Protocolo de comunicaciones para renderización distribuida en tiempo real

C. F. Perez-Monte[1,2], G. J. Mercado[1], J. C. Taffernaberry[1], M. F. Piccoli[2]

[1]GridTICs – Universidad Tecnológica Nacional - Facultad Reg. Mendoza (UTN-FRM)
Mendoza, Argentina

[2]LIDIC – Universidad Nacional de San Luis (UNSL)
San Luis, Argentina

{cristian.perez,gustavo.mercado,carlos.taffernaberry}@gridtics.frm.utn.edu.ar, mpiccoli@unsl.edu.ar

Resumen. *La renderización de volúmenes y otros procesos que requieren elevada potencia de cómputo para su procesamiento suelen utilizar sistemas de cómputo paralelos y distribuidos para resolver dicho problema. En el presente trabajo se describe un diseño de protocolo de comunicaciones orientado al mejor esfuerzo, energéticamente eficiente y tolerante a fallas para resolver problemas de alta complejidad computacional en tiempo real. Para lograr estos objetivos se propone un protocolo de capa de aplicación, el cual utiliza UDP en capa de transporte, IPv6 (unicast - multicast) en capa de red y Gigabit Ethernet con control de flujo en capa de enlace de datos. Finalmente algunos resultados experimentales son mostrados.*

1. Introducción

La renderización foto realista de volúmenes en tiempo real requiere una elevada potencia de cómputo para su ejecución debido a la resolución, velocidad de cuadros y efectos de iluminación requeridos. Esta demanda computacional puede ser resuelta mediante la utilización de un sistema distribuido compuesto por numerosos nodos conectados a través de una red. Esto es posible si el problema se resuelve aplicando técnicas de paralelización, las cuales permiten distribuir el procesamiento entre los nodos, y usando una red con ciertos requisitos, tal como buena performance para las comunicaciones. En este trabajo planteamos un modelo de sistema, donde los nodos tienen diferentes roles, y un modelo de distribución de tareas, los cuales son el punto de partida para el diseño del protocolo de comunicaciones.

En las próximas secciones introducimos los conceptos básicos utilizados en el presente desarrollo.

1.1. Renderización en tiempo real y foto realista

Un modelo 3D es una representación de un objeto 3D, el cual puede estar descripto ya sea a través de polígonos, a través de un volumen compuesto de voxels u otra técnica. Renderización es un término frecuentemente utilizado para describir el proceso por el cual a un modelo generalmente 3D se le realiza un procesamiento por el cual, mediante determinadas variables de entradas, se genera una imagen 2D desde un punto de vista del observador. Tanto la ubicación del observador como su orientación visual son las variables de entrada mas importantes, sin embargo pueden existir otras como la ubicación de la iluminación. En la renderización en tiempo real, las variables de entrada pueden variar constantemente debido al cambio constante de posición del observador o las luces, implicando una generación constante de imágenes 2D.

La renderización foto realista tiene como objetivo generar una imagen con una realidad visual lo mas cercana posible a la realidad usando técnicas visuales como iluminación y sombras que requieren un procesamiento elevado.

1.2. Renderización paralela distribuida

Se denominan técnicas a determinadas formas de paralelizar un procesamiento gráfico. Por otro lado se denomina política a una técnica o conjunto de técnicas aplicadas en el sistema.

En renderización paralela distribuida existe una clasificación, la taxonomía de Molnar [Molnar 1994] [Cox 1995], la cual básicamente establece dos tipos de técnicas basadas en la manera en cómo distribuye el trabajo entre los diferentes nodos de un sistema distribuido. Estas dos técnicas se conocen con el nombre de *Sort first* o *2D* y *Sort last* o *DB*, sus principales características son:

- La renderización paralela distribuida *Sort first* o *2D* propone la división de la pantalla en áreas no solapadas, asignando una a cada nodo o unidad de cómputo disponible. De esta forma, cada nodo renderiza una sección determinada de la pantalla completa. Generalmente los algoritmos de renderización utilizados en este caso son del tipo *Image Order*, existen algunos desarrollos en sistemas con memoria compartida [Palmer 1998] y en sistemas GPU-CPU con memoria distribuida [Bajaj 2000]. Si bien en ambos casos se logra un buen desempeño, la resolución de las imágenes no es muy alta. Finalmente con resolución de imágenes muy superiores, así como también volúmenes de gran tamaño, se logran excelentes resultados [Schwarz 2010].

