Centre*. School of Computing and Mathematical Sciences, Glasgow Caledonian University, UK. The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Septiembre 2007.
- [5] Cisco Systems, Inc. *Cisco ICM Software Release 4.6.2 Script Editor Guide - Target Selection*. [En línea] http://www.cisco.com/en/US/products/sw/custcosw/ps1001/products_user_guide_chapter09186a00803bd167.html [Consulta: 8 Marzo 2014].
- [6] Cisco Systems, Inc. *Understanding Expected Delay (ED)*. [En línea] <http://www.cisco.com/c/en/us/support/docs/voice-unified-communications/icm-router-software/29522-expected-delay.html> [Consulta: 9 Marzo 2014].
- [7] Capdehourat, Germán. *Análisis y Diseño de Call Centers*. Trabajo de Evaluación de Performance de Redes de Telecomunicaciones, 2006.
- [8] Norma COPC (*Costumer Operation Performance Center*) 2000 PSIC.
- [9] Naranjo, Juan Fernando. *Modelo para la Gestión de Información en un Centro de Contactos*. Universidad Nacional de Colombia – Sede Medellín, 2009.
- [10] The MathWorks, Inc. *Compute a Time Average of a Signal*. [En línea] <http://www.mathworks.com/help/simevents/ug/deriving-custom-statistics.html#bqm4scv-1>
- [11] EasyErlang. *Introduction To Traffic Modeling And Resource Allocation In Call Centers*. [En línea] <http://www.easyerlang.com/pdfs/call-center-basics.pdf> [Consulta: 17 Agosto 2014]
- [12] Deutsh, Douglas A.; Smith, David B.; Smith, Matthew R. *Automatic Call Distribution Center With Queue Position Restoration For Call-Back Customers*. Patente - US 6724885 B1. 20 Abril 2004.
- [13] Gorrotxategi, Gorka; Baz, Iñaki. *Voz sobre IP y Asterisk*. IRONTEC, Internet y Sistema sobre GNU/Linux. [En línea] <http://www.irontec.com/files/cursoAsteriskVozIP-3-introduccionAsterisk.pdf> [Consulta: 23 Agosto 2014]
- [14] Arquitectura de Asterisk. [En línea] <http://www.wikiasterisk.com/index.php/Arquitectura> [Consulta: 11 Agosto 2014].
- [15] Bryant, Russell; Madsen, Leif; Meggelen, Jim Van. *Asterisk, The Future Of Telephony Is Now*. O’ Reilly, 4th Edition.
- [16] Digium, The Asterisk Company. *Asterisk Architecture*. [En línea] <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Architecture> [Consulta: 11 Agosto 2014].
- [17] Voip Info. *Asterisk Configuration Files*. [En línea] <http://www.voip-info.org/wiki/view/Asterisk+config+files> [Consulta: 24 Agosto 2014].

- [18] Iseki, Fumikazu; Sato, Yuki; Kim, Moo Wan. *VoIP System based on Asterisk for Enterprise Network*. ICACT, Febrero 13 - 16 de 2011.
- [19] The MathWorks, Inc. *SimEvents User's Guide R2014b*. 2014.
- [20] The MathWorks, Inc. *SimEvents Documentation*. [En línea] <http://www.mathworks.com/help/simevents/index.html?refresh=true> [Consulta: 20 Septiembre 2014].
- [21] The MathWorks, Inc. *Video: Simulación de Eventos Discretos*. [En línea] http://www.mathworks.com/videos/discrete-event-and-hybrid-modeling-with-simevents-82003.html?form_seq=conf546&verified [Consulta: 28 Septiembre 2014].
- [22] Ariadne Training. *UML Applied – Second Edition: Object Oriented Analysis and Design*. 2005.
- [23] Canchala, Armando. *UML, ejemplo sencillo sobre Modelado de un Proyecto*. Microsoft Developer Network. [En línea] <http://msdn.microsoft.com/es-es/library/bb972214.aspx#XSLTsection125121120120> [Consultado: 26 Octubre 2014].
- [24] Mehrotra, Vijay; Fame, Jasson. *Call Center Simulation Modeling: Methods, Challenges, and Opportunities*. Proceedings of the 2003 Winter Simulation Conference, IEEE.
- [25] Capers, Walter. *Creating a C++ Thread Class*. Code Project. [En línea] <http://www.codeproject.com/Articles/21114/Creating-a-C-Thread-Class>
- [26] Tougher, Rob. *Linux Socket Programming In C++*. Linux Gazette. [En línea] <http://tldp.org/LDP/LG/issue74/tougher.html>
- [27] Wallace, Rodney B.; Saltzman, Robert M. *Comparing Skill-Based routing Call Center Simulations usign C Programming and Arena Models*. Proceedings of the 2005 Winter Simulation Conference IEEE.

ANEXOS

A-1 Principales archivos de configuración de Asterisk.

Nombre del Archivo	Descripción
<i>modules.conf</i>	Se definen los módulos que se desean cargar al iniciar Asterisk.
<i>extensions.conf</i>	Se define el plan de marcación (<i>dialplan</i>). En este archivo es donde se configuran las acciones que realizarán cada una de las llamadas.
<i>chan_dahdi.conf</i>	Se define y configuran los canales que utilizaran las tarjetas de telefonía análoga y/o digital.
<i>sip.conf</i>	Se definen las cuentas y los parámetros del protocolo SIP.
<i>iax2.conf</i>	Se definen las cuentas y los parámetros del protocolo IAX2.
<i>agents.conf</i>	Se crean los agentes que harán parte del distribuidor automático de llamadas (ACD).
<i>queues.conf</i>	Se crean las colas que se manejaran en el distribuidor automático de llamadas (ACD).
<i>manager.conf</i>	Se crean las cuentas y se configuran los parámetros de la interfaz AMI.
<i>cdr_mysql.conf</i>	Se realiza la configuración de la base de datos MySQL donde se almacenará el CDR y el queueelog.
<i>cdr_pgsql.conf</i>	Se realiza la configuración de la base de datos PostgreSQL donde se almacenará el CDR y el queueelog.

