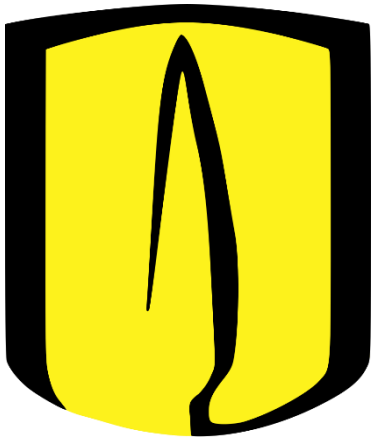
Informe Caso 3

TECNOLOGÍA E INFRAESTRUCTURA DE CÓMPUTO

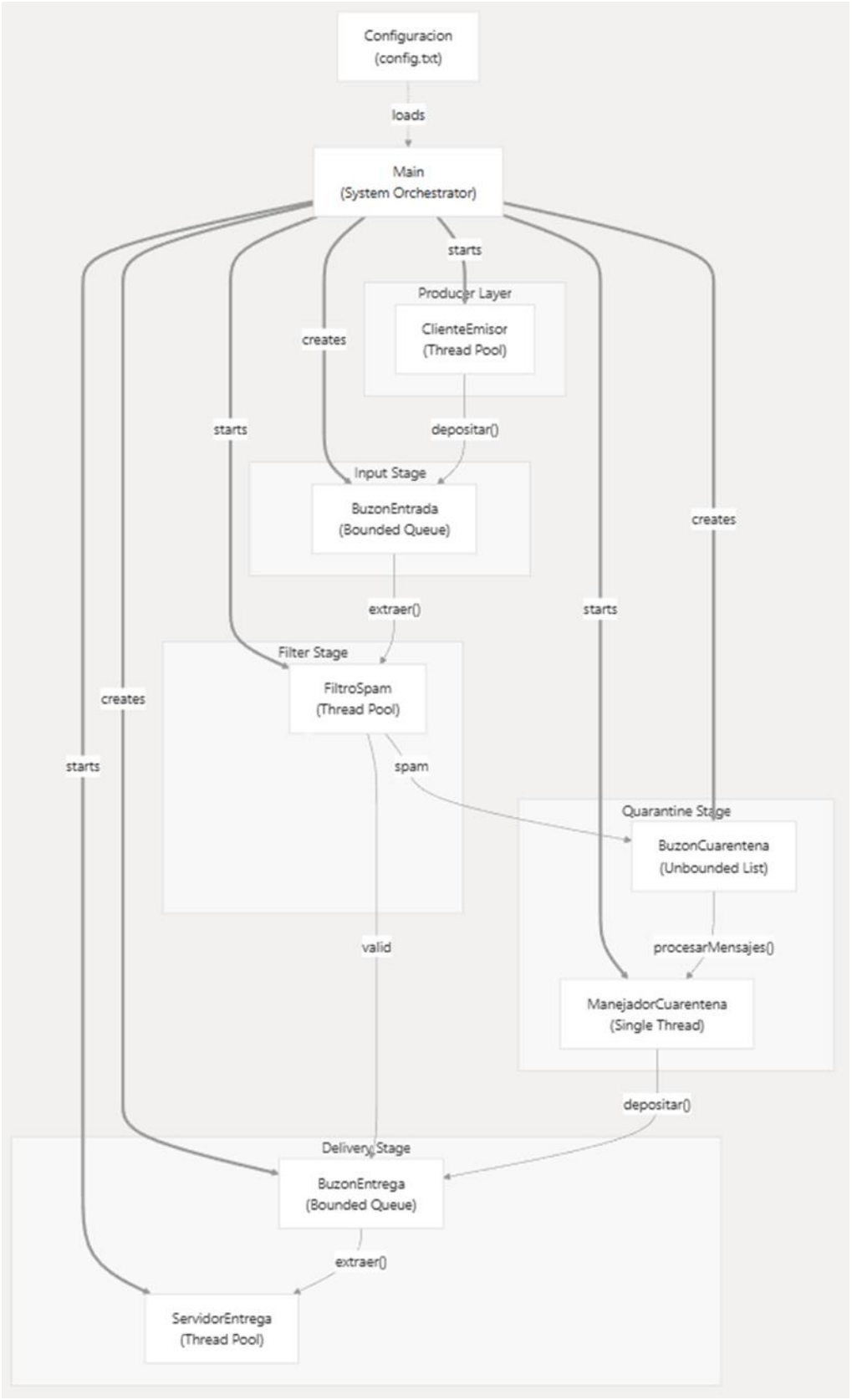


Juan David Uribe - 202322433
Daniel Vargas - 202123892
11-3-2025

Tabla de Contenidos

Funcionamiento del Sistema

El sistema se organiza como una cadena de procesamiento de mensajes a través de varios buzones sincronizados.