- En la técnica *Sort last* o *DB*, la renderización es hecha dividiendo sectores de volúmenes o primitivas, aplicar la renderización y finalmente realizar la composición de los resultados parciales y obtener la imagen final. Este método es especialmente adecuado para algoritmos de renderizado *Object Order* para resoluciones de la pantalla de salida no demasiado altas y grandes tamaños de volúmenes [Marchesin 2008] [Engel 2006]. Ésta es una técnica altamente escalable con una desventaja importante, la necesidad de interacción entre los diferentes nodos para realizar la composición.

Por último existe, fuera de la taxonomía de Molnar, una tercera técnica de renderización distribuida denominada *Alternate Frame Rendering* (*AFR*), muy utilizada en ambientes con varias GPUs en una única PC [Monfort 2009]. Esta se basa en la distribución de cuadros de pantalla.

De las tres técnicas antes citadas, *2D* y *AFR* pueden ser aplicadas en nuestro sistema, constituyendo una buena solución para la aplicación en la renderización foto realista en tiempo real. Otras técnicas específicas propias de algunos algoritmos de renderización también pueden utilizarse.

Todas estas técnicas pueden combinarse y definir nuevas políticas aplicables a ambientes distribuidos-paralelos para resolver problemas con gran demanda computacional.

El estudio de estas políticas de renderización exceden el alcance de la presente publicación pero el protocolo propuesto está adaptado para funcionar con cualquiera de ellas.

2. Sistema de Renderización Distribuida en Tiempo Real

El sistema propuesto está compuesto por 3 tipos de nodos, cada uno de los cuales tiene un rol definido. Los roles que puede asumir un nodo en el sistema son:

- *Nodo Administrador:* Es el nodo que recibe y administra las variables de entrada recibidas desde un dispositivo de entrada para transmitirlas por red a los *Nodos Procesador*. Adicionalmente debe anunciar a los *Nodos Procesador* que nodo o *Nodos Integrador* les corresponde. Opcionalmente puede determinar o no que procesa cada *Nodo Procesador* de acuerdo a políticas de implementación.
- *Nodo Procesador:*Es el nodo que realiza el renderizado y envía el trabajo finalizado por red al *Nodo Integrador*. Puede asumir, en el caso de que el *Nodo Administrador* no lo haga, la función de auto asignarse la tarea a procesar.
- *Nodo Integrador:* Es el nodo que recibe todas las tareas procesadas y las integra pudiendo ser el encargado también de realizar la visualización en tiempo real de la escena renderizada. Pueden existir múltiples *Nodos Integrador* pero existe una limitación, cada *Nodo Integrador* tendrá asignado determinados segmentos de la imagen que no podrán estar asignados simultáneamente a otros *Nodos Integrador*.

Considerando los distintos tipos de nodos, el sistema de renderización distribuido consta de un *Nodo Administrador* para coordinar las tareas a realizar por los demás nodos, el cual puede estar separado del resto del sistema en una red alejada permitiendo el control del sistema remotamente; varios *Nodos Procesador*, quienes realizan la renderización propiamente dicha, y los *Nodos Integrador* responsables de reunir y realizar la visualización en tiempo real del volumen. Estos dos últimos deben estar ubicados físicamente cercanos para su conexión a través de una red de alta performance y baja latencia. En la siguiente sección analizamos el modelo de comunicación propuesto para poder gestionar este sistema.

