**Informe caso 3**

1. Cambios realizados con respecto al caso 2

Cambios en el cliente:

En primer lugar se corrigieron ciertos detalles del caso anterior en el método recibirCertificado() (líneas 146 a 164). Del mismo modo se agregó un nuevo método que almacena en un documento de texto los tiempos de ejecución de la recepción de certificados y de las llaves. Este método es reportarTiempos() (líneas 237 a 251). Esto también se hizo con el uso de dos nuevos atributos tiempoExe y tiempoKey (líneas 74 y 79 respectivamente).

No se realizaron cambios en el servidor.

1. Identificación de la plataforma

* Arquitectura: 64 bits
* Número de núcleos: 2 cores
* Velocidad del procesador: 2.5 GHz
* Tamaño de la memoria RAM: 1024MB (1GB)
* Espacio del disco duro: 40GB

1. Conclusiones sobre el comportamiento del sistema

* Con respecto a las gráficas obtenidas del caso se puede evidenciar que el tiempo de actualización es mucho más alto en el al haber un mayor número de threads. Esto es debido a que el para manejar 8 threads la maquina en la que se corrió el programa tiene muy poca eficacia puesto que tiene que estar cambiando de hilo y se pierda tiempo en el proceso. Por otra parte, al manejar dos hilos se obtuvo en las tres cargas diferentes el menor tiempo ya que optimiza el número de cores con el que corre la máquina.
* En cuanto al número de transacciones perdidas, hubo una mayor denotación en las gráficas que involucraron una mayor cantidad de cargas transaccionales. Este suceso se debe a que si hay una sobrecarga de transacciones y no hay suficientes hilos para atenderlas habrá un gran número de fallas al realizar el proceso.
* Por último, el según el comportamiento de las gráficas al haber más threads. Esto es debido a que cada hilo procesa una tarea y para cada tarea se le debe asignar un espacio en memoria. Por ende, si hay más tareas, se requieren más hilos causando un mayor consumo de memoria.

1. Comportamiento de la aplicación ante diferentes niveles de seguridad

* El resultado esperado de una aplicación que no implemente funciones de seguridad es disminución de tiempos de actualización, ya que debe implementar más métodos de seguridad que consumen más tiempo. En cuanto a transacciones fallidas disminuyen, pero no una cifra significativa, ya que ahora las peticiones no tienen que ser protegidas, pues muchas veces hay fallos en los métodos de seguridad que ahora no hay que implementar. Por último, el consumo de CPU se mantendrá igual, pues el que se encarga de seguridad es el programa, el gasto del CPU se debe más a la carga puesta en él.
* Según los resultados obtenidos con los dos niveles de seguridad, se puede evidenciar que en efecto los tiempos de actualización son menores, pues ya no consume el tiempo de las funciones de seguridad. En cuanto a las transacciones fallidas también disminuyeron, pero no significativamente según el comportamiento de las gráficas. Y por parte del gasto del procesador, disminuyó en una menor proporción comparativamente con los otros factores.