1. Carga de Configuración

El proceso inicia con la clase Configuración, la cual siempre lee los parámetros del archivo config.txt. Esta clase obtiene valores como número de clientes emisores, filtros y servidores, cantidad de mensajes por cliente, capacidades de los buzones de entrada y entrega.

2. Orquestación (Main)

La clase Main:

1. Crea los tres de buzones de comunicación: Entrada, Cuarentena y Entrega.
2. Crea e inicia los hilos correspondientes a cada capa (clientes, filtros, manejador y servidores).
3. Sincronizar la finalización de todos los hilos mediante join().

3. Capa Productora

Cada ClienteEmisor representa un thread productor que:

1. Envía un mensaje de tipo INICIO al buzón de entrada para marcar el comienzo de su sesión.
2. Genera y deposita un número configurado de correos (mensajes de tipo CORREO), de los cuales alrededor de la mitad son marcados aleatoriamente como spam.
3. Finalmente, envía un mensaje de tipo FIN indicando que ha completado su envío.

Los clientes operan sobre el BuzonEntrada, que es una cola con capacidad limitada y utiliza espera pasiva para sincronizar la producción.

4. Etapa de Entrada

El BuzonEntrada almacena temporalmente los mensajes de los clientes hasta que un FiltroSpam los extrae para su revisión.

Si la cola está llena, los clientes esperan (bloqueados con wait()) hasta que haya espacio disponible.
Si está vacía, los filtros esperan a que se deposite un nuevo mensaje.

5. Etapa de Filtrado

Cada FiltroSpam actúa como un hilo consumidor-productor intermedio que:

- Consume mensajes del buzón de entrada.
- Clasifica cada mensaje:
    - Si es spam, lo envía al BuzonCuarentena.
    - Si es válido, lo envía directamente al BuzonEntrega.
- Coordina el cierre con otros filtros mediante un candado estático de sincronización.

Además, hay una barrera que les permite saber cuándo todos los clientes han iniciado y finalizado, y cuándo es de cerrar el sistema.

El primer filtro que detecta que todos los clientes han terminado envía los mensajes de terminación (FIN) al manejador de cuarentena y a los servidores.

6. Etapa de Cuarentena

El BuzonCuarentena es una lista sin límite de capacidad, donde se almacenan los mensajes sospechosos.

Estos mensajes permanecen en cuarentena durante un tiempo aleatorio entre 10 y 20 segundos.

El ManejadorCuarentena, que funciona como un hilo único, despierta cada segundo y:

- Reduce el tiempo restante de cada mensaje.
- Descarta aquellos marcados como maliciosos (según probabilidad del 1/7).
- Transfiere los mensajes que completaron su tiempo de cuarentena al BuzonEntrega.

Cuando recibe un mensaje FIN y el buzón de cuarentena está vacío, este hilo finaliza su ejecución.

7. Etapa de Entrega

El BuzonEntrega es una cola de capacidad limitada que recibe mensajes tanto de los filtros (correos válidos) como del manejador de cuarentena (mensajes liberados). Este buzón cuenta con:

- Productores (filtros y manejador) que usan espera activa cuando la cola está llena.

- Consumidores (servidores) que usan espera semi-activa con wait(timeout) cuando la cola está vacía.

Cada ServidorEntrega es un hilo consumidor que extrae mensajes, los "procesa" (simula un retardo aleatorio) y finalmente muestra que fueron entregados.

Cuando recibe un mensaje FIN, el servidor termina su ejecución.

8. Cierre del Sistema

El flujo de cierre del sistema se realiza de la siguiente manera:

1. Los filtros detectan que todos los clientes han terminado.
2. Se espera a que los buzones de entrada y cuarentena queden vacíos.
3. Se envían mensajes FIN:
    - Uno por cada servidor (para terminar el procesamiento de entrega).
    - Uno al manejador de cuarentena (para cerrar su ciclo).
4. Se cierra el buzón de entrada, despertando cualquier hilo bloqueado.
5. Finalmente, Main espera la terminación de todos los hilos mediante join() y verifica que los buzones estén vacíos para imprimir su estado.

Sincronización entre Parejas de Objetos

ClienteEmisor — BuzonEntrada

Los clientes emisores depositan mensajes en el buzón de entrada utilizando sincronización con espera pasiva.

El método depositar() está sincronizado y emplea wait() y notifyAll().

Cuando el buzón alcanza su capacidad máxima, el cliente se bloquea con wait() hasta que un filtro extraiga un mensaje.

Al liberar espacio, el buzón invoca notifyAll() para despertar a los clientes bloqueados y permitir que continúen depositando mensajes.

FiltroSpam — BuzonEntrada

Los filtros extraen mensajes del buzón de entrada de forma concurrente.

El método extraer() está sincronizado y también usa espera pasiva.