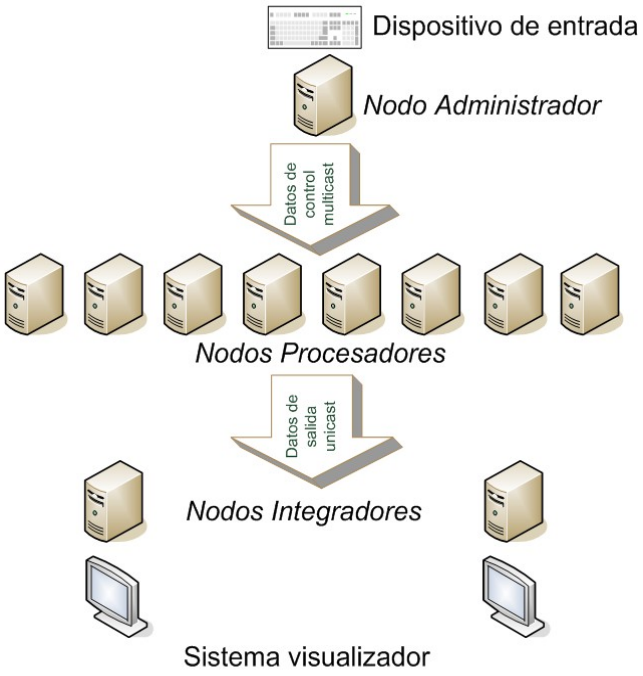

Figura 1. Esquema del sistema

3. Modelo de comunicación propuesto

Los nodos en el sistema interaccionan entre si de una manera determinada (ver figura 1), estableciendo de este modo dos tipos de comunicaciones, las cuales son:

Información de control : Esta comunicación tiene lugar desde el *Nodo Administrador* a los *Nodos Procesador* para enviar la información de las variables de entrada e información de pertenencia de los segmentos de tareas que le permite conocer a los *Nodos Administrador* a cual *Nodos Integrador* enviar cada segmento. Dado los escasos requisitos necesarios de ancho de banda, esta información puede ser transferida a través de enlaces WAN a redes remotas. La utilización de multicast disminuye los efectos que la latencia de los enlaces WAN pueda ocasionar en el funcionamiento del sistema.

Información de salida: Esta comunicación se lleva a cabo entre los *Nodo Procesador* y los *Nodo Integrador*. Cada una de ellas contiene la información del streaming multimedia distribuido, el cual está compuesto por los sucesivos cuadros en el tiempo generados por cada uno de los *Nodo Procesador*. Los requisitos de gran ancho de banda y baja latencia requieren de un enlace LAN exclusivo para la comunicación de salida entre los *Nodo Procesador* y los *Nodo Integrador*.

En las siguientes secciones se detallan las principales características del protocolo y el modo de funcionamiento.

3.1. Segmentación de datos

Generalmente en entornos de computación masivamente paralela como el usado en el ambiente de las arquitecturas GPUs (compuesta por grupos de procesadores SIMD) es común usar modelos en los cuales todas las tareas a procesar se dividen en bloques, los cuales son asignados a diferentes procesadores SIMD de la arquitectura (Kirk 2010). Generalmente es común en sistemas distribuidos contar con nodos con GPU, por ello es importante utilizar un modelo de comunicación el cual mantenga sobre el sistema completo un esquema similar al utilizado por cada nodo. Es por ello que una tarea renderizada (información de salida) que está compuesta por cuadros sucesivos en el tiempo, se puede dividir en segmentos, cada uno de los cuales representa una porción de un cuadro en un momento determinado del tiempo.

Se puede definir a este segmento como una unidad mínima de procesamiento a ser asignada a un *Nodo Procesador* de la red, esto es equivalente a los bloques de CUDA o work-group de OpenCL (Unidad mínima que procesamiento que puede asignarse a un Streaming Processor, SM) La diferencia entre ellos está en que un bloque es asignado a un sólo SM de la GPU, en cambio un mismo segmento de un mismo cuadro puede ser procesado (de forma diferente o no) en más de un nodo, por lo cual el protocolo debe poder contemplar esta situación. Métodos de renderización iterativos o tolerantes a fallas nodales pueden justificar este modo de funcionamiento. Para los casos de sistemas en los cuales cada *Nodo Procesador* utiliza su GPU para procesar y conseguir una máxima performance, los segmentos deberán ser múltiplos de los work-group de OpenCL, por ello el protocolo permite que dicho tamaño sea configurable adaptándose a la arquitectura utilizada.

Los *Nodos Integrador* en cambio solo pueden ser asignados para recibir segmentos específicos. Un mismo segmento no puede ser recibido por más de un *Nodo Integrador*. Este tipo de arquitectura permite que los *Nodos Integrador*, en caso de ser mas de uno, realicen también la visualización en forma distribuida.

