Universidad de las Ciencias Informáticas
Facultad 1

Título: Disminución del tiempo de descarga del repositorio para la distribución GNU/Linux Nova.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autora:

Sandra Arango Winograd

Tutores:

Ing. Abel Fírvida Donéstevez

Ing. Jorge Luis Machín Castillo

La Habana

Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los __ días del mes de _____ del año_____ .

_____

Sandra Arango Winograd
Firma del Autor

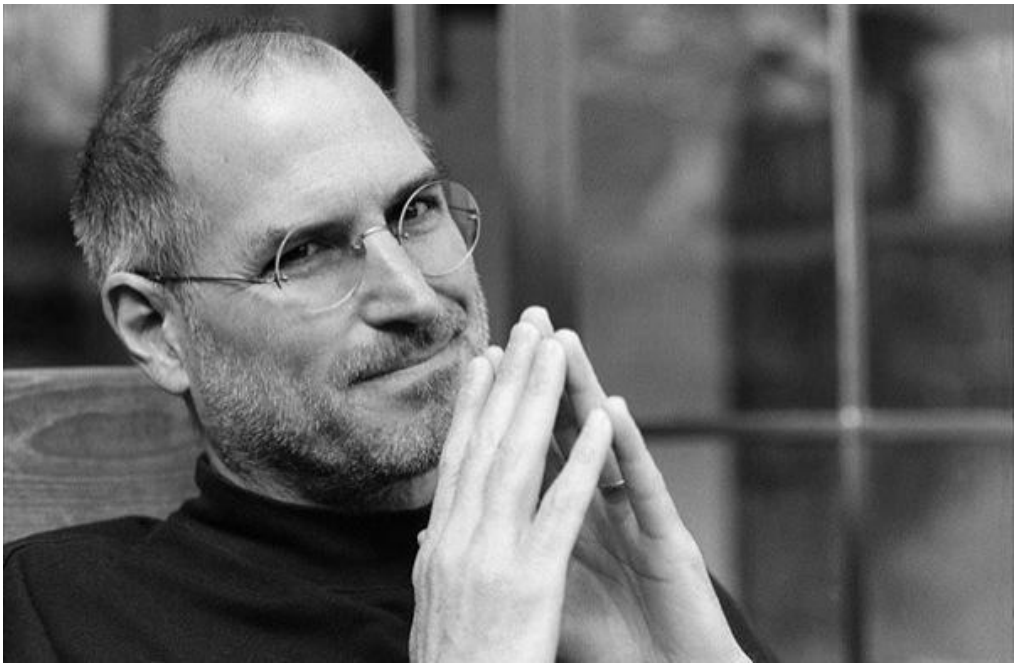
_____

Ing. Abel Fírvida Donéstevez
Firma del Tutor

_____

Ing. Jorge Luis Machín Castillo
Firma del Tutor

FRASE



"Tu trabajo va a llenar gran parte de tu vida, y la única forma de estar realmente satisfecho con él es hacer lo que creas que es un gran trabajo. Y la única manera de hacer un trabajo genial es amar lo que haces. Si no lo has encontrado, sigue buscando. No te detengas. Al igual que con todos los asuntos del corazón, lo sabrás cuando lo encuentres. Y, como cualquier gran relación, sólo se pondrá mejor y mejor, conforme los años pasen. Así que sigue buscando hasta que lo encuentres. No te detengas".

STEVE JOBS.

DEDICATORIA

A quien ha dedicado su vida a mí:

Gracias por hacerme la persona que soy, a ti te lo debo todo, a tus sacrificios, a tu amor incondicional, a tus apoyos constantes que nunca faltaron, a ser mi confidente, a ser mi amiga, a levantarme y darme fuerzas cuando todo parece estar oscuro, a tus mimos y cuido, a ser lo que más quiero en el mundo…

A ti Mami

RESUMEN

En el país actualmente las descargas del repositorio de Nova por diversas razones son lentas y a menudo los administradores de red en instituciones de la OACE[1] no cuentan con el espacio en los servidores para ofrecer hospedaje a esta parte fundamental de la distribución cubana de GNU/Linux. El presente trabajo propone variantes de cómo disminuir el tiempo en las descargas del repositorio. A través de su aplicación, se demuestra no solo la reducción por tres vías efectivas de los tiempos de transmisión de paquetes en las descargas, sino también de sus tamaños para el almacenamiento del repositorio en los servidores. Actualmente las soluciones reducen más de un 50% tanto los tiempos de transmisión de datos, como el volumen actual del repositorio en la distribución.

Palabras claves: compresión, tiempo de descarga, parches, repositorio de Nova.

1 Organismos de la Administración Central del Estado.

ÍNDICE

INTRODUCCIÓN

El programa rector de la informatización de la sociedad cubana plantea desde el año 2004 que la misma se hará utilizando solamente plataformas de software libre, incluyendo dentro del proceso a los Organismos de la Administración Central del Estado (OACE).

Para apoyar dicho procedimiento surge la distribución cubana GNU/Linux Nova como el resultado del Grupo Técnico que forma parte del Grupo Ejecutivo Nacional para la Migración a Software Libre. En el año 2011 uno de sus representantes el Msc. Allan Pierra Fuentes[2] estableció los principios para el desarrollo, uso y aplicación de las Tecnologías de la Información y las Comunicaciones (TIC) en el Gobierno, en los cuales se recogen las bases que deben seguir los sistemas informáticos desarrollados en Cuba de acuerdo a sus características específicas[3]. Dichos fundamentos se conocen coloquialmente como las 4S de Pierra: Seguridad, Soberanía Tecnológica, Socio-adaptabilidad y Sostenibilidad; que sumadas a las cuatro libertades del software libre garantizan la realización de productos informáticos enfocados a la migración tecnológica que se desea realizar en el país y se resumen de la siguiente manera [1]:

Seguridad: El modelo de desarrollo colaborativo que propone el movimiento de software libre, el acceso al código fuente y el exhaustivo proceso de revisión y auditoría de código garantizará un sistema seguro de ataques y sin puertas traseras.

Soberanía Tecnológica: Es la capacidad del país para desarrollarse en dicho campo en forma autónoma. No supone autarquía (independencia absoluta) sino capacidad de decidir sobre su uso y desarrollo.

Socio-adaptabilidad: Las bases tecnológicas para la informatización de Cuba, deben ser hechas por cubanos y para los cubanos, logrando inigualable adaptabilidad a las condiciones de nuestro País.

Sostenibilidad: La constante asimilación e investigación de las nuevas tecnologías, la planificación, los modelos novedosos de comercialización y el uso racional de los recursos humanos, materiales y naturales, garantizarán soluciones, vigencia y sostenibilidad a largo plazo.

2 Allan Pierra Fuentes: autor de los principios para el desarrollo, uso y aplicación de las TIC en el Gobierno.

3 Condiciones políticas, sociales y económicas.

Específicamente maximizar la socio-adaptabilidad es uno de los objetivos en los que más se ha incidido en Nova, lo cual le da un punto a favor con respecto a otras distribuciones libres concebidas en el mundo, acorde a las carencias tecnológicas del país.

Uno de los factores más contraproducentes en el desarrollo informático de la isla es el límite del ancho de banda, que gira entre los 64 Kbps y los 2 Mbps, haciendo actualmente las tareas enfocadas a la red sean morosas, especialmente las descargas del repositorio[4] de Nova. Aunque se han desarrollado acciones técnicas con respecto al mejoramiento del repositorio de la distribución para disminuir tal afectación, los clientes no quedan del todo satisfechos, haciendo que la condicionante de la demora en las descargas del repositorio atente contra la buena aceptación del sistema operativo cubano y al mismo tiempo con los planes de migración expuestos por los OACE en el programa rector de la informatización de la sociedad cubana.

Atendiendo a la situación problemática expuesta, se plantea el siguiente **problema científico**: ¿Cómo elevar la socio-adaptabilidad del repositorio de la distribución GNU/Linux Nova?

Para dar respuesta al problema científico planteado, se asume como **objeto de estudio** los mecanismos que permitan la disminución del tiempo de descarga a través de la red, centrando el **campo de acción** a los mecanismos que disminuyan el tiempo en la descarga de los repositorios de distribuciones basadas en Debian. Proponiéndose como **objetivo general** disminuir el tiempo de descarga del repositorio de la distribución cubana GNU/Linux Nova para aumentar su socio-adaptabilidad.

A partir del objetivo general expuesto se derivan los siguientes **objetivos específicos**:

- Analizar factores que pueden influir en la agilización de procesos de descarga de repositorios en las distribuciones Debian.
- Definir las propuestas para disminuir la transferencia de la descarga del repositorio de la distribución Nova.
- Aplicar las propuestas en el entorno de trabajo de la distribución Nova.
- Validar la efectividad de las propuestas aplicadas a la distribución Nova.

4 Conjunto de paquetes de software disponibles para una distribución GNU/Linux almacenados de forma centralizada.

Partiendo de los objetivos específicos y para darles cumplimiento, se proponen las **tareas de investigación**:

1. Sistematización de factores que influyen en las descargas de repositorios Debian, y valoración de las herramientas y procedimientos existentes que permitan integrarlos a Nova.
2. Descripción de las soluciones propuestas que son favorables para la descarga del repositorio de Nova.
3. Implementación y práctica de las propuestas en Nova.
4. Elaboración y aplicación de validaciones a las integraciones concebidas.

Para guiar el desarrollo de la investigación se sostiene como **idea a defender** que con el decremento de los tiempos de transferencia en las descargas del repositorio de la distribución GNU/Linux Nova, se podrá contribuir a elevar su socio-adaptabilidad en Cuba.

Entre los métodos científicos de investigación que se siguieron para cumplir las tareas planteadas están:

Métodos teóricos

Método Analítico-Sintético:

Permitió analizar el objeto de estudio para poder investigar cada una de sus particularidades tomando información de diferentes fuentes bibliográficas, extraer los elementos más importantes que se relacionan con la descarga de repositorios que puedan ser aplicadas en el campo de acción.

Métodos empíricos

Método de Observación:

Se utilizó para observar el comportamiento de las herramientas y procedimientos existentes que permitan una mejoría en los tiempos de las descargas del repositorio y la reducción de su tamaño.

Método Experimental:

Permitió probar las herramientas y procesos observados, adaptando las condiciones para obtener conocimientos sobre los resultados en determinadas pruebas a las descargas del repositorio en la distribución Nova.

Métodos particulares

La Entrevista:

Se realizaron entrevistas no estructuradas a expertos en temas que posibilitaron el entendimiento de los problemas existentes en las descargas del repositorio de Nova. De la misma forma se realizaron entrevistas en centros empresariales de La Habana, donde se determinaron las condiciones y afectaciones reales sobre situaciones tecnológicas referentes al ancho de banda.