Si el buzón está vacío, el filtro se bloquea con wait() hasta que un cliente deposite un nuevo mensaje.

Una vez extraído un mensaje, se llama a notifyAll() para reactivar posibles clientes que estaban esperando espacio libre.

FiltroSpam — BuzonCuarentena

Los filtros depositan mensajes marcados como spam en el buzón de cuarentena.

El método depositar() está sincronizado, pero no contiene espera, ya que el buzón tiene capacidad ilimitada.

El mensaje se agrega directamente a la lista interna y se llama a notifyAll() para despertar posibles hilos en espera (como el manejador de cuarentena).

FiltroSpam — BuzonEntrega

Los filtros depositan mensajes válidos en el buzón de entrega utilizando espera semi-activa.

El método depositar() permanece sincronizado y, cuando el buzón está lleno, el hilo realiza una espera activa controlada (por ejemplo, mediante ciclos o cediendo la CPU).

Esto reduce la latencia de entrega a costa de un mayor uso de CPU, tal como lo exige el caso.

ManejadorCuarentena — BuzonCuarentena

El manejador de cuarentena procesa periódicamente los mensajes en cuarentena, con un ciclo de aproximadamente un segundo.

Invoca el método procesarMensajes(deltaTiempo), el cual es sincronizado y garantiza acceso exclusivo durante el procesamiento.

Este método disminuye el tiempo de cuarentena de cada mensaje, descarta los considerados maliciosos (según un número aleatorio múltiplo de 7) y devuelve los mensajes listos para ser enviados al buzón de entrega.

ManejadorCuarentena — BuzonEntrega

Los mensajes que completan su tiempo de cuarentena son depositados en el buzón de entrega por el manejador.

El mecanismo es el mismo que emplean los filtros de spam: se utiliza una espera semi-activa en caso de que el buzón esté lleno, manteniendo el control del flujo de mensajes.

ServidorEntrega — BuzonEntrega

Los servidores de entrega extraen mensajes del buzón de entrega para procesarlos.

El método extraer() utiliza espera semi-activa mediante wait(timeout).

Si el buzón está vacío, el servidor espera unos milisegundos antes de reintentar, evitando bloqueos indefinidos y permitiendo una salida limpia.

Después de extraer un mensaje, se llama a notifyAll() para despertar a los productores que pudieran estar esperando espacio libre.

Escenarios de prueba

#	**Prueba**	Clientes	Msgs/Cliente	Filtros	Serv.	Cap. Ent.	Cap. Entrega	Total Msgs	Tiempo (seg)	Rendimiento	Observaciones
1	Configuración Balanceada Estándar	3	5	2	2	5	10	15	13.23	1.13 msg/s	Configuración equilibrada, ejecución fluida sin problema
2	Carga Mínima con Pocos Recursos	1	2	1	1	5	10	2	17.19	0.12 msg/s	Algo lenta debido a la inicialización usada solo para unos pocos mensajes
3	Muchos Clientes con Pocos Recursos	6	4	1	1	3	3	24	13.23	1.81 msg/s	Cuello de botella intencional, un solo filtro y servidor manejan bien la carga
4	Muchos Recursos con Carga Media	5	5	3	3	10	15	25	20.26	1.23 msg/s	Múltiples threads causan overhead de sincronización

5	Buffers Pequeños	3	4	2	2	**1**	**1**	12	15.25	0.79 msg/s	Buffers unitarios fuerzan sincronización constante, muchos bloqueos
6	Carga Máxima con Muchos Recursos	10	10	5	5	20	30	100	18.25	5.48 msg/s	Mejor rendimiento, demuestra escalabilidad con alto volumen

El sistema evidencia que, con pocas cargas, el exceso de threads introduce más costo que eficiencia. Por ejemplo, en la Prueba 4, aumentar a tres filtros y tres servidores incrementó el tiempo total a más de 20 segundos.

En contraste, la Prueba 6 demostró que el sistema escala correctamente bajo carga real: con 100 mensajes alcanzó su mejor rendimiento, procesando más de cinco mensajes por segundo. Esto confirma que el diseño concurrente se aprovecha plenamente cuando hay suficiente trabajo para distribuir.

Por otro lado, la Prueba 5 evidenció el impacto del tamaño de los buffers. Usar buzones unitarios ralentizó el sistema un 50 %, ya que los bloqueos entre productores y consumidores aumentan cuando hay poca capacidad de almacenamiento intermedio.

Finalmente, las pruebas también mostraron que el sistema tiene un overhead fijo importante (alrededor de 13–15 s) asociado al inicio y finalización de hilos. Con cargas muy pequeñas, este tiempo domina el desempeño, como se vio en la Prueba 2. En cambio, con configuraciones balanceadas, el sistema mantiene una eficiencia razonable incluso con cuellos de botella controlados (Prueba 3)