3.2. Modo de funcionamiento

El modo de funcionamiento a través de tiempo consta de las varias etapas en donde los estados generales del sistema compuestos por las variables de entrada y sus correspondiente variables de tiempo son administradas por el *Nodo Administrador.*

Los *Nodos Procesador* se anuncian al *Nodo Administrador*. Luego éste envía al grupo multicast de todos los *Nodos Procesador* los datos de control, las variables de entrada capturadas de un dispositivo de entrada con información del tiempo al momento de la captura. El envío se realiza periódicamente con lapsos de tiempo inferiores al tiempo de refresco del sistema visualizador asegurando así que siempre los *Nodos Procesador* tengan tarea para realizar. El *Nodo Administrador,* también a través de datos de control, le indica a los *Nodos Procesador* a cual *Nodo Integrador* le corresponderá cada segmento de pantalla, no pudiendo haber mas de un mismo *Nodo Integrador* asignado a un segmento determinado.

Cada *Nodo Procesador* recibe la información de control, eligiendo la mas actual posible y descartando el resto, y a partir de la cual realiza el procesamiento adecuado y genera un tráfico de información de salida compuesto de segmentos de los cuadros, los cuales envían por tráfico unicast a los *Nodos Integrador* correspondientes a cada segmento. El envío de esta información se realiza tan rápido como cada *Nodo Procesador* es capaz de hacerlo siendo el límite impuesto ya sea por el procesamiento o por el ancho de banda de la red que conecta los *Nodos Procesador* con los *Nodos Integrador.*

Los N*odos Integrador* tienen como función reunir el streaming multimedia distribuido y finalmente visualizarlo en un sistema visualizador compuesto de tantas pantallas como *Nodos Integrador* existan.

4. Características e interoperatividad del protocolo

El protocolo de aplicación diseñado tiene las siguientes características:

- Mejor esfuerzo: El protocolo al igual que el sistema en su conjunto deben tener una filosofía del “mejor esfuerzo” en el sentido que ante limitaciones reales tales como ancho de banda de enlace, *Nodos Procesador* con diferente potencia de cómputo, nodos ocupados, etc, el sistema debe hacer el mejor esfuerzo posible para llevar a cabo la tarea. Es por ello que el protocolo IPv6-UDP es orientado a datagramas siguiendo también un modelo de entrega de mejor esfuerzo.
- Tolerante a saturación de enlace: De la mano con el concepto anterior, el protocolo debe ser capaz de aprovechar lo mas posible el ancho de banda de red y ante saturación del mismo debe ser capaz de continuar en operación normal. La pérdida de información no repercute en el sistema ya que cada datagrama incluye toda la información necesaria sobre el estado del sistema.
- Energéticamente eficiente: El procesamiento de información insume un consumo de energía considerable sobre los *Nodos Procesador,* y dado que el sistema de comunicaciones realiza el máximo esfuerzo por transportar los datos de salida pero no siempre puede lograrlo, es conveniente procesar solo la cantidad de datos que el enlace entre los *Nodos Procesador* y los *Nodos Integrador* sea capaz de transportar.
- Tolerante a fallas de *Nodos Procesador* : Cuando la cantidad de *Nodos Procesador* crece, las posibilidades que al menos un nodo falle se incrementa, es por ello que, si la política de renderización lo acompaña, el protocolo está preparado para tolerar la falla de un nodo y continuar en operación.

4.1. Arquitectura de la pila (stack) de protocolos

La arquitectura del stack de protocolos que se utiliza para cumplir con los objetivos precedentes, es la siguiente:

Capa de enlace de datos: En el caso de los enlaces entre los *Nodos Procesador* y *Nodos Integrador* se utiliza Ethernet con control de flujo (opcional) (IEEE 1997). En caso de tener control de flujo se permite que ante saturación de enlace los *Nodos Procesador* solo procesen lo que son capaces de enviar por la red reduciendo el consumo energético.

Capa de red: Protocolo IPv6 (Deering 1998), el estándar de red de alcance mundial en su versión mas reciente que lo hace adecuado para computación distribuida y con multicast para reducir el tráfico de información de control.