Tabla 20. Principales archivos de configuración de Asterisk

A-2 Simulación de la utilización de agentes bajo un escenario homogéneo en el que se conectan 23 agentes a cada servidor de ACD.

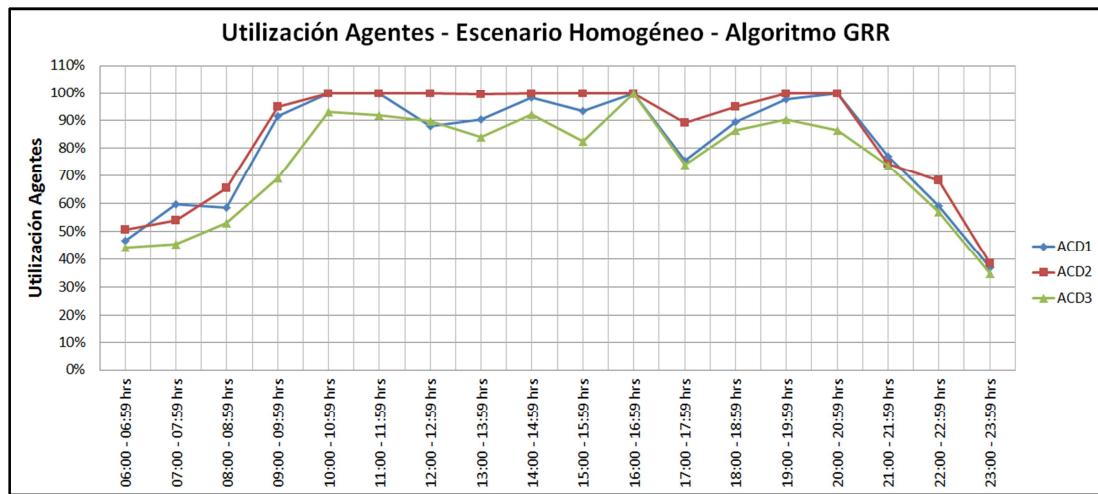


Figura 69. Utilización de los agentes empleando el algoritmo GRR bajo el escenario homogéneo – Modelo de simulación

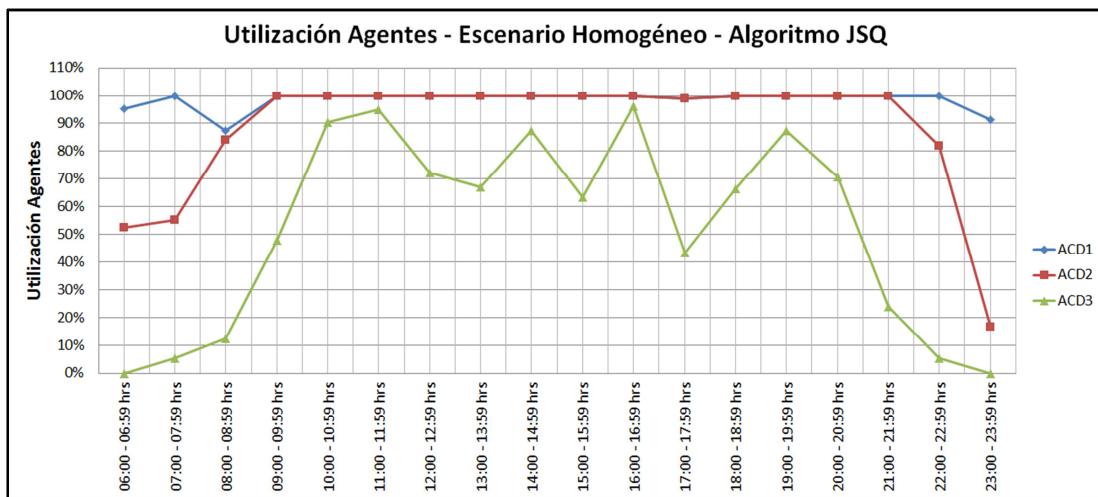


Figura 70. Utilización de los agentes empleando el algoritmo JSQ bajo el escenario homogéneo – Modelo de simulación

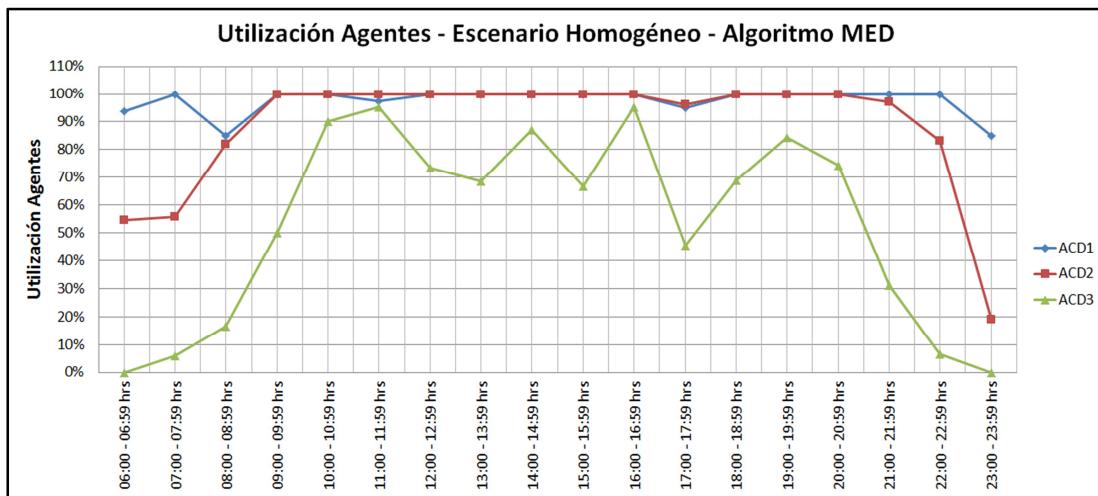


Figura 71. Utilización de los agentes empleando el algoritmo MED bajo el escenario homogéneo – Modelo de simulación