El contenido del trabajo investigativo está dividido en diferentes secciones, donde se tienen tres capítulos que cuentan con el siguiente contenido:

Capítulo 1: "Agilización en los tiempos de las descargas de repositorios"

Se realiza un estudio de los tipos de mejoras para minimizar los tiempos en las descargas que se pueden desarrollar atendiendo fundamentalmente a la variable de tamaño del repositorio. En cada caso se muestran experimentos para probar la veracidad del estudio, compuesto por conceptos y análisis de procesos de descargas, utilizando determinadas herramientas que se puedan aplicar en la distribución Nova.

Capítulo 2: "Propuestas para la disminución de tiempos de descarga del repositorio de Nova"

Se describen los tipos de mejoras que se aplicarán en la próxima versión de Nova, teniendo por cada descripción la forma práctica de integrarlo a la distribución.

Capítulo 3: "Validación de las propuestas"

Se representa la descripción de las validaciones a realizar para cada integración efectuada en el Capítulo 2, verificando su correcto funcionamiento y utilidad para el mejoramiento del sistema operativo cubano.

CAPÍTULO 1: “Agilización en los tiempos de las descargas de repositorios”.

El objetivo del capítulo que inicia es realizar el estudio del arte relacionado con las opciones actuales que puedan existir para lograr una descarga agilizada de los paquetes de un repositorio, ayudando a determinar si es posible su aplicación en la distribución cubana Nova. Por tal motivo es necesaria la definición de convenciones sobre determinados conceptos que son fundamentales en los epígrafes venideros para el buen entendimiento del lector de los contenidos que se describen.

1.1 Convenciones a utilizar.

1.1.1 Paquetes Debian.

Un software en informática se refiere a los programas o aplicaciones que se pueden ejecutar en un ordenador. En sistemas GNU/Linux, las aplicaciones se manejan en términos de paquetes de software.

Existen dos distinciones para los paquetes de software, en primer lugar están los paquetes de código fuente que contienen ficheros de código fuente original o lista de instrucciones que están escrito por programadores y necesariamente deben ser compilados[5] para dar paso a una segunda clasificación: paquetes binarios.

En el caso de la distribución Ubuntu, los paquetes binarios se utilizan para almacenar todo lo que un programa en particular requiere para ejecutarse. Sus desarrolladores definen el término como una colección de archivos construidos en uno solo que se copian en directorios adecuados a través de órdenes de los llamados *scripts*[6] de instalación. También es la forma que tienen los ordenadores para poder entender o interpretar el código fuente que lo generó y así poder ejecutarlos. Existen diferentes tipos de paquetes binarios en dependencia de la compilación que se realice, según el sistema o arquitectura que se utilice [2].

Por otra parte en Debian se puntualiza que un paquete binario contiene los archivos ejecutables, bibliotecas y la documentación asociada con un programa particular o con un conjunto de programas

5 Proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel) para que pueda ser ejecutado por la computadora.

6 Fichero que contiene código de programación de un determinado paquete.

relacionados [3]. Mientras que para la distribución de Fedora cada paquete binario es un archivo comprimido que contiene información del producto, archivos del programa, iconos, documentación y *scripts* de administración donde se incluye al mismo tiempo una firma digital para comprobar su procedencia [4].

Tomando los conceptos oficiales para cada distribución, se concreta el término de paquete binario como archivos comprimidos seguros, donde se encuentra todo lo que un programa específico necesita para ejecutarse. Está formado esencialmente por ficheros asociados a una aplicación particular o conjunto de aplicaciones, que contienen ejecutables, bibliotecas y la documentación necesaria. En definitiva, son el resultado del empaquetado o compilación de los paquetes de código fuente, en dependencia del lenguaje utilizado en su programación, para sistemas o arquitecturas en específico. Por tal motivo existen distintos formatos de paquetes binarios como los .rpm para sistemas de RedHat, openSuse y Mandriva, .pkg de Arch Linux, .tgz de Salckware y .deb para sistemas Debian.

Un **paquete Debian** es entonces un paquete binario que contiene la extensión de formato .deb y se usa específicamente para distribuciones derivadas de Debian o que utilicen su paquetería como en el caso del sistema Nova.

Los paquetes .deb son archivos ar[7] que contienen un conjunto de ficheros en una estructura interna que cuenta con tres archivos: debian-binary, control y data. En el caso de los miembros control y data son archivos comprimidos, donde el primero contiene la información obligatoria del paquete como el nombre, versión, mantenedor, descripción y otros campos opcionales. Data, por su parte, contiene un fichero con el sistema de archivos que instalará el paquete [5].

1.1.2 Gestores de paquetes.

Los gestores de paquetes son herramientas que permiten el proceso automatizado de instalación, actualización o eliminación de paquetes de software desde los repositorios, adicionándole funciones especiales como son comprobación de firma digital, gestionar dependencias según se necesiten y comprobar la suma de verificación del paquete. Entre los gestores de paquetes que acepta Nova se encuentran dpkg, apt, aptitude y synaptic, utilizando en dicha cadena, la aparición de características mejoradas del anterior gestor [6].

7 Archivador de Unix que mantiene grupos de ficheros como un único fichero archivo.

1.1.3 Repositorio.

En los sistemas operativos GNU/Linux, existen diversos paquetes de software que se desarrollan para la propia distribución y otros que se añaden por aportes de comunidades de software libre. El lugar en donde se almacenan y organizan las aplicaciones para que el usuario pueda instalarlo en su ordenador a través de gestores de paquetes o descargarlo manualmente, se denomina en Linux repositorios. Existen en la actualidad distintos conceptos sobre el término, Aaron Isotton, desarrollador de Debian apunta en el sitio oficial de la distribución que: “Un repositorio es un grupo de paquetes organizados en un árbol de directorios especiales los cuales contienen también ficheros adicionales que indican los índices y el chequeo de sumas de los paquetes” [7].

En fuentes oficiales de otras distribuciones también se hace una alusión práctica a lo que se le llama repositorio en Linux. Tal es el caso de Ubuntu, que lo emplea para referirse a servidores que almacenan un conjunto de paquetes de software y que son administrados por los gestores de programas [8]. De la misma forma se tiene coincidencia en el sitio de Fedora, cuando lo tratan como una colección de software ordenado, clasificado y disponible para su uso con herramientas compatibles que lo usen para descargar y manipular software [9].

Atendiendo a las referencias anteriores y a otras tomadas de las fuentes bibliográficas, se define para la investigación que los sistemas de repositorios en distribuciones GNU/Linux son recursos centralizados que almacenan un historial de versiones de paquetes de software de una determinada distribución, alojados de forma estructurada y organizada por índices de los paquetes que estén disponibles, asegurando su autenticidad y que no estén dañados, de forma que los usuarios puedan administrarlos usando gestores de paquetes que lo transfieren vía ftp o http.

1.1.3.1 Estructura de repositorio en Nova.

Los repositorios de Nova actualmente siguen la misma estructura que los de Ubuntu, los cuales están divididos en dos archivos fundamentales: dist y pools.

Ambos directorios cuentan con una estructura formada por los componentes: main, restricted, multiverse y universe, donde el directorio dist almacena en cada uno, los índices de paquetes que corresponden a las diversas versiones y arquitecturas que estén contenidas en el repositorio de una determinada distribución. Pools por su parte, archiva subsecciones ordenadas alfabéticamente, dentro de las cuales se

encuentran los paquetes de las distribuciones. Cada componente almacena sus recursos en dependencia de la estabilidad, mantenedor y licencia de las liberaciones de las nuevas versiones [10].

Existen igualmente en la estructura del repositorio dos ficheros especiales llamados índices que son el Packages.gz para almacenar la información de los paquetes binarios y el Sources.gz para los de código fuente, de donde el gestor de paquetes encontrará la ruta de la aplicación solicitada por el usuario en el repositorio. Todos los datos para acceder al repositorio como su dirección[8], los componentes y tipos de paquetes a descargar, se indican en el archivo de configuración sources.list[9] de cada sistema [7].

1.1.4 Parches de paquetes de software.

Las aplicaciones deben ser capaces de cumplir ciertos requisitos que fallan en determinado momento debido a errores detectados y cambios en las funcionalidades o en las tecnologías utilizadas; por tal asunto es necesario desarrollar nuevas versiones que suplan las dificultades de la anterior.

Anteriormente tal situación solo podía ser resuelta sustituyendo el programa entero en un ordenador por una nueva versión, aunque sus diferencias fueran mínimas, fácilmente daban lugar a más cantidades de datos y la tediosa tarea de la reinstalación del software. Por tales motivos se crean los ficheros que almacenan diferencias entre versiones de una misma aplicación, para que solamente se instalaran en el ordenador los pequeños cambios que solucionen errores o añadan nuevas funcionalidades para un determinado programa, sin necesidad de reinstalarlo.

Por dicho motivo surgen los parches o deltas, que según diccionarios para términos informáticos, se refiere a un trozo de código objeto que se inserta en un programa ejecutable como remedio temporal de un error [11]. Otras fuentes, como la investigación “Algoritmos para compresión delta y sincronización de archivos remotos”, coinciden en que es un esquema donde se describen las diferencias entre la vieja y la nueva versión de un archivo [12]. Por tal razón se define al parche o delta, como un fichero que contiene el tamaño mínimo de información de la diferencia entre versiones de un mismo programa. Puede ser aplicado a diferentes tipos de ficheros, como en el caso de Linux que lo usa para alojar los cambios que ocurren entre los paquetes de software.

8 Dirección en la red del repositorio, se denota con la palabra url.

9 Archivo de configuración del gestor de paquetes APT instalado por defecto en Ubuntu que se ubica en /etc/apt/sources.list

Las herramientas para la creación de deltas tienen dos funcionalidades principales: una para crear parches y otra para aplicarlos. La creación de parches consiste en que se cree el archivo de diferencia entre el par de versiones de un mismo paquete. Por otra parte, la aplicación de un parche ya creado, se refiere a que con la misma herramienta, especificando determinadas opciones, se pueda crear la nueva versión del paquete .deb, idéntica a que si se tuviera la original, con solo el viejo paquete (ya sea instalado o no) y el parche.

1.1.5 Compresión de archivos.

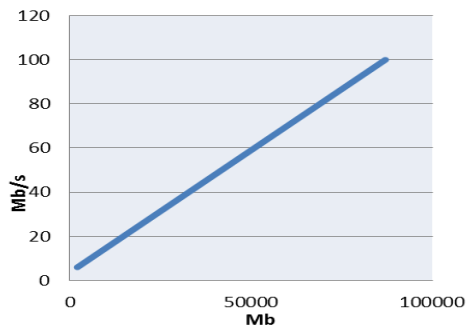
Comprimir un archivo indica la cantidad de volumen de datos que se puede reducir manteniendo la misma información a procesar, transmitir o guardar. Por tanto su objetivo fundamental es la reducción de tamaño, conociéndose como un caso particular de la codificación donde la característica principal es minimizar el volumen del fichero de código original.

