Calidad de Servicio en Entornos Virtuales Distribuidos

P. Moreno, P. Morillo, J. M. Orduña
Dept. de Informática
Universidad de Valencia
Juan.Orduna@uv.es

J. Duato
DISCA
Universidad Politécnica de Valencia
jduato@gap.upv.es

Resumen

Una de las cuestiones clave en el diseño de los sistemas DVE es el problema del particionado, que consiste en asignar eficientemente los clientes (avatares 3-D) a los servidores del sistema. Algunos métodos propuestos permiten aumentar de forma significativa la productividad del sistema. Sin embargo, estas propuestas no contemplan el cumplimiento de ningún requisito temporal.

En el presente trabajo, demostramos que el problema de la calidad de servicio en sistemas DVE se puede abordar desde el método de particionado. Además proponemos un método de particionado que no sólo proporciona una alta productividad al sistema, sino que también satisface (si es posible) cualquier límite de latencia que los avatares puedan requerir. Los resultados de la evaluación demuestran que este método permite aumentar enormemente el número de avatares a los que se les proporciona calidad de servicio a la vez que consigue la mayor productividad posible.

1. Introducción

Los sistemas DVE han experimentado un crecimiento espectacular en los últimos años. Estos sistemas permiten a múltiples usuarios, trabajando en diversos equipos informáticos, interconectarse a través de distintas redes (incluida Internet) para trabajar juntos en un mundo virtual compartido. Una entidad denominada *avatar*, cuyo estado es controlado por el usuario, representa a cada usuario en el sistema DVE. Los sistemas DVE se usan actualmente en un numerosas aplicaciones, tales como el el entrenamiento distribuido (civil o militar), el e-learning o los videojuegos multijugador.

Las arquitecturas basadas en los servidores en red se han convertido en un estándar "defacto"para los sistemas DVE [1, 2]. En estas arquitecturas, las computadoras cliente se conectan exclusivamente a uno de los servidores del sistema. Cuando un cliente mueve un avatar, también envía un mensaje de actualización a su servidor, que asimismo debe propagar este mensaje a los otros servidores y clientes. Se han propuesto conceptos como las áreas de influencia (AOI) [1] para limitar el número de mensajes de actualización. Estos conceptos definen un área de vecindad para los avatares, de manera que un cliente dado c que controla a un avatar dado i debe notificar todos los movimientos de i (enviando un mensaje de actualización) solamente a los clientes que controlan a avatares situados en la vecindad del avatar i. Los avatares en el AOI del avatar i se denominan *avatares vecinos* del avatar i.

El *problema del particionado* [2] se ha demostrado como el punto clave en el diseño de los sistemas DVE basados en servidores en red. Este problema consiste en distribuir eficientemente la carga de trabajo entre los diversos servidores en el sistema (asignando avatares a los servidores). En un trabajo anterior, propusimos un método de particionado dinámico para solucionar esta cuestión [3]. Este método proporciona un aumento significativo de la productividad de los sistemas DVE. Sin embargo, las medidas de prestaciones más importantes de estos sistemas (como en cualquier

sistema cliente-servidor) son no solamente la productividad sino también la latencia. La latencia se puede definir como el intervalo de tiempo desde el instante en que cualquier vecino de un avatar dado i hace un movimiento hasta el instante en que se notifica al avatar i ese movimiento. La latencia es un indicador de la *Calidad de Servicio (QoS)* proporcionada a los usuarios por el sistema, puesto que determina la rapidez con la que se notifica a los clientes cualquier cambio en el mundo virtual. En este trabajo, demostramos que el problema de proporcionar QoS a los avatares en un sistema DVE se puede tratar por medio del método de particionado. También, proponemos un método de particionado basado en una técnica meta-heurística que busque la mejor compensación entre la latencia del sistema, la productividad del sistema y la eficiencia del particionado. Los resultados de evaluación demuestran que este método puede maximizar la productividad del sistema mientras que incrementa también el número de avatares a los que se les proporciona QoS.

El resto del trabajo se organiza de la siguiente manera: La sección 2 detalla los aspectos que se deben solucionar para conseguir QoS en sistemas DVE. La sección 3 describe el método heurístico de búsqueda usado para proporcionar QoS en sistemas DVE. Tras ello, la sección 4 presenta la evaluación del funcionamiento del método propuesto. Finalmente, la sección 5 presenta las conclusiones y trabajo futuro.