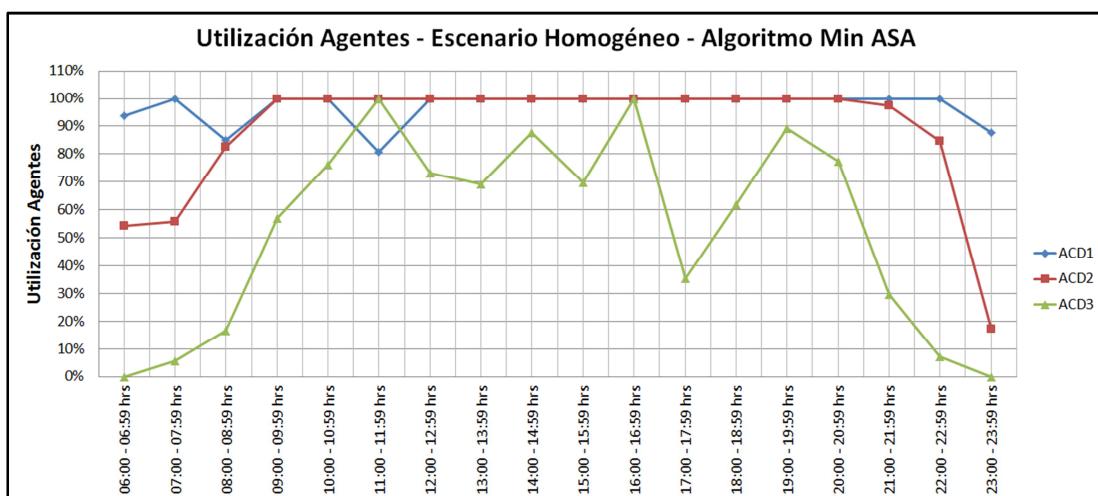


Figura 72. Utilización de los agentes empleando el algoritmo Min ASA bajo el escenario homogéneo – Modelo de simulación

A-3 Simulación de la utilización de agentes bajo un escenario heterogéneo en el cual se conectan 22 agentes al servidor ACD1, 20 agentes al servidor ACD2 y 25 agentes al servidor ACD3.

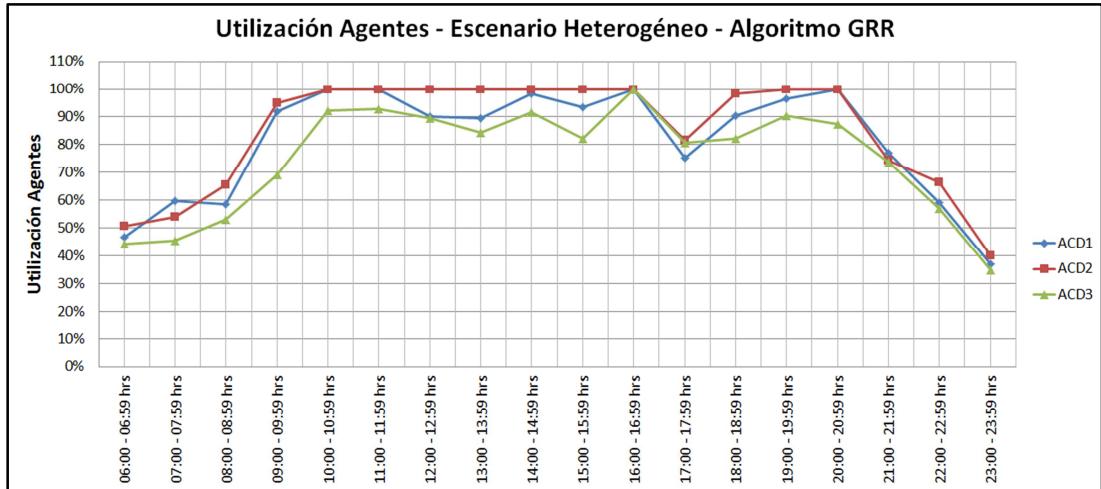


Figura 73. Utilización de los agentes empleando el algoritmo GRR bajo el escenario heterogéneo – Modelo de simulación

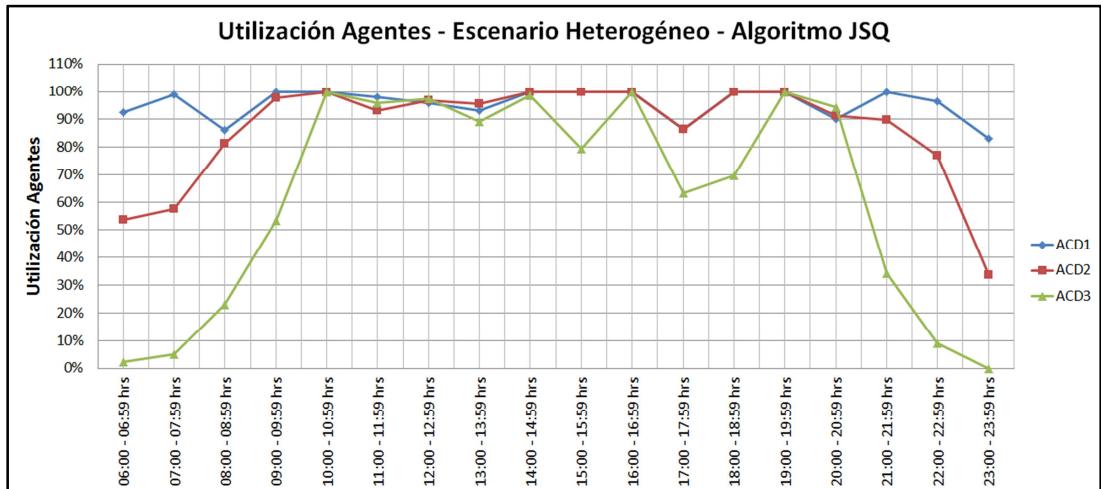


Figura 74. Utilización de los agentes empleando el algoritmo JSQ bajo el escenario heterogéneo – Modelo de simulación