Capa de transporte: Protocolo UDP (Postel 1980), implementando un protocolo con un modelo de entrega de mejor esfuerzo y tolerante a saturación de enlace.

4.2. Descripción detallada del protocolo

En el protocolo se distinguen dos tipos de comunicación, las cuales son Información de Control Multicast e Información de Salida Unicast:

- Información de Control Multicast:

Está constituida por datagramas que tienen como función principal el envío de los datos de entrada. Incluyen 3 cabeceras, de las cuales dos son opcionales para el envío de los datos.

- Cabecera de control mandatoria: Posee la información obligatoria de cada segmento de control.
- Cabecera de control opcional general: Información opcional para anunciar a los *Nodos Procesador*, dicha información puede ser sobre los propios *Nodos Procesador* (que segmentos debe procesar cada uno) o sobre los *Nodos Integrador* (que segmentos tienen asignado cada *Nodo Integrador*).
- Cabecera de control opcional específica por nodo: Se envía en esta cabecera la información específica para cada *Nodo procesador* o *Nodo Integrador*. La cabecera opcional general indica cuantas de estas cabeceras opcionales existirán.

El datagrama de la información de control se muestra en la tabla (ver tabla 1).

Tabla 1. Información de control multicast

	Campos	Descripción
Cabecera IPv6 + Cabeceras opciones	Definidos por el protocolo	Definidos por el protocolo según RFC 2460
	Dirección IPv6 de destino	Dirección IPv6 de destino multicast del grupo de los Nodos Procesadores según RFC 3306
Cabecera UDP	Definidos por el protocolo	Definidos por el protocolo según RFC 768

Cabecera de control mandatoria	Input Var Type	Descriptor de las variables de entrada
	Time stamp	Campo de marca de tiempo de variable de entrada
	Control Counter	Campo de número de dato de control enviado
	No Var Counter	Campo de número de dato de control enviado con idéntica variable de entrada.
	Next Header	Siguiente cabecera
Cabecera de control opcional general	Node Type	Tipo de nodo (procesador o integrador)
	Node Counter	Cantidad de nodos
	Node Info	Tipo y cantidad de información del nodo
	Next Header	Siguiente cabecera
Cabecera de control opcional específica por nodo	ID Node	Identificador único de nodo: 64 bits de menor peso de dirección IPv6 de nodo procesador o integrador
	Segment Info	Campo de información específica para nodo: Información de segmento asignado para recibir por parte de nodo integrador o asignado para procesar por parte de nodo procesador
	Next Header	Siguiente cabecera
Cuerpo	-	Datos de variables de entrada

- Información de Salida Unicast:

Como se indicó anteriormente esta información se divide en segmentos.

Cada uno de esos segmentos tienen además 3 cabeceras compuestas de varios campos.

- Cabecera de información general del sistema: Esta cabecera hace una descripción general de las variables del sistema completo y dicha información se repite en cada segmento permitiendo así la tolerancia a fallas nodales o de red.
- Cabecera de información general del cuadro: Información que se envía en cada segmento que generalmente no varía de segmento a segmento de un mismo cuadro.
- Cabecera específica de segmento: Información específica de segmento.

El datagrama de la información de salida se muestra en la tabla (ver tabla 2).

Tabla 2. Información de salida unicast

	Campos	Descripción
Cabecera IPv6 + Cabeceras opciones	Definidos por el protocolo	Definidos por el protocolo según RFC 2460
	Dirección IPv6 de destino	Dirección IPv6 de destino unicast de nodo integrador correspondiente al segmento transmitido
Cabecera UDP	Definidos por el protocolo	Definidos por el protocolo según RFC 768
Cabecera de información general del sistema	Frame Dim	Campo de cantidad de dimensiones del streaming de salida (generalmente es 2)
	Frame Resolution	Campos de resolución total del cuadro en cada dimensión
	Pixel type	Campo de cantidad de bit por pixel y tipo de representación
Cabecera de información general del cuadro	Segment Dim	Cantidad de dimensiones del segmento
	Segment Resolution	Tamaño del segmento en cada dimensión
	Total Segment	Cantidad total de segmentos del cuadro
Cabecera específica de segmento	ID Segment	Identificador de segmento
	Representation Segment	Representación del segmento. Campo de segmento comprimido o no. Compresión con pérdida o no de calidad. Tipo de compresión
	Segment Size	Tamaño comprimido del segmento
	Time stamp	Campo de marca de tiempo de variable de entrada: Cada segmento tendrá marcas de tiempo que lo identifiquen como perteneciente a determinada variable de entrada y por lo tanto perteneciente a determinado cuadro en el tiempo.
	Segment Counter	Campo de número de segmento generado con igual identificador de segmento.