2. Calidad de Servicio en Sistemas Virtuales distribuidos

El problema de proporcionar QoS a los clientes en un sistema de DVE se ha descrito ya, y se han propuesto algunas estrategias para solucionarlo ([4],[5]). Una de éstas [5] utiliza métodos que compensan la latencia para reparar los efectos del jitter. Otro método [4] consiste en modificar la resolución de los modelos 3-D dependiendo de la velocidad de conexión del cliente. Ambas estrategias intentan proporcionar QoS a los avatares en detrimento de la calidad de los gráficos. Sin embargo, ninguna de ellas considera el comportamiento no lineal de los sistemas DVE con la carga de trabajo asignada a cada servidor, según lo descrito en [6]. Por lo tanto, no pueden garantizar que el sistema no se saturará, degradándose en gran medida la latencia. Un trabajo reciente demuestra que si el retardo de ida y vuelta (round-trip delay) de los mensajes de actualización de los avatares no es mayor de 250 milisegundos, los usuarios perciben que el sistema responde rápidamente [5]. Hemos asumido este valor de umbral como la condición para proporcionar QoS a un avatar. Sin embargo, puede ser modificado si es necesario, dependiendo de los requisitos de la aplicación.

El retardo de ida y vuelta para los mensajes enviados por un avatar dado está muy relacionado con el problema del particionado. Si se asigna un avatar dado i y todos sus vecinos al mismo servidor, el tiempo requerido para informar al avatar i de los movimientos de cualquier avatar vecino será más pequeño que si alguno de los vecinos se asigna a un servidor distinto. Hemos evaluado diversos sistemas DVE mediante simulación (según lo descrito en section 4), para medir el efecto que puede tener el asignar avatares vecinos a diversos servidores en la QoS proporcionada a un avatar dado. Un ejemplo significativo de esta evaluación se demuestra en la figura 1. Esta figura muestra en el eje de ordenadas el retardo medio de ida y vuelta obtenido para los mensajes enviados por un avatar dado i. El eje de abcisas muestra la cantidad de avatares vecinos de i asignados al mismo servidor que i. Cada punto en el diagrama representa el valor medio de las latencias medias obtenidas después de 30 simulaciones diferentes. La desviación estándar de las diversas simulaciones no estaba por encima de 25 ms en ningún caso. El factor de presencia $f_p(i)$ (el número de avatares en cuya AOI aparece el avatar i) en estas simulaciones era igual a 60 para todos ellos.

La figura 1 muestra que el retardo de ida y vuelta para los mensajes enviados por el avatar i aumenta linealmente cuantos más avatares vecinos de i se asignan a servidores diferentes. En concreto, podemos ver que el sistema pro-

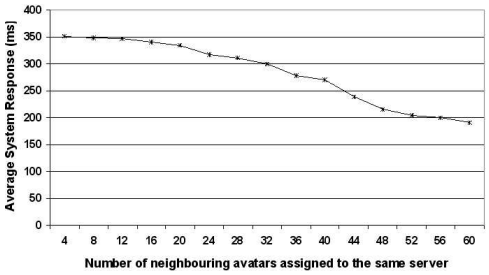

Figura 1: Effect of assigning avatars to different servers on message latencies.

porciona QoS (retardo de ida y vuelta medio por debajo de 250ms.) al avatar i si 44 o más avatares vecinos del avatar i (de un total de 60 avatares vecinos) se asignan al mismo servidor que i. Repetimos el mismo experimento para varias configuraciones distintas de realidad virtual y obtuvimos resultados muy similares. El retardo de ida y vuelta medio para los mensajes enviados por un avatar dado aumenta linealmente con el número de los avatares vecinos asignados a un servidor diferente. Por lo tanto, una manera factible de proporcionar QoS a los avatares es agrupar avatares vecinos tanto como sea posible en el mismo servidor. Sin embargo, agrupar avatares en un servidor puede desequilibrar la carga. Ello a su vez puede degradar el funcionamiento del sistema [6, 3], haciendo que no solamente no se proporcione QoS a algunos avatares, sino disminuyendo de forma importante la productividad y aumentando considerablemente la latencia para todos los avatares. El método de particionado debe garantizar que el porcentaje de la utilización de la CPU en todos los servidores está por debajo del 100 %, sin importar cualquier otra consideración. Además, el proporcionar QoS a los avatares puede tener efecto en el número de avatares migrados de uno a otro servidor en cada ejecución del algoritmo de particionado, ya que puede ser necesario migrar gran cantidad de avatares. En este sentido, un trabajo reciente demuestra que ningún método de particionado eficiente debe migrar más del 30 % de avatares en el Sistema [7]. Por tanto, estos tres aspectos deben ser considerados para proporcionar QoS al solucionar el problema de la particionado.