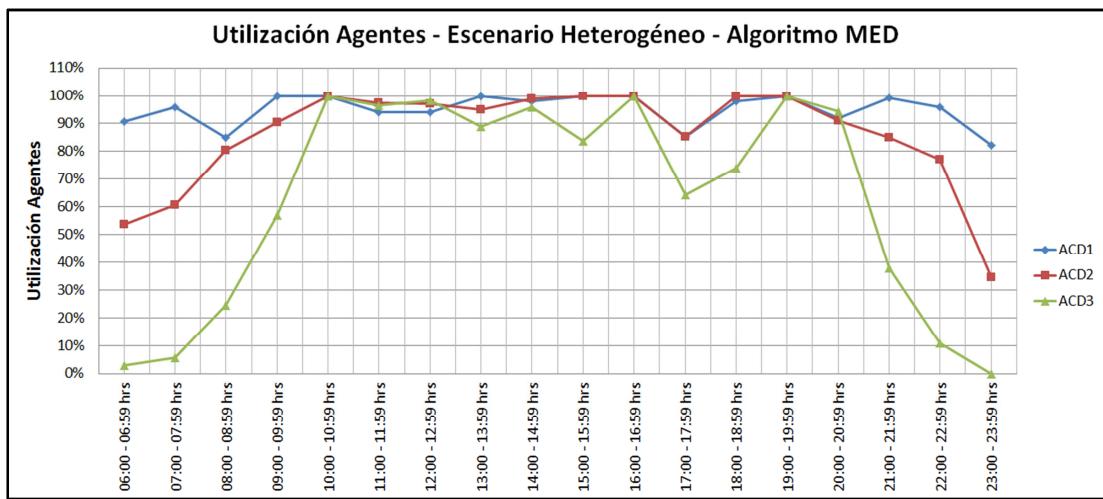


Figura 75. Utilización de los agentes empleando el algoritmo MED bajo el escenario heterogéneo – Modelo de simulación

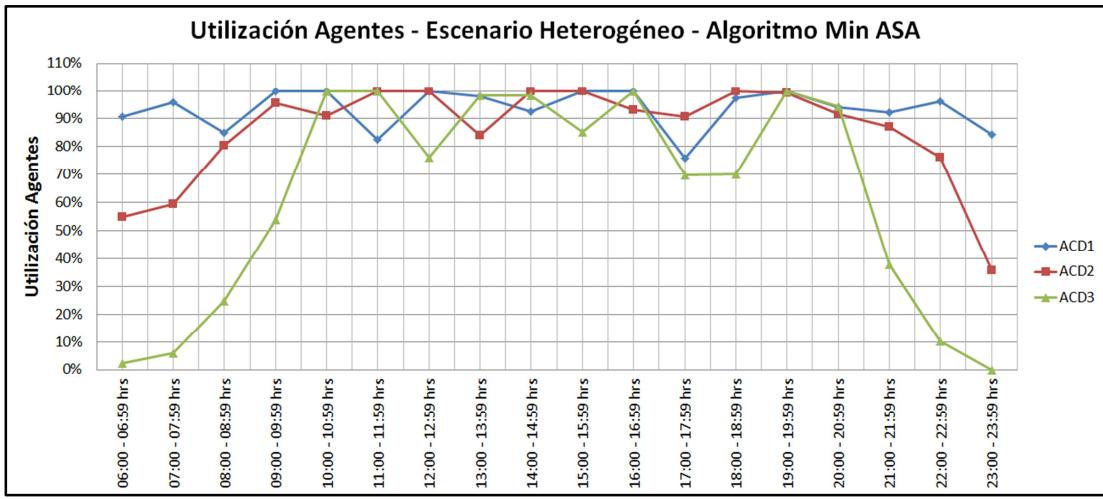


Figura 76. Utilización de los agentes empleando el algoritmo Min ASA bajo el escenario heterogéneo – Modelo de simulación

A-4 Clases desarrolladas para el balanceador de carga.

```
//Definición de la Clase AsteriskManagerIVR
class AsteriskManagerIVR : public CThread {
public:

    string DirIp; //Direccion IP del servidor Asterisk
    int Puerto; //Puerto del manager de Asterisk
    string UserManager; //Usuario del manager de Asterisk
    string PassManager; //Clave del manager de Asterisk
    int ivrNumber; //Numero de Asterisk IVR o Acceso
    ClientSocket AsteriskManagerWR; //Objeto tipo clase ClientSocket
    string DatosEvento[9]; //Variable utilizada para capturar los datos de un evento
    string DatosResponse[4]; //Variable utilizada para capturar los datos de una respuesta

    //Constructor
    AsteriskManagerIVR(string ip, int port, string user, string pass, int n) {
        DirIp = ip;
        Puerto = port;
        UserManager = user;
        PassManager = pass;
        ivrNumber = n;
    }

    //Destructor
    ~AsteriskManagerIVR() {
        ...
    };

//Definición de la Clase AsteriskManagerAgent
class AsteriskManagerAgent : public CThread {
public:

    string DirIp; //Direccion IP del servidor Asterisk
    int Puerto; //Puerto del manager de Asterisk
    string UserManager; //Usuario del manager de Asterisk
    string PassManager; //Clave del manager de Asterisk
    ClientSocket AsteriskManagerWR; //Objeto tipo clase ClientSocket
    string DatosEvento[9]; //Variable utilizada para capturar los datos de un evento
    string DatosResponse[2]; //Variable utilizada para capturar los datos de una respuesta

    //Constructor
    AsteriskManagerAgent(string ip, int port, string user, string pass) {
        DirIp = ip;
        Puerto = port;
        UserManager = user;
        PassManager = pass;
    }
```

```

//Destructor
~AsteriskManagerAgent() {
    ...
};

//Definición de la Clase LeerArchivoConfiguracion
class LeerArchivoConfiguracion : public CThread {
public:
    string archivolectura; //Nombre del archivo de configuración

//Constructor
LeerArchivoConfiguracion(string archivo)
{
    archivolectura = archivo;
}

//Destructor
~LeerArchivoConfiguracion() {
    ...
};