1.2 Opciones para reducir el tiempo de descarga de repositorios.

A continuación se hará una descripción de las opciones que existen para poder agilizar la descarga de repositorios en cuanto al tiempo de transmisión de datos por la red a un mismo ancho de banda. Para ello se realizaron los estudios que a continuación se muestran, basándose en la primicia mientras menor sea el tamaño de un repositorio, mejor será su tiempo en la descarga.

1.2.1 Disminución de espacio por reestructuración organizacional.

La mayoría de las empresas cubanas según entrevistas realizadas cuentan hoy con una conectividad a un de ancho de banda máximo de 2 Mbps (Ver Anexo 1). En este marco para descargar un repositorio de Nova de 32 Gb por el canal de red nacional tardaría aproximadamente 36 h ya que como se aprecia en la gráfica 1.1, mientras menor sea el ancho de banda, menor será la cantidad de archivos a transmitir y por ende menor será el tiempo de la descarga de los archivos.



Gráfica 1.1: Mbyte descargados a cada ancho de banda.

Tal situación hace que se haya adoptado un nuevo método en las empresas para mejorar los tiempos en las descargas del repositorio de Nova u otras distribuciones y es la de establecer un repositorio local en una red LAN[10], donde se soporta una velocidad máxima de 100 Mbps por la que acceden las computadoras de dichas instituciones y de este modo las descargas se hacen con una velocidad superior que si accediera cada usuario al repositorio oficial a 2 Mbps.

Para este caso existe el personal adecuado que cuenta con los permisos para acceder al servidor del repositorio oficial de la distribución de Nova por el canal nacional, descargarlo hacia la computadora asignada y luego compartirlo localmente. Otra variante es que el personal se dirija físicamente al centro donde se encuentra el repositorio y lo traslade mediante dispositivos de almacenamiento.

Enfocados en el mismo objetivo, en la sesión inaugural del Taller de Desarrollo de Nova llevado a cabo en marzo 2012, se planteó la necesidad de reducir la cantidad de paquetes disponibles en el repositorio, proponiendo su posible reestructuración en cuanto a las aplicaciones a almacenar.

Dicha situación se plantea, pues según las experiencias de los usuarios, no siempre se utilizan todos los paquetes del repositorio. Para ello se propuso crear mecanismos como encuestas a la comunidad de software libre que arrojen reportes de las aplicaciones más usadas para determinar cuáles son los paquetes imprescindibles que debe tener un repositorio. Se concluyó entonces con la idea crear un repositorio reducido que contenga dichos paquetes básicos disponibles para los usuarios que lo deseen y así mejorar su tiempo de descarga por la red. Del mismo modo se mantendría el repositorio original para no limitar la disponibilidad de aplicaciones a la sociedad cubana [13].

1.2.2 Disminución de espacio por compresión.

El tamaño de un repositorio es proporcional a la suma de la cantidad de paquetes que contiene, pues los demás ficheros que almacena poseen un tamaño despreciable con respecto al total:

Tamaño de repositorio = Σ tamaño de los paquetes

La fórmula indica que reducir el tamaño de los paquetes que contiene un repositorio, puede crear efectos positivos en su espacio resultante y si dicho volumen es menor, mejor será entonces su transmisión por la red.

10 Siglas de Local Area Network, Red de área local.

Precisamente un paquete .deb contiene en su interior tras la compilación tres componentes, dos de ellos control y data, están comprimidos generalmente en tar.gz, aunque existen otros formatos de compresión soportados por la distribución que podrían ser más favorables.

A continuación se realiza un estudio estadístico con los formatos de compresión más usados y soportados en la distribución de Nova, para determinar cuál de todos sería el propicio para la compresión de los paquetes. Para diagnosticar tal herramienta se tomaron en cuenta las variables **tamaño de compresión** con respecto al tamaño original, **tiempo en la compresión** y el no menos importante, el **tiempo en la descompresión**, variable fundamental para el usuario final.

Debido a que la población de paquetes es compleja de analizar por ser infinita, se determinó seguir una técnica de muestreo mixto o complejo, en donde se seguirá un muestreo aleatorio simple y sistemático en un período de diez días. De cada experimento se pretende, luego de su estimación puntual, obtener las medias de los compresores y determinar los porcentajes de promedios obtenidos para considerar el más favorable.

De esta forma se espera obtener el resultado, bajo las mismas condiciones tecnológicas, de las comparaciones con respecto al formato tar.bz que es el que se utiliza actualmente en la compresión interna de un paquete binario luego de su compilación.

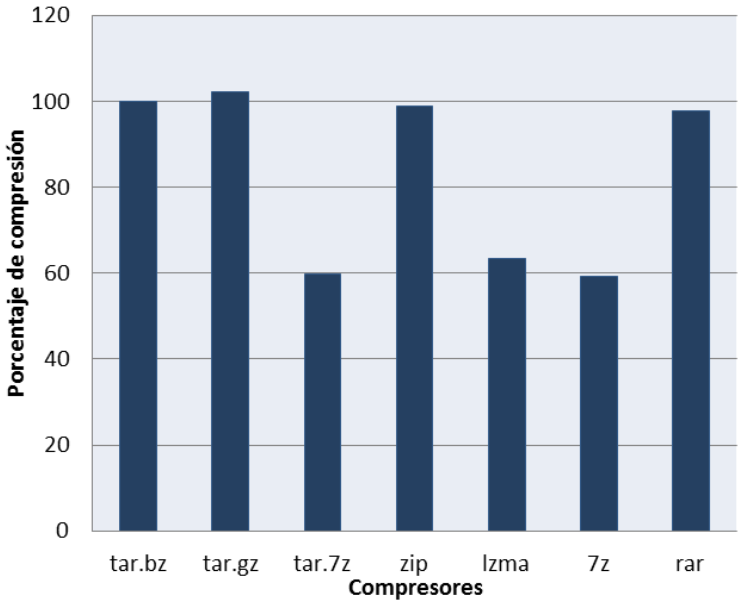
1.2.2.1 Evaluación en tamaño de compresión.

La tabla 1.1 contiene la relación de la media de los tamaños comprimidos (Kb) en todas las muestras tomadas sistemáticamente y su comparación con respecto a tar.bz, así como sus respectivos porcentajes.

	tar.bz	**tar.gz**	**tar.7z**	**zip**	**lzma**	**7z**	**rar**
X	110,5	113	66	109,2	70,1	65,6	108
%	100	102,2	59,7	98,8	63,4	59,3	97,7

Tabla 1.1: Comparación de los tamaños de las compresiones para cada formato.

Tras los resultados mostrados en la tabla 1.1, se presenta la siguiente gráfica para un mejor entendimiento.



Gráfica 1.1: Porcentaje de compresiones para cada formato.

En la gráfica 1.1 se muestran los porcentajes de la media de los tamaños tras la compresión de la muestra para cada formato, ilustrando que se obtuvo una mejor compresión con 7z, tar.7z y lzma respectivamente.

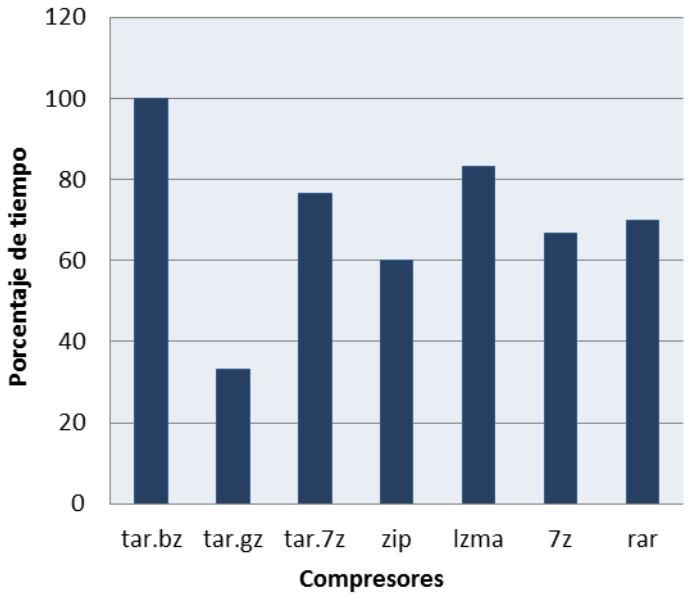
1.2.2.2 Evaluación en tiempo de compresión.

En el siguiente análisis de la tabla 1.2, se realiza la comparación de los **tiempos de compresión** de las diferentes herramientas, tomando en cada caso la media de los tiempos en segundos que tarda en comprimir cada una de ellas con la misma muestra y los por cientos que representa.

	tar.bz	tar.gz	tar.7z	zip	lzma	7z	rar
X	3	1	2,3	1,8	2,5	2	2,1
%	100	33,3	76,6	60	83,3	66,6	70

Tabla 1.2: Comparación de tiempos en segundos de compresión para cada formato.

Según los resultados de los porcentajes de los tiempos en segundos que le tomó a cada herramienta realizar sus compresiones, se obtiene la gráfica siguiente:



Gráfica 1.2: Porcentajes de tiempos totales de compresión para cada formato.

Analizando los por cientos de los promedios de tiempo de compresión para cada formato de la gráfica 1.2, se evidencia que de todos, los que tardan menos en realizar su trabajo son tar.gz y zip. Sin embargo, de las tres herramientas que mejor reducen el tamaño de compresión de archivos, 7z es la que mantiene menor tiempo en la compresión.

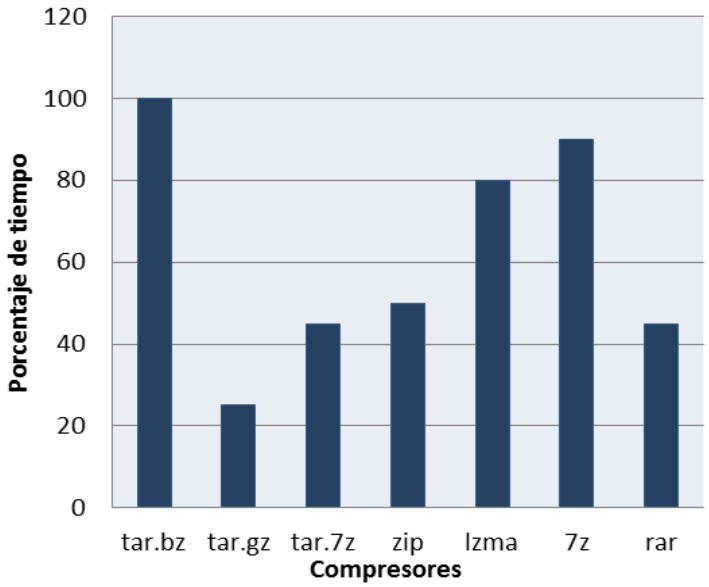
1.2.2.3 Evaluación en tiempo de descompresión.

De igual forma en que se realizaron las comparaciones anteriores y tomando la misma muestra, se examinan en la tabla 1.3, los porcentajes de las medias de tiempos en segundos que tardan en **descomprimirse** los paquetes de la muestra.

	tar.bz	**tar.gz**	**tar.7z**	**zip**	**lzma**	**7z**	**rar**
X	2	0,5	0,9	1	1,6	1,8	0,9
%	100	25	45	50	80	90	45

Tabla 1.3: Comparación de tiempos en segundos en descomprimir por cada formato.