3. Método de Particionado

Para proporcionar QoS, pueden aplicarse distintas técnicas meta-heurísticas. En un trabajo anterior, presentamos un estudio comparativo entre dos técnicas diferentes: una técnica meta-heurística evolutiva, llamada *Simulated Annealing (SA)*, y una técnica meta-heurística constructiva, llamada*Greedy Randomized Adaptive Search (Búsqueda aleatoria adaptativa) (GRASP)* [8]. Puesto que hemos obtenido un funcionamiento mejor en la solución de este problema particular con la técnica del GRASP, proponemos ésta como método de particionado para proporcionar QoS en los sistemas DVE.

3.1. Función de Calidad

Cada partición (asignación de clientes a los servidores) del sistema se asocia a una función objetivo F, denominada *función de calidad*, que asigna un coste a cada solución particular. La técnica meta-heurística debe encontrar un reparto cercano al óptimo con el valor más bajo de F que sea posible. La función objetivo F debe considerar los tres factores descritos anteriormente para proporcionar QoS a los avatares: el porcentaje de utilización de la CPU en todos los servidores, el número de los avatares que migran y el número de los avatares a los que se proporciona QoS. Hemos definido la función de calidad F como

$$F(P) = \Psi(P) + \Phi(P) + \Gamma(P) \quad (1)$$

donde el término $\Psi(P)$ evalúa el porcentaje estimado de utilización de la CPU en todos los servidores para la partición P. La valoración de este término se basa en el método de caracterización propuesto en [6]. El término $\Phi(P)$ es inversamente proporcional al número estimado de avatares en la partición P cuyos mensajes presenten un retardo medio de ida y vuelta inferior a 250 ms. Este término se estima teniendo en cuenta que el retardo de ida y vuelta

para los mensajes enviados por cada avatar depende del número de los avatares vecinos asignados al mismo servidor. Finalmente, el término $\Gamma(P)$ mide el número de avatares migrados a otro servidor por la partición P. Para calcular este término se utilizan la asignación actual y la asignación definida por la partición P.

El comportamiento de $\Phi(P)$ es inversamente proporcional al número de avatares cuyos mensajes muestran un retardo de ida y vuelta medio inferior a 250 ms. Este término evalúa el retardo de ida y vuelta para los mensajes enviados por cada avatar. El retardo de ida y vuelta medio actual para cada avatar se mide para la partición actual, y de estos valores se estima el retardo de ida y vuelta medio que cada avatar tendrá en la partición P, considerando que está en relación lineal con el número de los avatares vecinos asignados al mismo servidor. De ese modo, la función $h_{asr}(i)$ asigna un valor a la partición P para cada avatar, dependiendo del valor estimado.

$\Gamma(P)$ es también una función de evaluación compuesta por dos secciones. Para ser eficiente, el método de particionado no debe migrar más de un tercio de los avatares [7]. Por lo tanto, el comportamiento de la función de evaluación $\Gamma(Pp)$ es el que se muestra en la figura 2. $\Gamma(P)$ tiene una sección lineal desde cero hasta un tercio de los avatares existentes. De ese punto hacia arriba, $\Gamma(P)$ tiene un comportamiento parabólico. Así, cuantos más avatares migran en la partición P, mayor es la probabilidad de que P sea desechada.