	Render Type	Tipo de método usado para la renderización
	Iteration Counter	Según corresponda nivel de procesamiento ejercido como cantidad de iteraciones del segmento
	No var counter	Campo de número de segmento procesado con idéntica marca de tiempo e identificador de segmento, útil para políticas específicas de renderizado.
Cuerpo	-	Segmento

4.3. Requisitos de ancho de banda de red teóricos y latencia máxima del sistema

En la presente sección se determinan los requisitos mínimos de ancho de banda del sistema para que la red no sea limitante del rendimiento y la latencia que existe desde el momento que una variable de entrada es ingresada por el dispositivo de entrada hasta que la misma se hace efectiva en el sistema visualizador. Por último el rendimiento que poseerá el sistema en caso de que la red sea limitante del rendimiento.

Siendo:

m : Número de *Nodos Procesador*

l : Número de *Nodos Integrador*

R_X : Resolución horizontal

R_Y : Resolución vertical

B_S : Cantidad de bytes por cada pixel

R : Rendimiento en cantidad de cuadros por segundo

BW : Ancho de banda de red

T_s : Latencia de renderizado

T_{bw} : Tiempo de transferencia del cuadro

T_n : Latencia de red

T_{nap} : Latencia de red entre *Nodo Administrador* y *Nodo Procesador*

T_{npi} : Latencia de red entre *Nodo Procesador* y *Nodo Integrador*

T_t : Latencia de interactividad total

El caso extremo de ancho de banda utilizado por el canal de comunicaciones en la información de salida estará dado en su funcionamiento con la técnica AFR o políticas que la utilicen y será:

$$BW = \frac{R_X * R_Y * B_S * R * m}{l}$$

La latencia total del sistema estará comprendida por la suma de la latencia en la renderización de cada cuadro de la pantalla sumada a la latencia de la red y al tiempo de

transferencia del cuadro de pantalla en la red. El tiempo de latencia de red está determinado por la suma de la latencia *Nodo Administrador – Procesador* y *Nodo Procesador – Nodo Integrador*, en la cual esta última, por ser pequeña, puede despreciarse. El tiempo de transferencia del cuadro en la red puede determinarse según el ancho de banda de la red usada mientras que el tiempo de renderizado depende del hardware de los *Nodos Procesador*.

$$T_t = T_s + T_{bw} + T_n = T_s + T_{bw} + T_{npi} + T_{nap} \approx T_s + T_{bw} + T_{nap} = T_s + \frac{R_X + R_Y + B_S}{l * BW} + T_{nap}$$

Si bien la información de salida puede transferirse a través de redes de baja velocidad, el ancho de banda en este caso será un limitante en el rendimiento máximo del sistema expresado en cuadros por segundo (R), siendo este:

$$R = \frac{BW * l}{m * R_X * R_Y * B_S}$$

4.4. Tráfico de salida desperdiciado

Se le denomina *tráfico desperdiciado* al tráfico generado por los *Nodos Procesador* que no puede ser procesado por los *Nodos Integrador*.

Existen básicamente dos tipos de *tráfico desperdiciado*:

- *Tráfico desperdiciado* por *Nodo Integrador:* El *tráfico desperdiciado* por el *Nodo Integrador* generalmente es por de-sincronismo. Los *Nodos Procesador* trabajan en un modelo asincrónico de mejor esfuerzo, sin embargo puede ocurrir que los *Nodos Procesador* envíen segmentos que sean recibidos con atraso para ser visualizados, en tal caso los *Nodos Integrador* deben descartarlo.

También si los buffers de recepción del *Nodo Integrador* no poseen el tamaño suficiente o el *Nodo Integrador* no tiene la velocidad suficiente para procesarlos se pueden generar pérdidas de este tipo.