```

A-5 Configuración del plan de marcación de un servidor de acceso.

```
[globals]
;Prefijo de la Troncal SIP
TRUNK=SIP/t_gw1_to_acd

;Contexto utilizado para recibir las llamadas provenientes del Generador de Llamadas
[from_generador]
exten => 1410,1,NoOp(Llamada Inbound)
same => n,Set(CDR(userfield)=$(SIP_HEADER(X-Asterisk-UniqueID))) ;Se captura el UniqueID
proveniente del Generador de Llamadas y se guarda en el campo userfield
same => n,Set(_QueueID=$(SIP_HEADER(X-Asterisk-QueueID))) ;Se captura el QueueID de la
cola seleccionada por el usuario
same => n,Set(CDR(accountcode)=$QueueID) ;Se guarda el QueueID en el campo accountcode
same => n,Set(Context=bugarouter)
same => n,GotoIf($"{$QueueID}" = "1"?id1)
same => n,GotoIf($"{$QueueID}" = "2"?id2)
same => n,GotoIf($"{$QueueID}" = "3"?id3)
same => n,Log(ERROR, QUEUEID ${QueueID} NO ENCONTRADO)
same => n,Hangup()
same => n(id1),Set(Queue=cola1)
same => n,Goto(from_generador,1410,ue)
same => n(id2),Set(Queue=cola2)
same => n,Goto(from_generador,1410,ue)
same => n(id3),Set(Queue=cola3)
same => n,Goto(from_generador,1410,ue)
same => n(ue),NoOp(UserEvent Balanceador)
same => n,UserEvent(InboundCall,Channel: ${CHANNEL},Exten: ${EXTEN},Context:
${Context},Queue: ${Queue},UniqueID: ${UNIQUEID}) ;Evento que indica al balanceador de
carga que ingresó una llamada al sistema
same => n,Wait(5)
same => n,Hangup()

;Contexto utilizado para direccionar las llamadas al servidor de ACD seleccionado por el
balanceador
[bugarouter]
exten => *_X,1,NoOp(Llamada Direccionada al Servidor de Agentes ${EXTEN}:1})
same => n,SIPAddHeader(X-Asterisk-UniqueID:${UNIQUEID}); Se envía el Uniqueid de la
llamada a la máquina de ACD
same => n,Dial(${TRUNK}${EXTEN}:1/${QueueID},,hH)
same => n,NoOp(Hangupcause: ${HANGUPCAUSE})
same => n,Hangup()

exten => h,1,NoOp(Hangupcause: ${HANGUPCAUSE})
exten => h,n,ResetCDR(w) ;Se almacena el CDR actual antes de resetearlo
exten => h,n,NoCDR() ;Se utiliza para no almacenar el CDR 2 veces
exten => h,n,Hangup()
```

A-6 Resultados obtenidos del tamaño promedio de la cola bajo un escenario heterogéneo tanto en el modelo de simulación como en el prototipo del sistema.