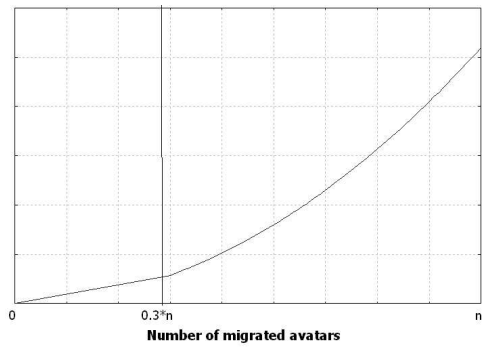

Figura 2: Función Γ.

3.2. Método heurístico

Proponemos la búsqueda GRASP como el método de particionado a emplear para proporcionar QoS a los avatares. La búsqueda GRASP ya se ha evaluado y se ha comparado con la otra técnica meta-heurística aplicada a este problema específico [8], y ha proporcionado los mejores resultados. La técnica GRASP comienza con una partición inicial. Esta partición inicial se proporciona mediante la técnica de balanceo de carga propuesta en [3]. Así, nos aseguramos de que la partición inicial está bien-equilibrada y todos los servidores tengan un porcentaje de la utilización de la CPU tan bajo como sea posible. Sin embargo, algunos avatares pueden no recibir QoS en esta partición. Llegados a este punto, se utilizará el método GRASP para buscar una partición pseudo-óptima que proporcione QoS al mayor número de avatares posible al mismo tiempo que migre el mínimo número de avatares. El primer paso en la puesta en práctica del GRASP consiste en clasificar (en orden descendente) por su factor de presencia f_p los avatares que cuyos mensajes muestran un retardo de ida y vuelta mayor de 250 ms. (esos avatares no obtienen QoS). La idea es proporcionar QoS a los avatares que requieran menos esfuerzos del sistema. Los avatares con el factor más alto de presencia deben intercambiar mensajes de actualización con muchos avatares. Por lo tanto, la migración de estos avatares al servidor apropiado puede disminuir el retardo de ida y vuelta de los mensajes para el número máximo de avatares posible (puede proporcionar QoS a muchos avatares con el menor esfuerzo). Por otra parte, si el método GRASP se centra solamente en esta clase de avatares, puede disminuir significativamente la función de Φ sin un aumento significativo de la función de Γ. Los primeros elementos c en la clasificación de avatares (de una población de avatares de n) se definen como *avatares críticos*. El método GRASP considera como no asignados a los avatares críticos, e intentará asignarlos a un servidor para que se les proporcione QoS. El resto de los n avatares (definidos como los e *avatares fáciles*, donde $n = c+e$) no serán reasignados, permaneciendo asignados al

mismo servidor al que lo fueron inicialmente. Cada iteración del método GRASP asigna uno de los avatares críticos. El número de iteraciones (el número de avatares reasignados) es el único parámetro que se fija desde el exterior en el método GRASP.

Cada iteración del método GRASP consiste en dos pasos: construcción y búsqueda local. La fase de *construcción* construye una solución factible eligiendo un avatar crítico por cada iteración, y la de*búsqueda local* deriva esta solución temporal siguiendo un criterio de vecindad. La puesta en práctica detallada de la búsqueda GRASP en cada iteración se muestra en [8]. No hemos incluido esta descripción aquí debido a limitaciones de espacio. Cuando el método GRASP acaba, su resultado es una partición con un valor de la función de calidad F lo más óptimo posible.