- *Tráfico desperdiciado* por pérdidas en el enlace: Todo tráfico generado por los *Nodos Procesador* que no es recibido por los *Nodos Integrador*.

Un caso específico es cuando se produce saturación de enlace en un medio sin control del flujo, por lo cual la pérdida de una trama que contiene un segmento o parte del mismo hará que dicho segmento no pueda ser reconstruido en el *Nodo Integrador*.

5. Resultados

Los resultados experimentales se realizaron en un cluster de computación gráfica de renderizado distribuido compuesto por 2 *Nodos Procesador*, un *Nodo Administrador* y un *Nodo Integrador* conectados mediante una red Ethernet Gigabit. Se utilizó un MTU estándar de 1528 bytes. El tamaño del segmento incluido en la carga del datagrama UDP fue definido en 48K.

Las características de la arquitectura de cada uno de los *Nodos Procesador* del sistema son:

- Nodo 1 procesador: Intel Core 2 Duo 4 GB de RAM GPU GTX560 1GB de RAM
- Nodo 2 procesador: AMD Fx6200 4 GB de RAM GPU GTX650 1 GB de RAM

En la siguiente tabla (ver Tabla 3) se muestra la potencia de cómputo combinada y anchos de banda medidos sobre el *Nodo Integrador* con diferentes composiciones del

sistema, valor de n y control de flujo en Ethernet activado o no.

Se le denomina *n* a la cantidad de iteraciones del método de renderizado que permite incrementar el nivel del realismo en forma previa al envío de cada cuadro y la potencia de cómputo de combinada de todos los *Nodos Procesador* intervinientes se presenta en MOPS (millones de operaciones /segundo).

Tabla 3. Potencia de cómputo combinada y anchos de banda

Composición del sistema distribuido	n	Control de flujo de Ethernet	Potencia de cómputo [MOPS]	Ancho de banda útil [MB/s]	Ancho de banda desperdiciado de segmentos perdidos por pérdidas en enlace [MB/s]	Potencia de cómputo desperdiciada [MOPS]
Nodo Procesador 1	1	Indistinto	9,1	54,6	0	0
	2		12,5	37,2	0	0
	4		13	19,5	0	0
Nodo Procesador 2	1		7,9	47,3	0	0
	2		8,5	12,7	0	0
	4		8,9	13,4	0	0
Ambos Nodos simultáneamente	1	Activado	15,6	93,6	0	0
	2		20,9	62,6	0	0
	4		21	31,5	0	0
	1	Desactivado	7,7	46,2	55,7	9,3
	2		14,2	42,6	7,3	2,4
	4		18,5	27,7	5,2	3,4

De la observación de los resultados, es posible determinar que en la configuración con control de flujo activada no hay pérdidas de segmentos y, por lo tanto, no existe desperdicio en potencia de cómputo, siendo así, toda la energía utilizada en el procesamiento de *Nodos Procesador* aprovechada.

En los casos de saturación de enlace, para *n=1* y con funcionamiento de los dos nodos en simultáneo, se pudo comprobar una reducción:

- En la velocidad del streaming con respecto a la suma del tráfico de los dos nodos separados.

- En la potencia combinada con respecto a la potencia de cómputo de los dos nodos separados.

Si bien la limitación de la potencia de cómputo combinada la impone el ancho de banda de la red utilizada, no hay desperdicio de energía. El ancho de banda utilizado es muy cercano al máximo de la red por lo cual se puede observar un mejor esfuerzo por conseguir el mayor rendimiento posible.

En las pruebas en las cuales se ha desactivado el control de flujo de ethernet, se puede observar que en los casos de saturación de enlace existe una pérdida considerable de tráfico, esto es debido a que existe una gran fragmentación del tráfico IPv6 en tramas ethernet ya que el MTU es mucho menor al tamaño del paquete IPv6, por lo cual, la pérdida de una única trama ethernet generará la pérdida de todo un segmento. Incrementar el MTU o decrementar el tamaño del segmento son posibles soluciones en estos casos.

En estas últimas pruebas, si bien el sistema funcionó con menor performance debido a la gran pérdida de tramas, mostró tolerancia a falla de segmentos pudiendo demostrarse su funcionamiento en los casos en los cuales se produce la falla de un nodo o existen pérdidas de segmentos en la red.