- **Algoritmo GRR**

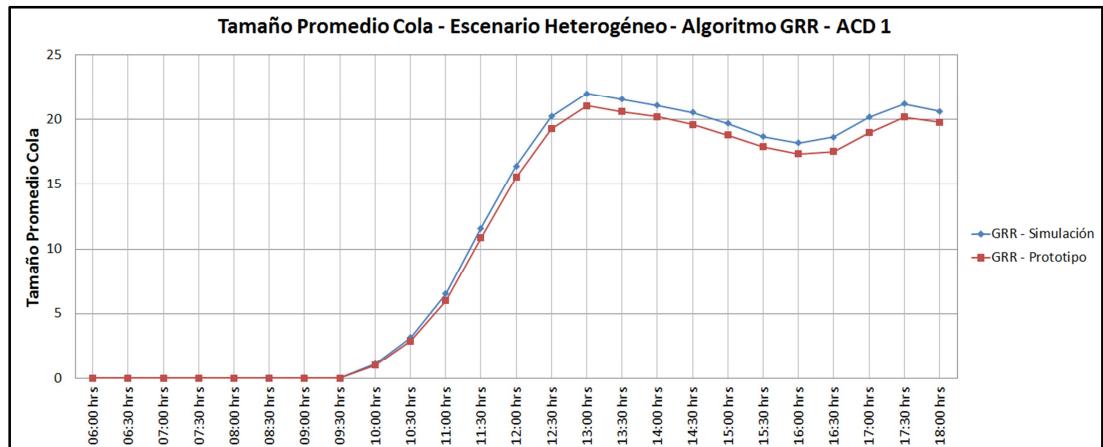


Figura 77. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo

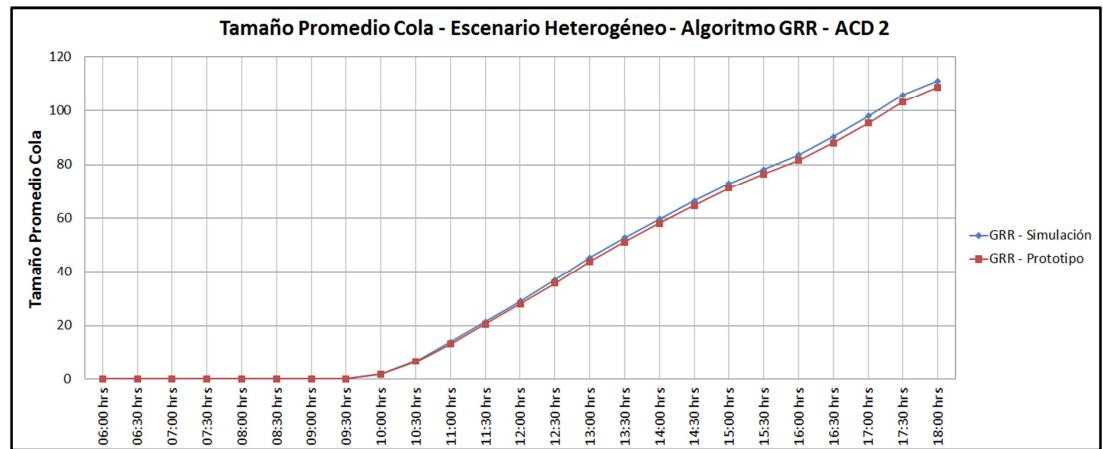


Figura 78. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo

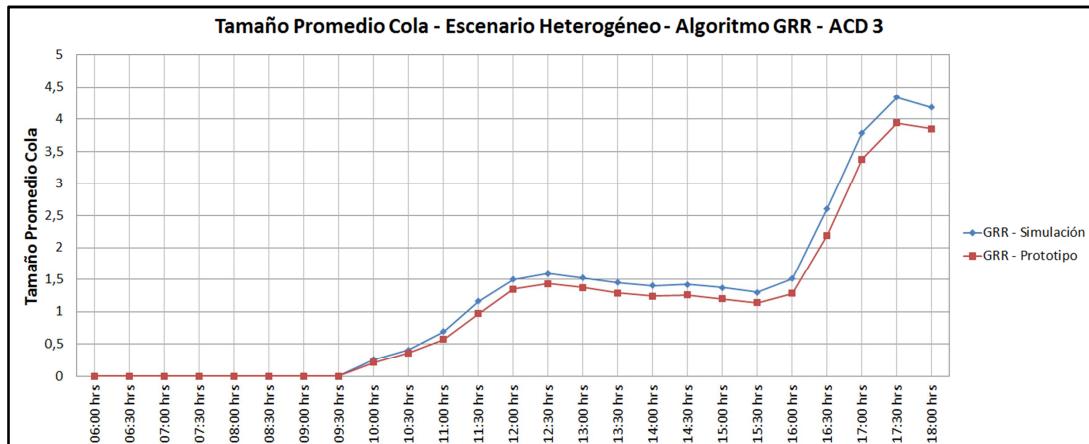


Figura 79. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo GRR bajo el escenario heterogéneo – Simulación vs. Prototipo