4. Evaluación de prestaciones

Proponemos la evaluación de los sistemas genéricos DVE a través de la simulación. La metodología de evaluación usada se basa en los estándares principales para modelar entornos virtuales distribuidos, tales como FIPA [9], DIS [10] y HLA [11]. Hemos desarrollado una herramienta de simulación que modela el comportamiento de un sistema genérico DVE integrado por varios servidores interconectados, y hemos realizado estudios experimentales para evaluar el funcionamiento de la técnica propuesta. En cada simulación, los avatares que comparten la misma AOI intercambian mensajes para notificar su posición en el mundo virtual 3D. La estructura del mensaje es la *Avatar Data Unit (ADU)* especificada por DIS [10]. Una simulación consiste en que cada avatar realiza 40 movimientos, con una frecuencia de un movimiento cada 2 segundos. Cada vez que un avatar i realiza un movimiento, el cliente que controla i envía un mensaje con una marca temporal a los clientes que controlan los avatares vecinos de i. Estos devuelven un mensaje ACK al cliente que controla i. Cuando la simulación acaba, cada computador puede determinar el retardo de ida y vuelta medio para todos los mensajes enviados durante la simulación. Definimos este valor medio para un avatar dado (computador cliente) i como asr_i (*respuesta media del sistema* para el avatar i). Se evita el sesgado del reloj, ya que es el mismo cliente que controla i quien añade las marcas temporales tanto inicial como de reconocimiento ACK. Para poder comparar hemos simulado el método de particionado propuesto en la sección anterior (GRASP), y el método de particionado ALB propuesto en [3] (el método de particionado que proporciona la partición inicial para el método GRASP). Hemos simulado sistemas DVE con tres diferentes patrones de movimiento de avatares: Patrón de cambio circular (Circular Changing Pattern) [12], Hot-Points-ALL (HPA) [13] y también Hot-Point-Near(HPN) [14]. CCP considera que todos los avatares en el mundo virtual se mueven circularmente, comenzando y terminando en la misma localización. HPA considera que existen ciertos "puntos calientes"donde todos los avatares se acercan más pronto o más tarde. Finalmente, HPN también considera estos puntos calientes, pero solamente los avatares situados dentro de un radio dado se acercan a estos puntos. Una iteración en un patrón de movimiento dado consiste en que todos los avatares del sistema realicen un solo movimiento. Además, se han considerado distintas distribuciones iniciales de avatares en el mundo virtual (uniforme, sesgadas y arracimadas), como en otros estudios [2, 12]. En la figura Figura 3 se muestra la distribución final de los avatares en un mundo virtual 2-D si estos patrones del movimiento se aplicaran a una partición inicial uniforme de avatares. En esta figura, el mundo virtual es un cuadrado y un punto gris representa a cada avatar.

El método ALB y el método GRASP se han ejecutado cada vez que una iteración acaba y la utilización de la CPU en cualquier servidor alcanza el 95 %. Además, se guarda la cantidad de avatares proporcionados QoS en esa iteración. El método GRASP se ejecuta después de cada iteración si este parámetro disminuye por debajo del valor guardado, para aumentar la QoS de nuevo. Debido a limitaciones de espacio, en el presente traba-

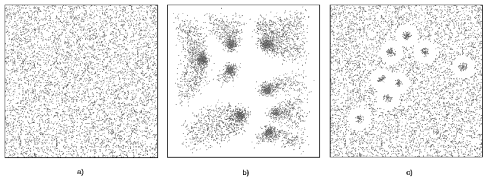

Figura 3: Distribución fianl de avatares para patrones de movimiento a) CCP, b) HPA, and c) HPN aplicados a una distribución inicial uniforme.

jo mostraremos solamente los resultados para las combinaciones más representativas de patrones de movimiento y distribuciones iniciales de avatares. El resto de las 9 combinaciones posibles de patrones del movimiento y distribuciones iniciales de avatares proporcionó resultados muy similares a los mostrados aquí para cada configuración de DVE. Hemos probado una gran cantidad de configuraciones diferentes de DVE. No obstante, para mayor brevedad, presentamos en este trabajo dos experimentos representativos, denominados MEDIUM1 y MEDIUM2. MEDIUM1, como representativo de los mundos virtuales pequeños, se compone de 250 avatares y de 3 servidores, y MEDIUM2, como representativo de los grandes se compone de 700 avatares y de 10 servidores. Obtuvimos resultados muy similares en todos los experimentos que realizamos con otras configuraciones de DVE. El parámetro c del método GRASP se fijó en un número óptimo de avatares críticos c de 50 y 125 para las configuraciones MEDIUM1 y MEDIUM2, respectivamente [8].