Tales resultados se muestran con más detalles en la gráfica 1.3:



Gráfica 1.3: Porcentajes de tiempos de descompresión por cada formato.

Considerando los porcentajes de los promedios de tiempos de descompresión, se demuestra que la mayoría de los formatos analizados tienen un elevado grado de rapidez en dicha operación, exceptuando el caso de **tar.bz** y **lzma**, visto este último como uno de los mejores compresores en cuanto a tamaño de la compresión en los estudios anteriores. Sin embargo, de los tres formatos que mejores porcentajes tienen en los tamaños de las compresiones, **tar.7z**, es el que ocupa menos tiempo en la descompresión.

1.2.2.4 Conclusiones del caso experimental.

Según las valoraciones hechas para la muestra aleatoria seleccionada y teniendo en cuenta las variables analizadas, se concluye con la siguiente observación:

Atendiendo a la necesidad de reducir el volumen del repositorio, se establece una prioridad entre las variables: primero que el tamaño de compresión de los paquetes sea mínimo y luego que los tiempos requeridos para descompresión, que acelera el proceso de instalación para el usuario final, y la compresión sean igual de pequeños; todo en este orden.

Por tal motivo se tienen de los tres mejores resultados en la reducción de tamaño: 7z, tar.7z y lzma, una selección de 7z y tar.7z, pues poseen los mejores tiempos en la compresión y descompresión de los tres formatos mencionados.

Hasta este punto, cabe entonces una comparación más detallada solamente entre tar.7z y 7z, donde descomprimiendo 7z es más rápido un 10% más que tar.7z, mientras que descomprimiendo tar.7z es más rápido un 45% más que 7z.

Sin embargo, tar.7z tiene una ventaja distintiva y es que emplea 7z archivándose con tar, haciendo que el contenido no pierda sus permisos y propiedades, situación que no maneja 7z y como además las diferencias entre los resultados de las herramientas no son las mas alarmantes, tar.7z es a consideración del caso experimental, la mejor opción para aplicar en la compresión de archivos internos tras la compilación de paquetes y poder tener los tamaños del paquete final lo más reducido posible.

1.2.2.5 7z o 7zip.

Es un programa creado por Igor Pavlov en el 2006 bajo la licencia GPL utilizado para archivar, comprimir y descomprimir archivos con la extensión .7z o .7zip a un alto grado de compresión y sin pérdida de datos [14]. Para instalarlo es necesario contar con el compresor y descompresor en modo consola p7zip disponible para las distribuciones GNU/Linux que en realidad es la versión en línea de comandos de 7zip [15].

Entre las características de 7z más importantes está que es compatible con formatos de compresión: zip, gzip, bzip2, tar, tiene arquitectura abierta, trabaja con enlaces simbólicos y tiene fuerte cifrado AES[11]-256. Funciona además con archivos de tamaños hasta 16000000000 GB y como ha quedado demostrado tiene alta relación de compresión en donde utiliza el algoritmo LZMA, versión mejorada y optimizada del algoritmo LZ77[12] [16].

1.2.3 Disminución de espacio utilizando parches de paquetes Debian.

El proceso de cálculo de un parche o delta, se trata también por los términos de compresión delta o compresión diferencial, sus aplicaciones pueden favorecer la rapidez en las descargas en las actualizaciones de los paquetes desde los repositorios a través de la red ya que minimiza el volumen de la transferencia en las actualizaciones, de ser paquetes completos a parches solamente [12].

11 Advanced Encryption Standard (AES) es un tipo de encriptación para datos electrónicos.

12 Algoritmo sin pérdida se utiliza cuando la información a comprimir es crítica y no se puede perder información [17].

1.2.3.1 Aplicaciones de los parches.

La utilización de la compresión diferencial es ventajosa pues en la práctica resuelven importantes problemas como son principalmente [18]:

- Explorar diferencias de archivo:
  Las herramientas deltas se pueden utilizar para visualizar diferencias entre distintos documentos de diferentes formatos o programas.

- Control de versiones:
  Las técnicas de compresión delta son utilizadas en los sistemas para mantener el historial de revisiones de proyectos de software y otros documentos en el menor espacio posible de almacenamiento.

- Morosa distribución de software a través de protocolos html y ftp:
  Los parches de software pueden ser transmitidos a través de una red con el fin de actualizar paquetes de software instalados de forma más rápida que si se transmite el original. De esta forma existe con su utilización un mejoramiento en el rendimiento de protocolos html y ftp en la distribución de software.

1.2.3.2 Caso experimental.

En la tesis de Colin Percival[13]: ”Naıve Differences of Executable Code”, según la tabla experimental en la que lleva el estudio entre diecinueve paquetes binarios (Figura 1.1), el autor llega a la conclusión de que entre las herramientas para el cálculo de parches a paquetes binarios: xdelta, .RTPatch, Exediff y bsdiff; el orden en eficiencia en cuanto a por ciento de compresión es Exediff, luego bsdiff, .RTPatch y en último lugar Xdelta, teniendo en cuenta que tanto exediff como .RTPatch son herramientas para plataformas Windows [19].

13 Desarrollador de bsdiff.

Program	Uncompressed	Compressed	Xdelta	.RTPatch	Exediff	BSDiff
alto: identical binaries	466944	148024	137	n/a	155	142
alto: gcc -O2 → gcc -O3	466944	148024	78390	34755	20793	33633
alto: changed reg. alloc.	450560	148024	97923	34571	15845	23246
alto: extra printf	466944	148024	50613	7524	6237	6299
agrep: 4.0 → 4.1	262144	114388	14631	5910	3531	6066
glimpse: 4.0 → 4.1	524288	222548	109252	37951	23200	31720
glimpseindex: 4.0 → 4.1	442368	193883	98632	25764	18473	21559
wgconvert: 4.0 → 4.1	368640	157536	75230	20712	15688	15806
agrep: 3.6 → 4.0	262144	114502	80346	58124	41554	53490
glimpse: 3.6 → 4.0	524288	222178	177434	140549	104350	130210
glimpseindex: 3.6 → 4.0	442368	193892	144927	105510	79085	97782
netscape: 3.01 → 3.04	6250496	2396661	1100430	351759	284608	302431
gimp: 0.99.19 → 1.00.00	1646592	642725	463878	301879	185962	284278
iconx: 9.1 → 9.3	548864	233056	139409	51195	38121	44961
gcc: 2.8.0 → 2.8.1	2899968	708301	549250	140284	76072	121371
rcc (lcc): 4.0 → 4.1	811008	221826	889	265	303	289
apache: 1.3.0 → 1.3.1	679936	180708	111421	48033	40460	38278
apache: 1.2.4 → 1.3.0	671744	179369	191920	216867	227233	180981
rcc (lcc): 3.2 → 3.6	434176	155090	84456	34098	22019	33136
Average Compression	100%	35.5%	19.4%	9.8%	7.3%	8.6%

Table 1: Sizes of updates produced by bzip2, Xdelta, .RTPatch, Exediff, and BSDiff

Figura 1.1: Tabla comparativa de Colin Percival.

Se llega con el estudio de Percival a otro análisis interesante y es que para una actualización es más favorable descargar parches que la compresión de la nueva versión del paquete, es decir, tiene mejor resultado para la actualización disponer de parches utilizando cualquier herramienta de las mencionadas anteriormente, que disponer del paquete comprimido normalmente, que si bien reduce los tamaños de almacenamiento de los paquetes para la actualización es más óptimo contar con los parches.

Es válido destacar que las herramientas vistas en la comparación no son los únicos métodos de creación de parches. Existen otros como el básico diff/patch para paquetes fuentes, el plugin presto para los usuarios de Fedora con paquetería .rpm y debdelta para las distribuciones de Debian y Ubuntu, este último liberado por primera vez en el 2006 para paquetes .deb.

Teniendo en cuenta que la realización de la investigación de Percival fue desarrollada en el 2003 y que debdelta aparece en el 2006, se lleva a cabo un nuevo estudio para comprobar el criterio en cuanto a la mejor herramienta para la creación de parches más reducidos, reutilizando la comparación analizada.

Se procede entonces a comparar debdelta con las herramientas adaptables a paquetería .deb comprobadas anteriormente y así determinar la que reduce más los parches. Además se establece una compa-

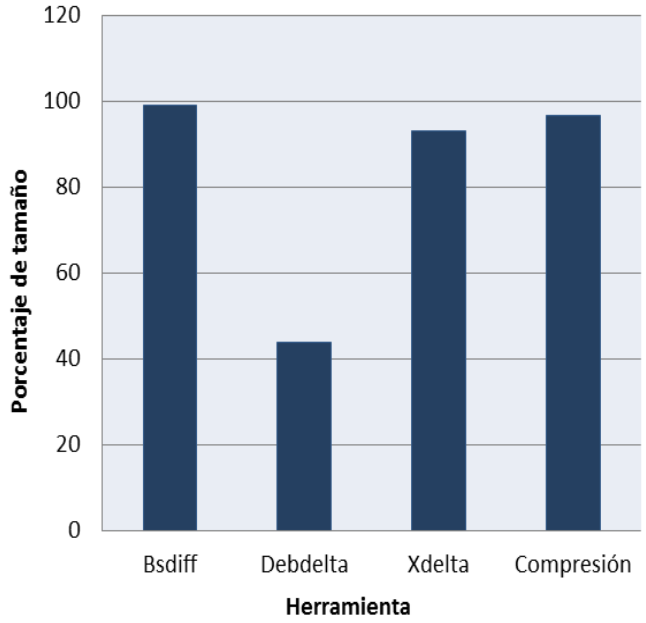
ración del tamaño de las nuevas versiones de los paquetes comprimidos con tar.7z y el tamaño de los parches. De esta forma se puede apreciar para la actualización de paquetes el espacio que se consigue ahorrar en la descarga de parches con respecto a la descarga de nuevas actualizaciones aunque estén comprimidas.

Como variable o punto de comparación se establece el **tamaño de los parches** producidos entre versiones de aproximadamente cuarenta paquetes binarios con cada una de las herramientas en estudio.

	Actualización	Compresión	Bsdiff	Debdelta	Xdelta
Total	7987,2	7731,5	7907,3	3514,3	7428,9
%	100	96,7	98,9	43,9	93,0

Tabla 1.4: Comparación de tamaños de parches herramientas deltas, paquete original y su compresión.

Tras el análisis comparativo mostrado en la tabla 1.4 de las diferentes herramientas en cuanto al tamaño de los parches que se producen entre mismos paquetes de diferentes versiones, se puede determinar que la herramienta debdelta, es la que de las revisadas en la bibliografía, mayor reducción en sus parches arroja. Para más detalle se ilustra la gráfica 1.4:



Gráfica 1.4: Tamaños de parches para cada herramienta delta y compresión del paquete.