- **Algoritmo JSQ**

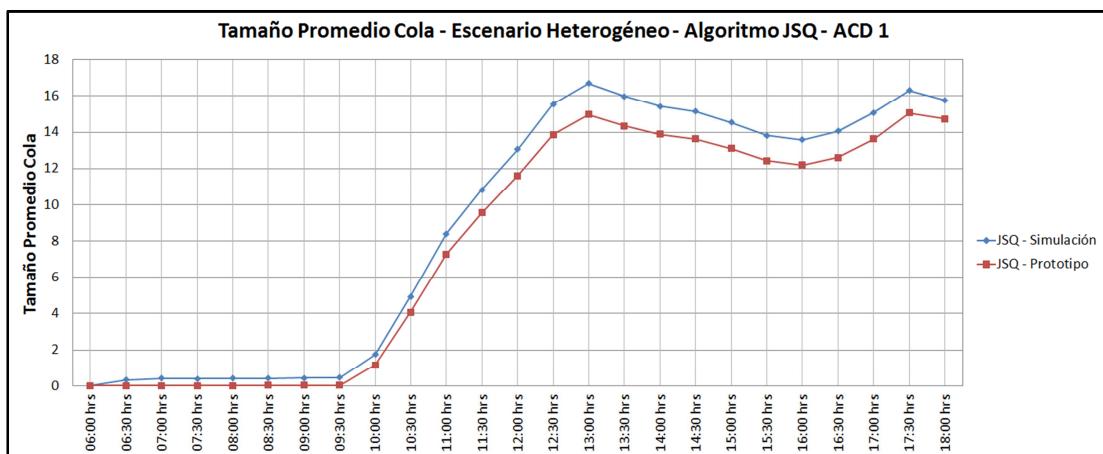


Figura 80. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo

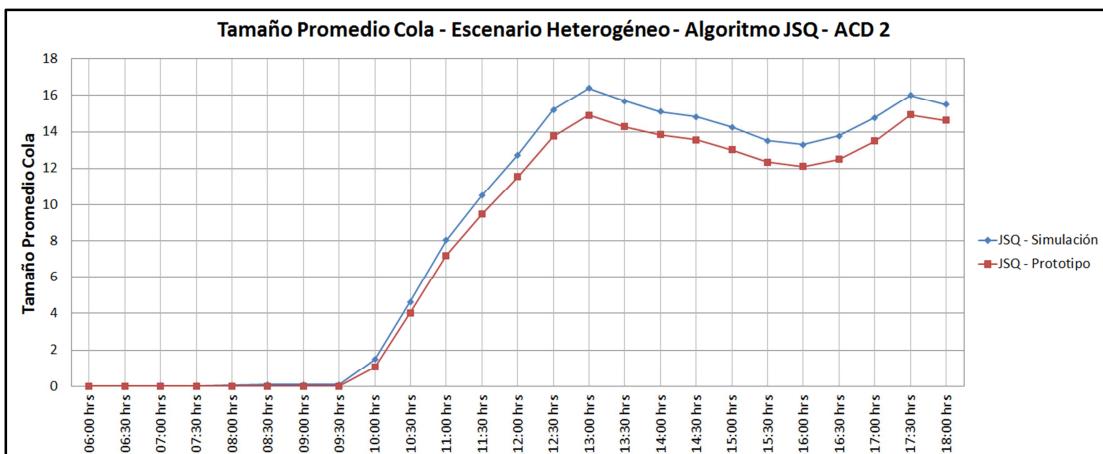


Figura 81. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo

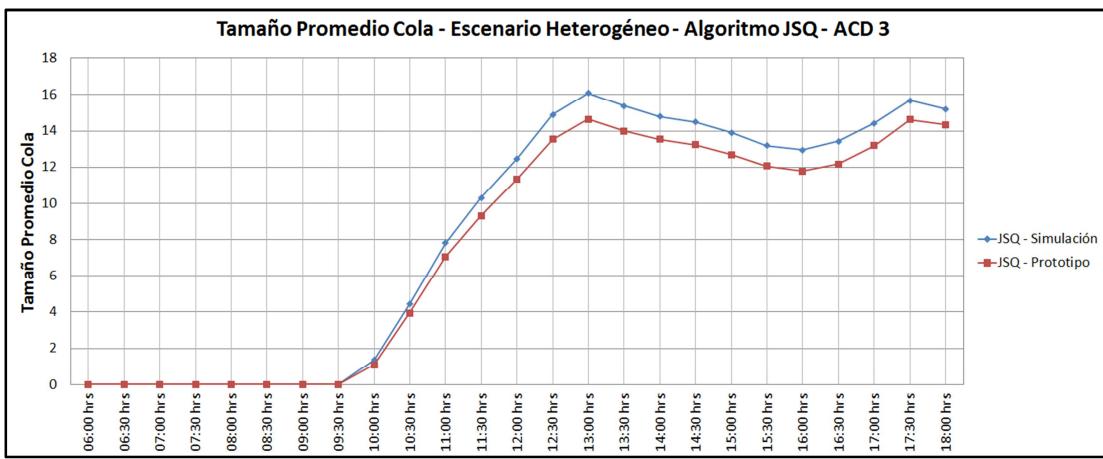


Figura 82. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo JSQ bajo el escenario heterogéneo – Simulación vs. Prototipo

- **Algoritmo MED**

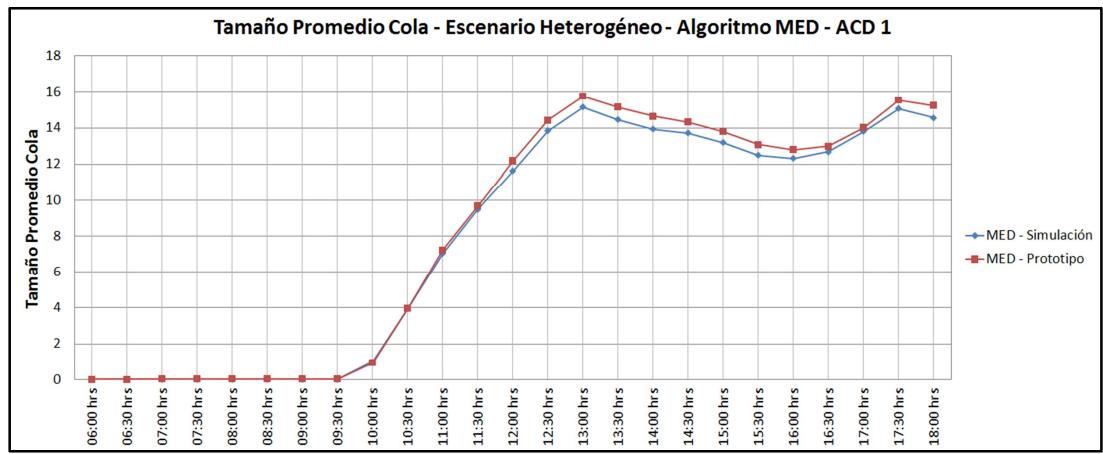


Figura 83. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo

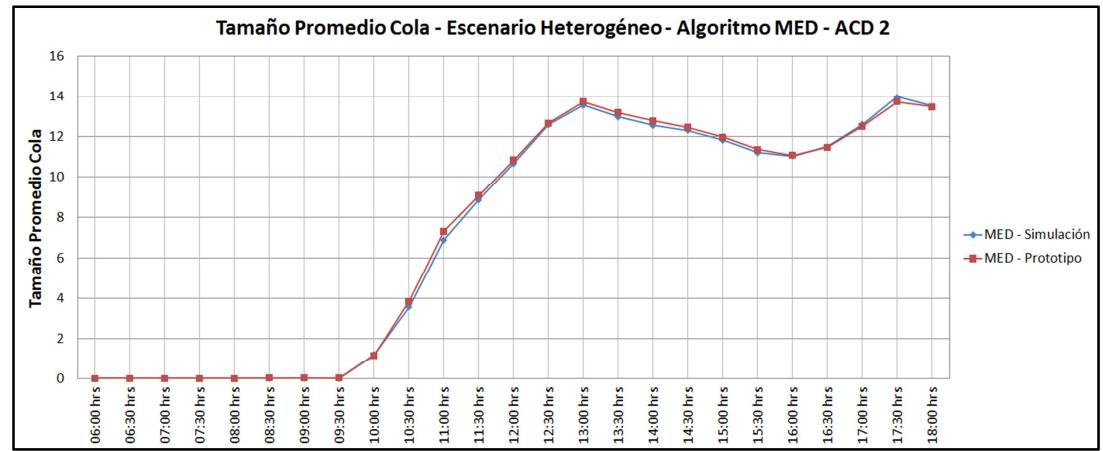


Figura 84. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo

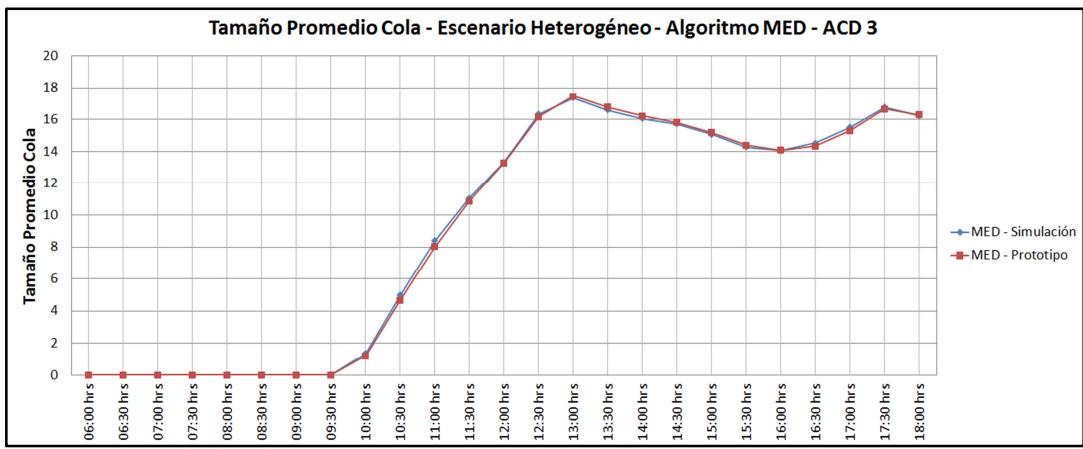


Figura 85. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo MED bajo el escenario heterogéneo – Simulación vs. Prototipo