La tabla 1 muestra los resultados de la evaluación del funcionamiento obtenidos para la configuración MEDIUM1 cuando las particiones proporcionadas por diversos métodos se simulan bajo 2 combinaciones de patrones de movimiento y distribuciones iniciales de avatares: HPA y distribución uniforme (HPA-Unif) y patrón HPN sobre distribución inicial clustered (HPN-Clus). Para cada combinación de patrón del movimiento y partición inicial de avatares, la tabla 1 presenta dos columnas, una para cada uno de los métodos de particionado considerados, ALB y GRASP (GR). Las primeras tres filas, etiquetadas con Sx, muestran el porcentaje *máximo* de utilización de la CPU alcanzado en cada servidor del DVE durante la simulación con cada método de particionado. La fila siguiente muestra el valor medio de ASR (en milisegundos) para los mensajes enviados por todos los avatares durante la simulación. La penúltima fila muestra el número de avatares que presentaron un retardo de ida y vuelta menor de 250 ms. Es decir, esta fila muestra el número de avatares a los que se les proporciona QoS en cada método de particionado. Finalmente, la última fila muestra el número de avatares migrados por cada método de particionado durante la simulación.

Cuadro 1: Results for MEDIUM1 DVE configuration

	HPA-Unif		**HPN-Clus**	
	ALB	GR	ALB	GR
S0	94	96	87	91
S1	97	91	99	98
S2	86	97	94	89
Av. ASR	219	168	247	189
QoS	168	238	170	235
Migr.	21	81	48	121

La tabla 1 muestra que para la configuración MEDIUM1 ambos métodos evitan la saturación del sistema DVE, puesto que ninguno de los servidores alcanza el 100 % de utilización de la CPU. Sin embargo, los valores medios de ASR obtenidos con el método GRASP son significativamente más bajos que los obtenidos con el método ALB para ambos patrones del movimiento. En el caso del patrón de HPN, el valor medio de ASR proporcionado por el método ALB está cerca del límite de 250 ms., y el del método GRASP es mucho más bajo. En términos de cantidad de avatares a los que se les proporcionó QoS, el método propuesto aumenta considerablemente esta medida de funcionamiento. El método de ALB proporciona solamente QoS a alrededor del 68 % de la población en ambos patrones HPN-uniforme y HPA-Clustered, mientras que el método prop-

uesto puede proporcionar QoS a alrededor del 95 % de la población de avatares para los mismos patrones. Finalmente, aunque el método propuesto migra más avatares que el método ALB, no migra más del 30 % de la población (250 avatares) durante toda la simulación. Por lo tanto, GRASP está lejos de migrar más del 30 % de la población en cualquier ejecución simple del método de búsqueda. La tabla 2 muestra los resultados obtenidos para una configuración MEDIUM2. Esta tabla tiene el mismo formato que la tabla 1, pero muestra los resultados para diez servidores. Los resultados presentados en esta tabla corresponden al patrón del movimiento HPN con una partición inicial uniforme de avatares (HPN-Unif) así como al patrón HPA con una partición inicial sesgada de avatares (HPA-skew).

Cuadro 2: Results for MEDIUM2 DVE configuration

	HPN-Unif		**HPA-Skew**	
	ALB	GR	ALB	GR
S0 (%)	97	98	100	99
S1 (%)	98	97	98	97
S2 (%)	98	96	97	100
S3 (%)	96	97	100	100
S4 (%)	98	94	98	100
S5 (%)	97	98	100	98
S6 (%)	98	96	97	100
S7 (%)	94	98	100	98
S8 (%)	95	97	100	100
S9 (%)	96	97	98	99
ASR	279	234	412	259
QoS	214	404	101	240
Migr.	56	139	103	271

La tabla 2 muestra que en el caso del patrón HPN el sistema trabajó por debajo de su punto de saturación y que la máxima utilización de CPU no fue superior al 98 % en ningún servidor. Sin embargo para el caso del patrón HPA el sistema trabajó bajo saturación, y solamente se produjeron 26 iteraciones antes de que el sistema se colapsara debido a la alta carga de trabajo producida por los avatares. Para este patrón de movimiento se midieron todos los valores para las 26 iteraciones. Como era de esperar, para este patrón de movimiento varios servidores alcanzaron el 100 % de utilización de la CPU con ambos métodos.