1.2.3.3 Conclusiones de la comparación.

Tras la apreciación de los porcentajes de los promedios de tamaños de las actualizaciones luego de la aplicación de cada herramienta, se puede confirmar que debdelta es la que produce tamaños de parches notablemente más pequeños. Aunque no es una herramienta independiente, sino una gama de aplicaciones, se adapta perfectamente a la necesidad que se tiene y por tanto se considera la herramienta idónea para una posible agilización en las descargas de actualizaciones de repositorios en Nova.

Otro de los análisis importantes es la reafirmación de que resulta más factible utilizar parches para actualizaciones que la descarga de una nueva versión del paquete normalmente, pese a que se encuentren comprimidas.

1.2.3.4 Debdelta.

Debdelta es un conjunto de aplicaciones para el trabajo con parches con extensión .debdelta que solo trabaja para paquetes binarios en sistemas Debian, creado por Andrea Mennucci bajo la licencia GNU[14], liberado en mayo del 2006 [20].

La herramienta trabaja con un repositorio de actualizaciones y cuenta con programas fundamentales como: debdelta que calcula la diferencia entre binarios y lo guarda en un archivo delta, debpatch que reconstruye una nueva versión de un paquete utilizando el parche y debdelta-upgrade que descarga deltas necesarias en los ordenadores de los usuarios finales**.**

Ventajas notables que añade debdelta es que cuenta con la restricción de que si el parche contiene una diferencia de un 70% de una versión con respecto a otra, entonces no se construye el delta, lo que supone una utilidad provechosa para el tamaño final de un repositorio de actualizaciones.

Cuenta con la opción distintiva de dar la posibilidad de usar conjuntamente con su compresión, algoritmos de distintas herramientas como xdelta, xdelta3 y bsdiff. Esta característica hace que la ejecución de debdelta se alentase un poco más de lo normal, pero para contrarrestar, cuenta con la alternativa de poder especificar el máximo de memoria del que va a usar bsdiff o xdelta, aunque si los usa todos tiene mejor compresión en los parches resultantes [20].

14 Licencia Pública General de GNU, es una licencia creada por la Free Software Foundation en 1989 y está orientada a proteger la libre distribución, modificación y uso de software.

Seguridad en debdelta:

Entre sus funcionalidades cuenta al igual que xdelta, con la verificación de suma MD5[15] y se le añade GnuPG[16] y SHA1[17] a las opciones que se le pueden especificar [21].

Un paquete Debian que se crea con en el delta descargado, es byte por byte idéntico al original, de modo que el soporte de autenticación criptográfica puede ser utilizado para afirmar que se puede confiar en que se instale un paquete original, a pesar de ser creado a partir de un parche.

Forma de uso de Debdelta:

Sistemas libres como Ubuntu y Debian tienen desarrollados repositorios de parches de código binario por las ventajas que supone en la rapidez en la descarga de las actualizaciones para sus usuarios finales y del ahorro de espacio en los propios repositorios.

El marco de trabajo que se utiliza para explotar la herramienta debdelta es tener un repositorio de parches que contendría los deltas de los paquetes del repositorio original y al cual accederían los usuarios para efectuar sus actualizaciones.

Para cumplir la estrategia descrita, se debe contar con un servidor que entre los recursos que puede almacenar, estén los repositorios de parches binarios. Cada parche es nombrado según una nomenclatura establecida por la distribución especificando el nombre del paquete, el par de versiones para las que está hecha la diferencia, donde pueden existir tantos parches como pares de versiones del paquete existan y por último la especificación de la arquitectura. Es por tal motivo que la única diferencia de la estructura entre repositorios de paquetes de software y de parches, está en que estos últimos no necesitan índices, pues en el algoritmo de debdelta la información que debería tomar de un índice la toma del mismo nombre del parche.

De forma paralela, el usuario final en su ordenador puede descargar desde el servidor del repositorio de parches, los deltas con la orden debdelta-upgrade, que es la encargada de descargar tanto los nuevos

15 Algoritmo de reducción criptográfico

16 Herramienta que permite encriptar y firmar sus datos por contar con un versátil sistema de gestión de claves, lo que permite la seguridad en los parches creados

17 Sistema de funciones hash criptográficas, que se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro, archivo, etc. Además de resumir o identificar un dato a través de la probabilidad, utilizando una función hash o algoritmo hash.

parches de los paquetes instalados que se registran en la caché, como descargar del repositorio habitual los paquetes binarios a actualizar para los que no se haya podido realizar un parche. Luego de la descarga, se aplican los deltas descargados a las viejas versiones instaladas y se crean entonces los nuevos paquetes .deb correspondientes.

De forma más explícita, cuando el usuario necesite actualizar un paquete, se recomienda utilizar inicialmente apt-update para que en la caché esté registrado la información de todos los paquetes disponibles para actualizar, lo cual serviría como guía a debdelta para buscar los parches necesarios. Luego la utilización de debdelta-upgrade para descargar todos los deltas disponibles y los aplicaría construyendo los nuevos paquetes .deb que almacena en la cache de apt, en caso de ejecutarse como root, o en el archivo temporal tmp, si se ejecuta como usuario sin privilegios. Una vez realizada esta descarga, con la llamada a apt-upgrade, ya se dispondrían de los nuevos paquetes para la actualización y los instalaría desde el mismo sistema sin necesidad de acceder al repositorio, siendo esta manera mucho más rápida que si no se hubiesen descargado antes los parches [20].

Dicha razón es la que hace que según el sitio oficial de la herramienta se recomiende en temas de actualizaciones utilizar la herramienta de la forma siguiente:

```
root# apt-update debdelta-upgrade apt-upgrade
```

1.3 Conclusiones del capítulo.

El capítulo que concluye, ha caracterizado aspectos fundamentales del estudio del arte que permite comprender posibles soluciones para garantizar el cumplimiento del objetivo principal de la investigación:

- Nova es una distribución GNU/Linux que utiliza paquetería de Ubuntu y que adopta los conceptos y estructuras relacionadas con los sistemas de repositorios. De igual forma usa los mismos gestores de paquetes .deb, donde los que trabajan a más bajo nivel son dpkg y apt, siendo apt el que resuelve mejor el trabajo con dependencias y no trabaja localmente como solo hace dpkg.
- Según la forma en que distribuyen los repositorios en las instituciones cubanas para agilizar las descargas, el método de establecer uno localmente, es el más adaptable a las condiciones tecnológicas; así como disponer de un repositorio reducido con los paquetes de software fundamentales para lograr un menor tiempo en su transferencia.

- Las herramientas para la compilación de paquetes .deb, dentro de su código fuente tienen la posibilidad de especificar en qué formato van a estar comprimidos sus paquetes fuentes y binarios. Tras el análisis relacionado con los métodos de compresión, el que disminuye mejor los tamaños de comprimidos, los tiempos en compresión y descompresión es tar.7z siendo el formato idóneo para que la descarga por la red de paquetes compilados en .deb reduzca en tiempo y congestión de datos por la red, agilizando las operaciones de instalación y actualización de paquetes desde los repositorios.
- El trabajo con los parches de paquetes binarios, tomando en cuenta las aplicaciones que tienen y las ventajas que supone, es considerado como solución posible para lograr una mayor rapidez en la descarga en las instalaciones y actualizaciones de paquetes en Nova. Al estudiar las herramientas relacionadas con la creación de parches, según las características, experiencias en distribuciones y resultados en casos experimentales, la más adecuada es debdelta.

Con el empleo de las propuestas analizadas, se estima que descargar o actualizar repositorios de la distribución con una conexión por el canal nacional de hasta 2Mbps se puede aspirar a los resultados siguientes:

Tamaños de repositorios (Gb)	Tamaños resultantes modificando el compresor en la compilación (Gb)	Tamaños resultantes utilizando la modificación en la compilación y los parches en la actualización (Gb)
20	11,9	5,2
32	19,1	8,3
40	23,8	10,4
60	35,8	15,7
Por ciento: 100,00%	***59,6%***	***30,1%***

Tabla 1.5: Evolución de posibles tamaños resultantes aplicando tar.7z en la compilación y parches para las actualizaciones.

Tal análisis indica que utilizando las propuestas de la modificación de compresores en el proceso de compilación, el tamaño del repositorio resultante puede ocupar solamente el 59,6 % de lo que actualmente ocupa. Si se le adiciona la integración de parches para las actualizaciones se descargarán entonces el 30,1% de los Mbyte del total de las actualizaciones que se descargan comúnmente. Dicho hecho indica que si se reduce el tamaño del repositorio, se favorecerá a disminuir su tiempo de descarga.

CAPÍTULO 2: “Propuestas para la disminución de tiempos de descarga del repositorio de Nova”.

Tras el análisis de los aspectos tratados en el capítulo 1, se tiene claro que todas las posibles soluciones están encaminadas a reducir en tiempo las descargas de paquetes por la red, dígase para usuarios que deseen obtenerlos manualmente o para aquellos que lo descarguen a través de gestores de paquetes para la instalación o actualización.

Hasta este punto se hacen posibles tres propuestas fundamentales que son posibles aplicar en Nova:

- **Estructural:** Tener los repositorios de la distribución en una red local para facilitar sus descargas, contando con la posibilidad de agilizarlas utilizando el repositorio minimizado.
- **Compilación:** Compilar los paquetes con el método de compresión más favorable en los fuentes y binarios.
- **Parches:** Instaurar repositorios deltas para descargar actualizaciones de paquetes binarios a través de parches.

Vale apuntar que al mismo tiempo que las propuestas de soluciones disminuyen la demora en la propagación de paquetes transmitidos por la red, se está ejecutando otra acción favorable y se trata en este caso de la disminución del espacio de almacenamiento en el servidor.

2.1 Propuesta estructural.

Siguiendo las experiencias de expertos desarrolladores de la distribución de Nova, se recomienda la siguiente propuesta:

Las instituciones cubanas continuarán con descargas del repositorio para centralizarlo en una red local, lo cual permite una conexión a 100 Mbps y que las actualizaciones de paquetes se realicen por personal autorizado desde el repositorio oficial de la distribución. Dicha actualización o posesión de un nuevo paquete, se pretende que se obtenga de forma más rápida por la futura integración de parches en los repositorios y las nuevas formas de compilación que se expondrán en las próximas secciones.

Para lograr mayor agilidad en este proceso y en vista a que inicialmente los clientes tengan los paquetes indispensables para la distribución, se sigue la estrategia de crear un repositorio minimizado, de esta

manera se descargarían con menor retardo por la reducción de tamaño que supone. Es por eso que se propone seleccionar todos los paquetes que estén instalados por defecto en las líneas de desarrollo: Nova Escritorio (E), Nova Ligero (L) y Nova Servidores (S) como se observa en la figura 2.1, de esta forma se garantiza que se tengan los paquetes básicos para el usuario final, manteniendo el repositorio original de paquetes para el caso de necesitar alguno en específico.