- **Algoritmo Min ASA**

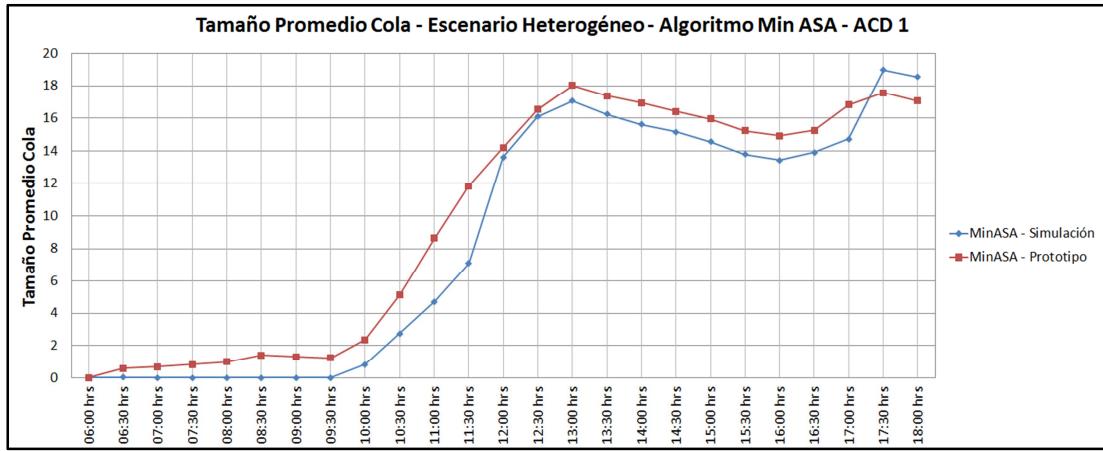


Figura 86. Tamaño promedio de la cola en el ACD1 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo

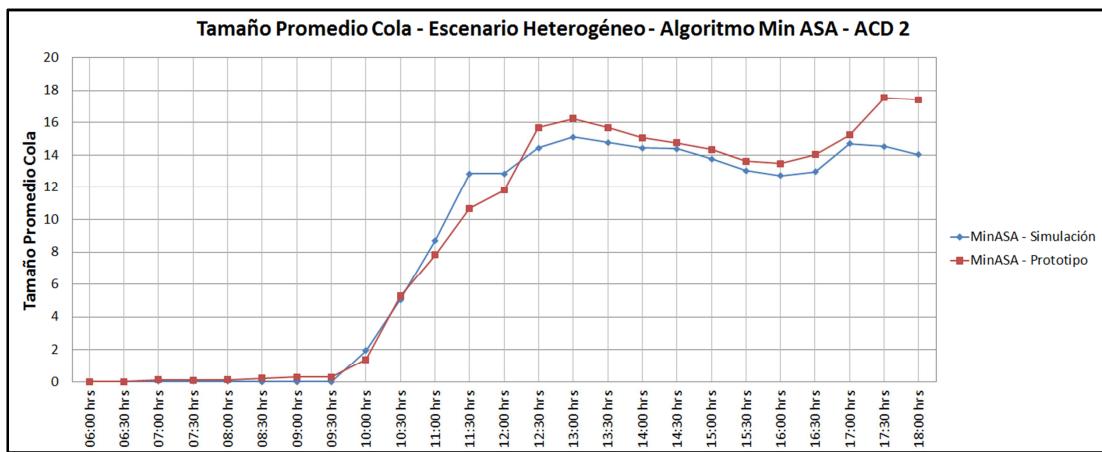


Figura 87. Tamaño promedio de la cola en el ACD2 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo

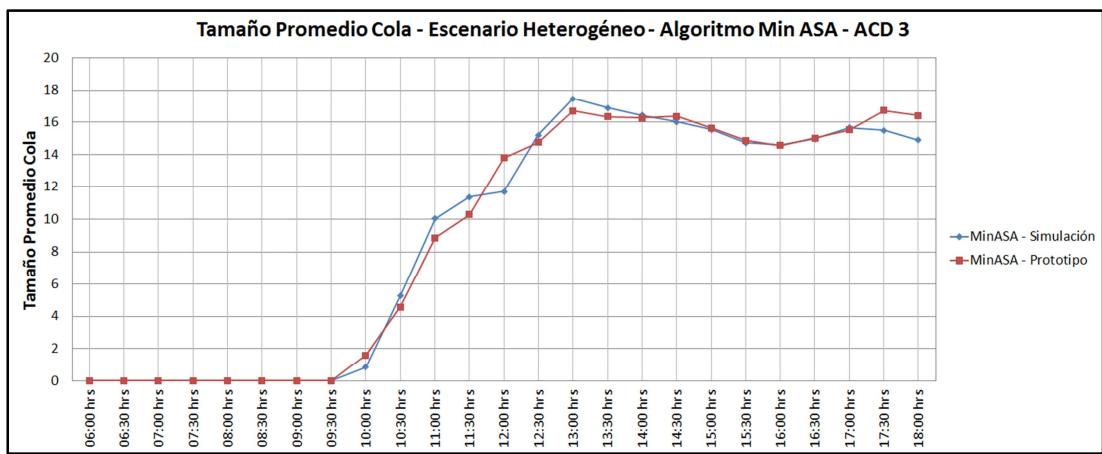


Figura 88. Tamaño promedio de la cola en el ACD3 utilizando el algoritmo Min ASA bajo el escenario heterogéneo – Simulación vs. Prototipo