La tabla 2 muestra las mayores diferencias de funcionamiento entre los dos métodos de particionado considerados. Efectivamente, los valores medios de ASR proporcionados por ambos métodos presentan diferencias signicativas para el patrón HPN, y el método GRASP disminuye este valor hasta la mitad del valor proporcionado por el método ALB en el caso del patrón de HPA. Estos resultados demuestran que cuanto mayor es la carga de trabajo soportada por el sistema, mayores son las diferencias de funcionamiento entre los dos métodos considerados. En términos del número de avatares a los que se proporcionó QoS, el método GRASP también presenta un aumento significativo. El método GRASP dobla prácticamente el número de avatares con QoS para los patrones de HPN y de HPA. Estos resultados muestran que el método propuesto puede aumentar el número de avatares a los que se les proporcione QoS no sólo cuando el sitema soporta poca carga, sino también cuando el sistema trabaja condiciones de saturación. Finalmente, el número de migraciones en ninguno de los métodos es mayor del 30 % de la población. Estos resultados validan el método GRASP como método de particionado capaz de proporcionar QoS a un alto número de avatares.

5. Conclusiones y Trabajo Futuro

En el presente trabajo, hemos propuesto un método de particionado para los sistemas DVE basado en una técnica meta-heurística que busca el mejor compromiso entre la latencia, la productividad del sistema y la eficiencia de particionado.

Los resultados de la evaluación del funcionamiento demuestran que el método propuesto puede mantener el sistema por debajo del punto de saturación (si esto es posible) y

al mismo tiempo puede aumentar el número de avatares a los que se les proporcione QoS. Agrupar avatares vecinos en el mismo servidor permite reducir carga computacional necesaria para mantener una vista consistente del mundo virtual. De este modo, el sistema puede realizar más rápido esta tarea, y la cantidad de avatares con QoS aumenta de manera significativa.

Como una posible futura línea de trabajo, nos planteamos estudiar el peso relativo de cada uno de los tres términos de la función de calidad $f(P)$. Los coeficientes de las tres funciones pueden ser distintos y esta situción puede mejorar el funcionamiento del método de particionado. Por otro lado, nos proponemos adaptar el método propuesto con el fin de permitir diferentes limitaciones temporales para distintos avatares.

Referencias

[1] S. Singhal and M. Zyda, *Networked Virtual Environments.* ACM Press, 1999.

[2] J. C. Lui and M. Chan, "An efficient partitioning algorithm for distributed virtual environment systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, 2002.

[3] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato, "An adaptive load balancing technique for distributed virtual environment systems," in *Proc. of Intl. Conf. PDCS'03*, IASTED. ACTA Press, 2003, pp. 256–261.

[4] Z. Choukair, D. Retailleau, and M. Hellstrom, "Environment for performing collaborative distributed virtual environments with qos," in *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'00).* IEEE Computer Society, 2000, pp. 111–118.

[5] T. Henderson and S. Bhatti, "Networked games: a qos-sensitive application for qos-insensitive users?" in *Proceedings of the ACM SIGCOMM 2003.* ACM Press / ACM SIGCOMM, 2003, pp. 141–147.

[6] P. Morillo, J. M. Orduna, M. Fernández, and J. Duato, "On the characterization of distributed virtual environment systems," in *Euro-Par' 2003 - LNCS 2790*, ACM. Springer-Verlag, 2003, pp. 1190–1198.

[7] K. Lee and D. Lee, "A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution," in *Proceedings of VRST 2003.* ACM, 2003, pp. 160–168.

[8] P. Morillo, J. M. Orduña, M. Fernández, and J. Duato, "A comparison study of metaheuristic techniques for providing qos to avatars in dve systems," in *ICCSA' 2004 - LNCS 3044.* Springer-Verlag, 2004, pp. 661–670.

[9] FIPA, *FIPA Agent Management Specification*, 2000.

[10] IEEE, *1278.1 IEEE Standard for Distributed Interactive Simulation-Application Protocols (ANSI)*, 1997.

[11] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture.* Prentice-Hall PTR, 1999.

[12] N. Beatrice, S. Antonio, L. Rynson, and L. Frederick, "A multiserver architecture for distributed virtual walkthrough," in *Proceedings of ACM VRST'02*, 2002.

[13] F. C. Greenhalgh, "Analysing movement and world transitions in virtual reality tele-conferencing," in *Proceedings of ECSCW'97*, 1997.

[14] M. Matijasevic, K. P. Valavanis, D. Gracanin, and I. Lovrek, "Application of a multi-user distributed virtual environment framework to mobile robot teleoperation over the internet," *Machine Intelligence & Robotic Control*, vol. 1, no. 1, pp. 11–26, 1999.