Los paquetes del repositorio minimizado (PRM) se determinan siguiendo la fórmula que a continuación se muestra:

PRM=L U E U S

Dicha relación se representa con el siguiente diagrama de conjuntos:

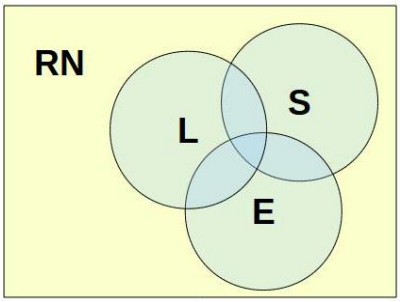


Figura 2.1: Representación de paquetes de distribuciones Nova.

Como se observa en la ilustración, donde el universo son todos los paquetes del Repositorio de Nova (RN) y los conjuntos los paquetes de bases de las líneas de desarrollo de Nova que hacen el repositorio minimizado, se descifra la solución final pretendiendo que en dicho repositorio se encuentre la unión de todos los paquetes de cada conjunto, lo que garantiza que se almacenen todos los paquetes básicos de Nova asegurando al mismo tiempo la condición de que ninguno se repita.

Una vez analizado lo que se desea obtener, se procede entonces a la manera de crear la lista de dichos paquetes. Para ello se tomó cada una de las imágenes de discos de instalación de las distribuciones de Nova (ISO[18]) creados por el equipo de desarrollo para obtener de ellos el listado de paquetes que están por defecto en el sistema. Luego es necesario crear, a partir de los paquetes seleccionados, un repositorio que sería el minimizado. El proceso a seguir es el que se muestra a continuación en el diagrama 2.1:

18 Archivo donde se almacena una copia o imagen exacta de un sistema de ficheros.

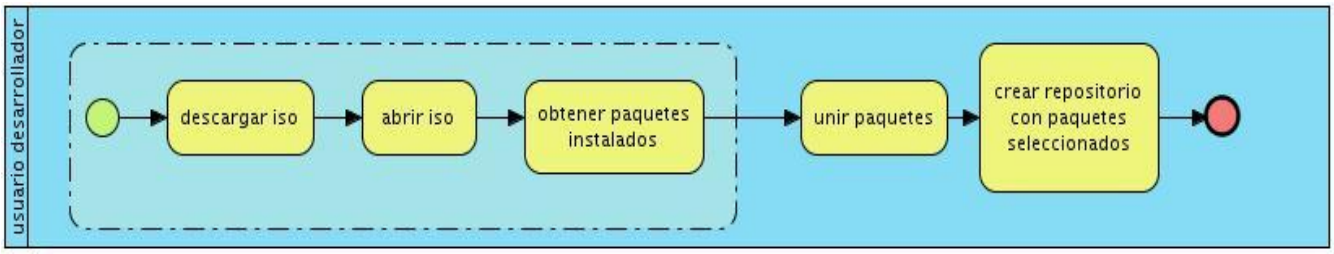


Diagrama 2.1: Proceso para crear repositorio minimizado.

En el caso de la acción de unir paquetes, se procese a ejecutar el script en python[19] del Anexo 2, para luego pasar a la acción de crear el repositorio, a través del script en bash[20] descrito en el Anexo 3. Vale aclarar que para las acciones señaladas en el recuadro discontinuo, se ejecuta para cada una de las tres líneas de Nova descritas. El repositorio minimizado contendría los paquetes almacenados en componentes en dependencia de donde estén ubicados los paquetes en el repositorio original de la distribución.

2.2 Utilización de la mejor compresión en la compilación de paquetes con dpkg.

Dpkg es un gestor de paquetes para Debian, pero que ha sido usado igualmente en otras distribuciones como Ubuntu y Nova. Entre sus objetivos claves, como gestor de paquetes están el de instalar, borrar, administrar y construir paquetes .deb.

Precisamente la funcionalidad de dpkg para automatizar el proceso de construir sus paquetes es dpkg-buildpackage, que tiene la responsabilidad de crear, a partir del paquete fuente y su archivo .dsc, el nuevo paquete fuente comprimido en los formatos de tar.gz o tar.bz2, el archivo .changes con las diferencias y el paquete.deb con sus respectivos archivos internos comprimidos en tar.gz.

Como se analizó a inicios del capítulo, una de las propuestas para la reducir la demora en las descargas tiene que ver específicamente con variar la forma en que se empaquetan los paquetes para hacer que sus archivos internos se compriman de manera más reducida luego de la compilación. Es por eso que se procede a la modificación de funcionalidades relacionadas con la compresión dentro de dpkg.

19 Lenguaje de programación aprobado por licencia de código abierto, soportado para la plataforma Unix.

20 Programa de interpretación de comandos para Unix.

Con la modificación del paquete fuente de dpkg, se está aportando la misma modificación para todas las herramientas usadas para la compilación que utilizan dicho gestor, como son apt, pbuilder y debuild mayormente utilizadas en distribuciones de paquetería .deb.

Para asegurar que durante la ejecución de dpkg-source y dpkg-buildpackage, se construyan paquetes fuentes y binarios respectivamente con el método de compresión tar.7z, se hizo necesario modificar algunas de las variables y métodos relacionados con la compresión, descompresión y construcción de paquetes fuentes y binarios, así como opciones de ficheros debian del código fuente de dpkg. Todos los cambios se almacenan en un parche que se le aplicará al paquete dpkg en sus futuras versiones para que cumplan con la característica específica para el sistema Nova (Anexo 4).

2.3 Utilización de la mejor compresión en la descarga de paquetes a través de apt.

El gestor de paquetes apt cuenta con el fichero de configuración principal apt.conf que no se crea por defecto en el momento de la instalación, pero que es muy útil en cuanto a tener determinadas opciones que serán uniformes para todas las utilidades de apt del sistema donde se configure. Apt.conf está estructurado en forma de árbol con las opciones organizadas en grupos funcionales donde uno de ellos es Acquire, que contiene un grupo de opciones que controlan la descarga de paquetes. Precisamente la opción Order define en qué orden Acquire descargará los ficheros comprimidos según la preferencia especificada, es decir, se garantizará la descarga de archivos con el primer tipo de formato especificado y en caso de error se procederá a descargar los demás archivos en los formatos que le siguen en la lista.

En aras de reducir los tiempos en las descargas de apt se tendría la siguiente línea de configuración en apt.conf de acuerdo al orden de preferencia de los formatos de compresión analizados en el capítulo anterior [22].

```
Acquire::CompressionTypes::Order {“tar.7z”;”7z”;”lzma”};
```

Es válido aclarar que para cada formato de compresión, debe existir en el mismo fichero de configuración, pero en el grupo de opciones de Directorios, la especificación de la ubicación de los programas de cada uno de ellos en el sistema:

```
Dir::Bin::tar ”/bin/tar”;
Dir::Bin::7z ”/usr/bin/7z”;
Dir::Bin::lzma ”/usr/bin/lzma”;
```

De esta forma, se garantiza que apt priorice las descargas de ficheros con tar.7z, 7z y lzma, asegurando que las descargas se hagan en menos tiempo.

2.4 Integración de repositorio de parches a paquetería .deb en Nova.

Para la integración de parches al sistema de trabajo en Nova se debe definir en primer lugar una estructura de parches para paquetes binarios específicos para la distribución, su forma de almacenamiento en el repositorio especial para almacenar los deltas y la manera en que se trabajará. Luego, buscar la vía para hacer que los gestores de paquetes que actualmente utiliza Nova tengan una integración con la herramienta seleccionada para los parches y dicho gestor a su vez con los repositorios de parches para localizar las actualizaciones.

2.4.1 Marco de trabajo para parches.

Tras el previo análisis realizado, se determina que el marco de trabajo que se seguirá es el que se expone a continuación en aras del mejoramiento de agilizar el proceso de instalación y actualización de paquetes del sistema operativo Nova:

Se comenzará partiendo del requisito inicial de que debe existir un repositorio de parches de paquetes binarios con una estructura específica, que contendrá los parches de la forma en que se describirán en la sección 2.4.1.1, tomando como referencia la forma que Debian y Ubuntu utilizan debdelta.

El repositorio de parches estará configurado en el archivo de configuración de debdelta que al mismo tiempo será entendido por el gestor de paquetes seleccionado, de forma tal que el usuario cuando de la orden de instalar o actualizar cualquier aplicación, se establezcan dos situaciones fundamentales que tendría en cuenta el gestor: actualizar las dependencias de un paquete a instalar o actualizar por completo un paquete, siempre procurando que dichas actualizaciones estén disponibles en parches.

2.4.1.1 Propuesta de repositorios de parches para Nova.

Estructura de los parches de código binario para Nova:

Los parches debdeltas no contienen descripción ni dependencias como en el caso de los paquetes, por tal motivo se sugiere que los parches que genere debdeltas para Nova tengan la misma estructura que para Ubuntu y Debian: **P_O_N_A.debdelta**, siendo P el nombre del paquete en cuestión, O el número que indique la vieja versión con las revisiones, N el número que indique la nueva versión con las revisiones y finalmente A que sugiera la arquitectura para la cual se puede aplicar el parche.

Con esta información la forma de hallar la dirección de cualquier parche puede deducirse algorítmicamente. Cuando el usuario ejecute una actualización, la manera de encontrar el parche que le corresponde, en caso de que exista es muy sencillo, basta con dirigirse hasta la ruta del repositorio que se establece en el parámetro url en la configuración de debdelta ubicado en **/etc/debdelta/sources.conf** y luego encontrar el parche en el repositorio accedido en la ubicación: **url configurada/pool/inicial de P/P/P_O_N_A.debdelta**, sin necesidad de tener el fichero Package.gz para saber si existe el paquete a descargar.

Estructura de repositorio de parches para Nova:

Se propone la creación de un repositorio donde se almacenarán los debdeltas, exceptuando aquellos que sean superiores al 70% del tamaño original del paquete y los menores de 10kb. De esta forma se consigue el objetivo fundamental de tener menos espacio de almacenamiento para el nuevo repositorio de actualizaciones.
El repositorio debe ser similar al que tiene actualmente Nova, con la diferencia que en lugar de almacenar los .deb almacena los .debdelta, no contendrá el archivo Packages.gz y solamente contendrá el directorio pools, donde se organizarán por subcarpetas, nombradas por letras que indican la letra inicial de los paquetes, los ficheros que le correspondan con dicho carácter señalando el nombre del paquete y dentro, sus respectivos parches.

Según lo explicado puede darse la siguiente situación, si existe un parche foobar_1_2_all.debdelta, tendrá entonces su ubicación en la dirección pool/main/f/foobar/foobar_1_2_all.debdelta y cada vez que se almacene una nueva versión de foobar en el repositorio de los .deb, en el repositorio de los .debdeltas se generará en esa misma dirección otros parches con las diferencias entre las versiones que se gene-

raron del mismo paquete, siempre y cuando existan más de una y conteniendo siempre las diferencias de los parches con respecto a la última versión.