6. Conclusiones

El protocolo diseñado cumple con los requisitos necesarios para el control y transmisión de streaming multimedia distribuido. La segmentación de datos ha demostrado ser una técnica apropiada tanto para la transmisión de datos en tiempo real como así también un método para distribuir las cargas de procesamiento por un lado y la visualización por otro. Los protocolos orientados a conseguir el mejor esfuerzo demuestran tener una tolerancia alta a fallas tanto de red como de *Nodos Procesador*.

7. Trabajos futuros

Se realizarán pruebas de utilización por redes a través de Internet en los cuales el *Nodo Administrador* está ubicado en redes geográficamente alejadas de la red en donde se ubican los *Nodos Procesador* y los *Nodos Integrador*. Además se analizará la posibilidad de implementar el protocolo de forma mas general para la gestión remota de cualquier tipo de procesamiento masivamente paralelo y vectorial en aplicaciones en tiempo real.

8. Agradecimientos

Se agradece el asesoramiento y soporte con infraestructura de Dr. Cristian Luciano de Department of Mechanical and Industrial Engineering - University of Illinois at Chicago, de Dr. Silvio Rizzi de Argonne National Laboratory, e integrantes de los Grupos de Investigación GridTICs y LICPaD de UTN-FRM y LIDIC de UNSL.

El trabajo es sostenido económicamente gracias al financiamiento de los proyectos 25J084 “SARA Operation”, PICT2010/29 “Procesamiento para visualización utilizando algoritmos paralelos en GPU y distribuidos en red” ambos de UTN-FRM, PROICO-30310 de UNSL y otras fuentes de financiamiento de grupo GridTICs.

Finalmente se agradece el aporte económico de UTN con una beca de doctorado y al soporte académico de la carrera de doctorado de UNSL.

Referencias

Bajaj, C., Ihm, I., Park, S., and Song, D. (2000). “Compression-based ray casting of very large volume data in distributed environments.” In HPC ’00: Proceedings of the The

Fourth International Conference on High-Performance Computing in the Asia-Pacific Region-Volume 2, pages 720–725.

Cox M. (1995) “Algorithms for Parallel Rendering”. PhD thesis, Department of Computer Science, Princeton University.

Deering S., Hinden R. (1998) “Internet Protocol, Version 6 (Ipv6) Specification” IETF STANDARDS TRACK

Engel K., Hadwiger M., Kniss J. M. , Rezk-Salama C., Weiskopf D..(2006) "Real-Time Volume Graphics." A K Peters, Ltd.

IEEE Standards for Local and Metropolitan Area Networks: (1997) “Supplements to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 Or Better Balanced Twisted Pair Cable (100BASE-T2)," *I*EEE Std 802.3x-1997 and IEEE Std 802.3y-1997 (Supplement to ISO/IEC 8802-3: 1996; ANSI/IEEE Std 802.3, 1996 Edition), vol., no., pp.0_1,324, 1997”

Kirk D. B., Hwu W. W., (2010) “Programming Massively Parallel Processors, A Hands on Approach” , Elsevier, Morgan Kaufmann.

Marchesin, S., Mongenet, C., and Dischler, J.-M. (2008). Multi-gpu sort-last volume visualization. In Eurographics Symposium on Parallel Graphics and Visualization (EGPGV08).

Molnar, S., M. Cox, D. Ellsworth, and H. Fuchs. (1994) “A Sorting Classification of Parallel Rendering.” IEEE Computer Graphics and Algorithms, pages 23-32.

Monfort, J. R., Grossman M. (2009) “Scaling of 3D Game Engine Workloads on Modern Multi-GPU Systems” - The 1st ACM Conference on High Performance Graphics (HPG-09)

Palmer, M. E., Totty, B., and Taylor, S. (1998). “Ray casting on shared-memory architectures: Memoryhierarchy considerations in volume rendering”. IEEE Concurrency, 6(1):20–35

Postel J., (1980) “USER DATAGRAM PROTOCOL” IETF STANDARD PROTOCOL

Schwarz N., Leigh J. , (2010) “Distributed Volume Rendering for Scalable High-resolution Display Arrays”, Grapp – International Conference on Computer Graphics Theory and Applications