Siguiendo con la estructura planteada se puede tener la situación que sigue, teniendo en cuenta lo que debe ocurrir en los repositorios cuando se van almacenando nuevos paquetes .deb:

Tipos de repositorios	1ro de septiembre	2 de septiembre	3 de septiembre
Paquetes binarios en pool/main/f/foobar/	foobar_1_all.deb	foobar_2_all.deb	foobar_3_all.deb
Parches de paquetes binarios en pool/main/f/foobar/	No contendrá parches, solo existe una versión del paquete.	foobar_1_2_all.debdelta	foobar_1_3_all.debdelta foobar_2_3_all.debdelta

Tabla 2.1: Funcionamiento del almacenamiento de parches según los paquetes añadidos al repositorio.

De la forma anteriormente descrita se tiene que ambos repositorios, tanto el de parches como el de paquetes binarios, van a tener la misma estructura dentro de pool, con la diferencia de lo que contendrá cada uno, de esta forma, el usuario puede encontrar tanto las versiones del paquete binario como los parches para el mismo paquete, ambos en la misma dirección, pero de sus respectivos repositorios.

Siguiendo con el planteamiento, el flujo de trabajo debe determinar que si un usuario tiene instalada la versión 1 de foobar y actualiza el paquete el día 2 de septiembre y luego el 3 de septiembre, se debe aplicar el foobar_1_2_all.debdelta el día 2 y luego foobar_2_3_all.debdelta del día 3; pero si por otra parte actualiza solo el día 3 de septiembre, se debe aplicar foobar_1_3_all.debdelta.

2.4.1.2 Propuesta de integración de herramienta para deltas con gestor de paquetes.

Selección del gestor de paquetes a modificar:

Los gestores de paquetes .deb que utiliza Nova son una pequeña cadena: comienza con dpkg, le sigue apt, luego continúan aptitude y synaptic con sus nuevas características e interfaces, donde cada una utiliza la anterior en su cadena y las mejora.

La herramienta que se utilizará para la modificación de su código fuente y que contendrá la integración con los parches será apt pues es el gestor de paquetes de más bajo nivel que utiliza las funcionalidades de dpkg incorporándole el trabajo con repositorios externos. Apt es la base de los otros gestores como aptitude y synaptic donde el primero trabaja con dependencias huérfanas e interfaz ncurses y el segun-

do, que por tener interfaz gráfica más cómoda, la mayoría de usuarios la utiliza. Al tener ambas herramientas la funcionalidad de apt integrada ya tendrían incorporado por transitividad el entendimiento de los parches.

Selección de herramienta a modificar para crear deltas:
La herramienta que se seleccionó para la creación de los parches, tras el análisis del capítulo 1, es debdelta. El programa tiene entre sus archivos fuentes el llamado debdelta.py, donde se encuentra el desarrollo de todas las acciones de la herramienta y al que apuntan en forma de enlace simbólico cada una de las funcionalidades por separado. Entre ellas está la de actualización de debdelta con el método upgrade, donde se indica la acción de descargar desde el repositorio de los parches, las actualizaciones que estén disponibles para el sistema, según las aplicaciones que estén instaladas en el *host*, registradas en la caché de apt o los nuevos parches que existan en su repositorio especial.

La idea que se sigue es que cuando el usuario quiera instalar alguna aplicación el gestor integrado a debdelta, en este caso apt, le indique cuales son las dependencias que va a actualizar o en caso de hacer una actualización solamente indicar cuál es la aplicación a actualizar.

Por tal motivo debería existir la forma de que debdelta trabaje con una nueva funcionalidad que en vez de actualizar y descargar todos los parches disponibles, solamente lo haga con el que sea necesario, es decir con el que le indique apt y de esta forma agilizar el trabajo de debdelta, reduciendo su tiempo de descarga, solo a los parches de los paquetes que le sean especificados. Es por eso, que es necesaria la creación de un nuevo método upgrade para debdelta donde se indique solamente el o los paquetes a actualizar con parches.

2.4.1.3 Modificación de debdelta-upgrade.

La acción debdelta-upgrade, consiste en un método bastante extenso cuya funcionalidad básica es descargar todos los parches binarios de actualizaciones de los programas que estén instalados en el *host*. Cuenta también con otras acciones necesarias para una exitosa descarga y aplicación de deltas, entre ellas se pueden detectar, la capacidad para realizar las conversiones de los tamaños a mostrar de los archivos a descargar, verificación de permisos de directorios a descargar, chequeos de direcciones de repositorios, construcción y descarga de deltas, de acuerdo a su nomenclatura, entre otras funciones.

A continuación se muestra un diagrama con el flujo de procesos de debdellta-ugrade, donde se observa de forma básica la funcionalidad de debdelta para actualizar paquetes binarios a través de sus deltas, sin tener en cuenta las acciones adicionales que fueron enunciadas anteriormente.

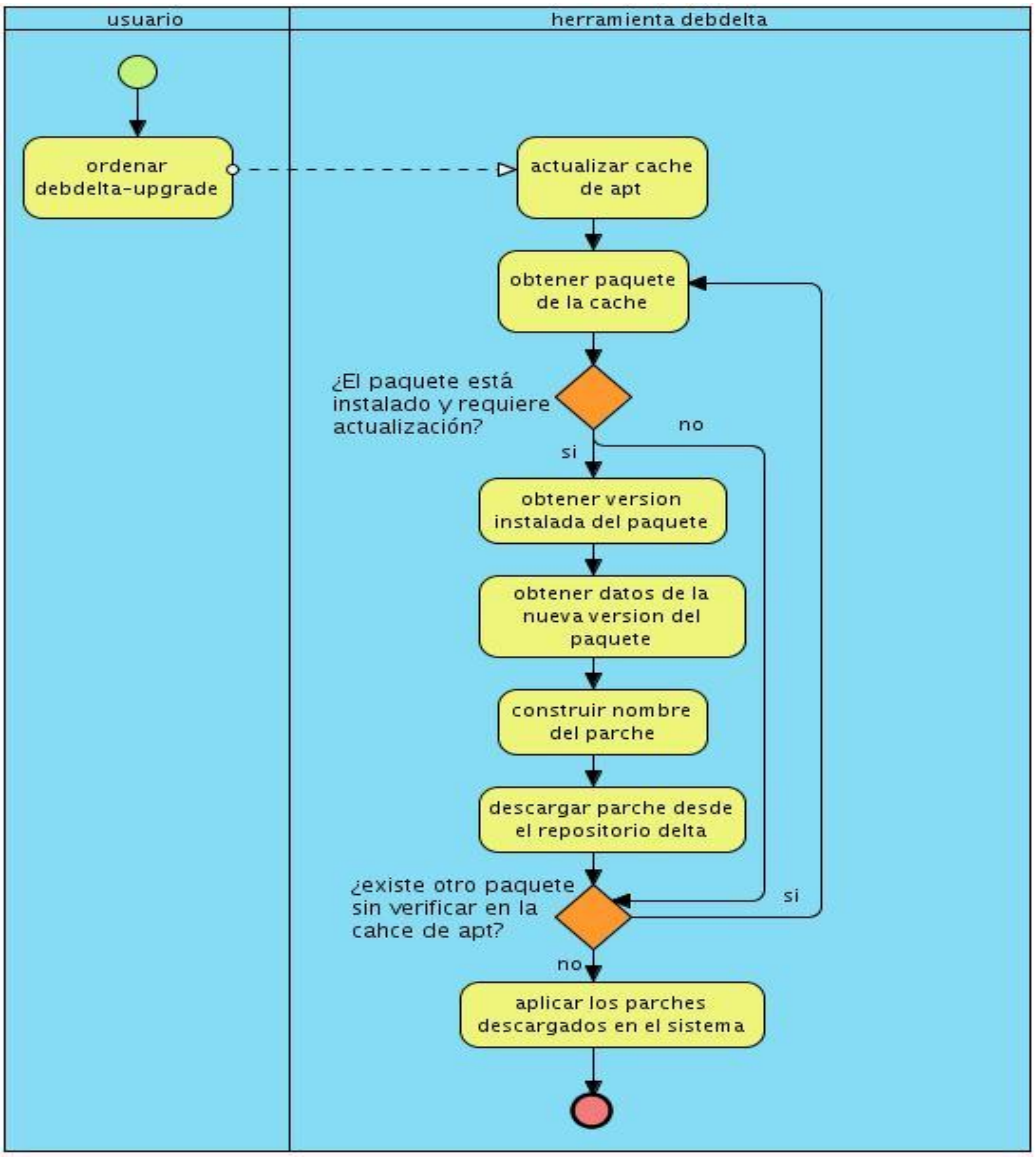


Diagrama 2.2: Flujo de procesos de debdelta-upgrade.

Con la muestra del diagrama 2.2 se puede observar el inconveniente que inicialmente debdellta-upgrade actualiza todos los paquetes en la cache, es en este sentido que se propone modificar el algoritmo, de tal forma que solamente descargue el o los paquetes que realmente el usuario necesita actualizar. Lo explicado se resuelve de la forma que se observa en el diagrama 2.3:

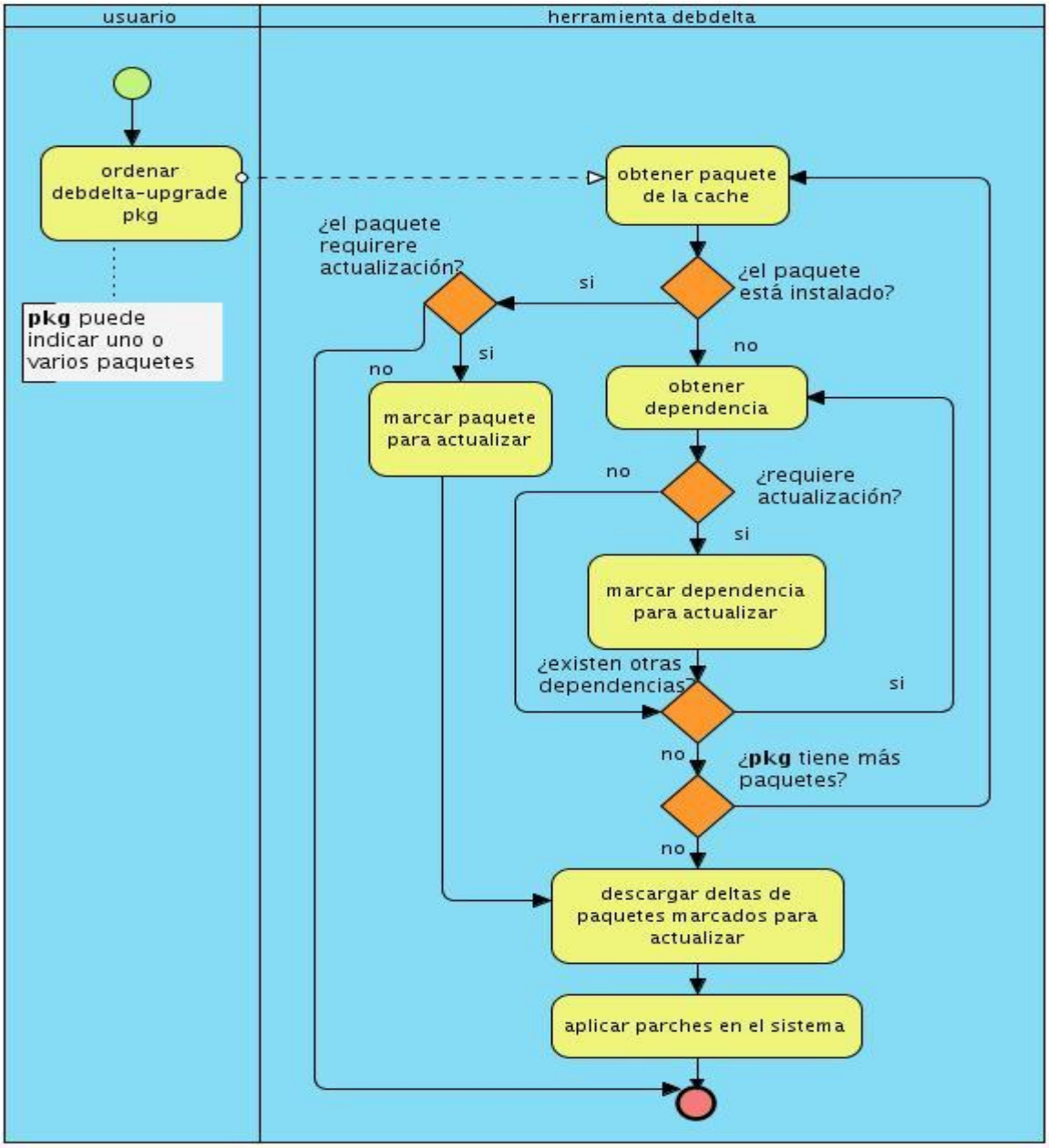


Diagrama 2.3: Flujo de procesos de debdelta-upgrade modificado.

Hay que tener en cuenta un aspecto importante para el entendimiento del diagrama 2.3: cuando pkg, indique varios paquetes, es necesario hacer el mismo algoritmo tantas iteraciones como número de paquetes existan. De esta forma cada paquete realizará todo los procesos descritos por la parte que ejecuta debdelta.

Analizando el diagrama que resuelve el inconveniente de realizar la aplicación de parches solo a los paquetes especificados, se llega a la creación de un nuevo debdelta-upgrade cuya diferencia con el original es el código en python que se indica en el Anexo 5.

2.4.1.4 Modificación de apt-get.

Una vez obtenida la funcionalidad de debdelta para actualizar los paquetes deseados, se procede a la inclusión de dicho proceso dentro del marco de apt. En aras de cumplir dicha transformación se estudiaron los ficheros del código fuente de apt, en donde el fichero apt-get.cc detalla todas las acciones de tal aplicación, entre ellas la de update, upgrade e install, en donde se incluyeron de diferentes formas, según las operaciones de cada método, la funcionalidad de debdelta (Ver Anexo 6).

Específicamente para insertar la funcionalidad de debdelta correctamente dentro de apt-get, se utilizó la librería stdlib.h de C ++ que incluye la propiedad **system**, permitiendo ejecutar en este tipo de código comandos del sistema.

La manera de incluir debdelta con system, se introdujo de tal forma que para actualizar algún paquete con la orden **apt-get install** dicho paquete a instalar, lo renueve en caso de que esté instalado o actualice sus dependencias, si requiere instalación. Por otra parte en el caso de **apt-get upgrade** se incluyó debdelta para que antes de instalar las nuevas actualizaciones del repositorio de la distribución, se priorice primero la posibilidad de descargarlas e instalarlas a través de deltas.

2.5 Conclusiones del capítulo.

Luego de haber realizado y analizado las propuestas para solucionar la mejora de la disminución de los tiempos en la descarga del repositorio de la distribución Nova, se concluyó la implementación de las mismas para pasar a un nuevo capítulo donde se comprobará su efectividad en el sistema y su acertada disminución en tiempos de descargas de paquetes.

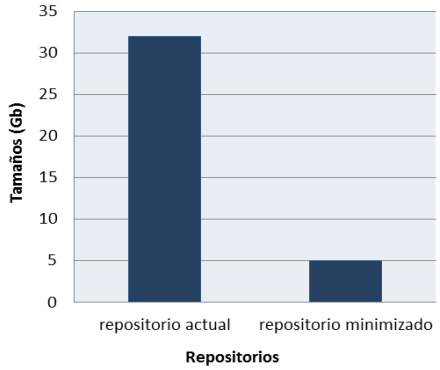
CAPÍTULO 3: “Validación de las propuestas”.

En el capítulo 2 se determinó de las propuestas estudiadas, la forma de implementarlas y de integrarlas al sistema Nova sobre la base de cumplir con el requisito fundamental de reducir el tamaño del repositorio de la distribución y con ello su tiempo de descarga. Es por eso que se hace necesario llevar a cabo un proceso de validación para así asegurar la calidad y funcionalidad de los procesos concebidos para satisfacer las necesidades descritas. Dichas comprobaciones se hacen con determinados casos de estudio en las mismas condiciones tecnológicas a una conexión de 100Mbps y están encaminadas a determinar la reducción de tiempo con los espacios reducidos en las muestras.

Para las validaciones se escogió una muestra aleatoria pequeña de paquetes, para demostrar que efectivamente se reduce el tamaño y con ella el tiempo de descarga. Se seleccionaron diez paquetes debido a que por el alcance de la tesis, se hace imposible en el período de tiempo estimado, se pueda realizar para el caso de las construcciones y actualizaciones de paquetes, compilaciones y creación de parches respectivamente para un repositorio de 38286 aplicaciones. Lo mismo no sucede para la comprobación de la efectividad del repositorio reducido, pues su caso de prueba es crear un único repositorio de paquetes ya existentes. Por dichas razones la selección cuantitativa, aunque sea mínima, dará a conocer un estimado del porcentaje en que se puede reducir el tamaño y el tiempo de descarga de un repositorio.

3.1 Validación de la optimización del repositorio reducido.

Una vez ejecutado el proceso detallado en la sección 2.1 para la creación del repositorio minimizado utilizando los paquetes instalados por defecto en las versiones de Nova, se procede a la comparación mostrada en la gráfica 3.1 de los tamaños entre el repositorio actual de la distribución y el reducido.



Gráfica 3.1 Tamaños de los repositorios original y minimizado.

La gráfica 3.1 demuestra que para un repositorio de 32 Gb si se seleccionan solamente los paquetes necesarios, se obtiene un repositorio con un volumen considerablemente menor que el original de sólo 5 Gb, lo que representa un 15,6 % del total, es decir existe un ahorro de espacio de aproximadamente 84,4 %.

A continuación se procede a la comparación de los tiempos de descarga de ambos repositorios. Para tal acción, se procede a descargar los repositorios con dos herramientas que permiten crear *mirrors*[21], ellos son debmirror y apt-mirror.

Herramienta para descarga.	Tiempo para repositorio actual.	Tiempo para repositorio reducido.	Diferencia de tiempo.	Por ciento de ahorro de tiempo
debmirror	90 min	11 min	79 min	87,70%
apt-mirror	87 min	10 min	77 min	88,50%

Tabla 3.1 Tiempos de descarga del repositorio minimizado.

Tras el análisis de la tabla 3.1 con las herramientas para las descargas, se concluye que la diferencia de tiempo tiene una media aproximada es de 78 min, lo que significa que el promedio de ahorro en tiempo de descarga de un repositorio reducido es de un 88 %.

3.2 Validación del proceso de compilación.

El proceso de compilación que se verifica, cumple con la condición que durante el trabajo de dpkg para construir paquetes, se utilice tanto para los de código fuente como para los archivos internos de los paquetes .deb, el algoritmo de compresión del formato tar.7z.

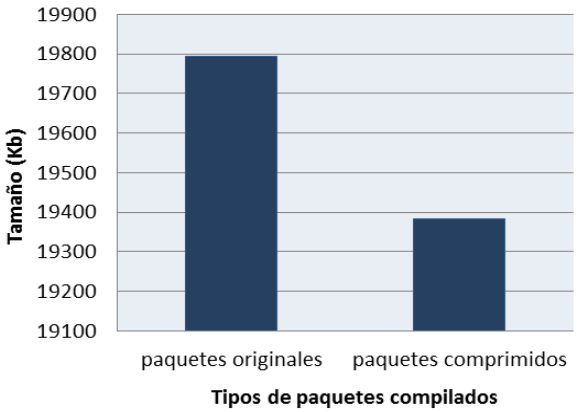
Para comprobar que disminuya la tardanza en la transferencia, se realiza la siguiente comparación de la tabla 3.2 con una muestra de un repositorio de 10 paquetes. Dicha verificación se hace con los tamaños y tiempos de descarga del repositorio creado con los paquetes compilados con métodos de tar.7z y el repositorio sin modificaciones.

21 Mirror: repositorio o espejo local.

Paquetes	Paquete compilado sin compresión tar.7z (Kb)	Paquete compilado con compresión tar.7z (Kb)
debian-goodies_0.51_all	372,4	356,9
adduser_3.113ubuntu2_all	2969,6	2764,8
debootstrap_1.0.37_all	719,3	672,8
base-files_5.0.0ubuntu20.10.04.3nova2~2011_i386	834,3	713,2
alien_8.79_all	800,6	739,3
d-shlibs_0.41ubuntu1_all	294,1	266,9
debianutils_4.0.2_i386	1005,3	1024
datefudge_1.14_i386	133	98,6
dbs_0.45_all	478,6	457,7
dctrl-tools.2.14_all	12185,6	12288

Tabla 3.2 Comparación de tamaños de paquetes compilados con tar.7z.

Tales resultados se ilustran en la gráfica 3.2:



Gráfica 3.2 Tamaños de repositorio original y repositorio comprimido internamente en tar.7z.

Dicho repositorio de muestra comprimido con el nuevo método de compresión, supone un ahorro del 3% del espacio en el servidor, pues representa el 97% del tamaño total. Si se sigue el mismo método de comprobación de tiempos de descargas con las mismas herramientas que en el caso anterior, se obtienen los siguientes resultados:

Herramienta para descarga.	Tiempo para repositorio de muestra.	Tiempo para repositorio comprimido.	Diferencia de tiempo.	Por ciento de ahorro
debmirror	1 s	0,9 s	0,1 s	10,00%
apt-mirror	1 s	0,7 s	0,3 s	30,00%

Tabla 3.3 Comparación de tiempos de descarga del repositorio comprimido con tar.7z.

Las soluciones obtenidas en la tabla 3.3 indican que con el método de compresión tar.7z, se reduce el tiempo con un promedio de ahorro del 20 %.

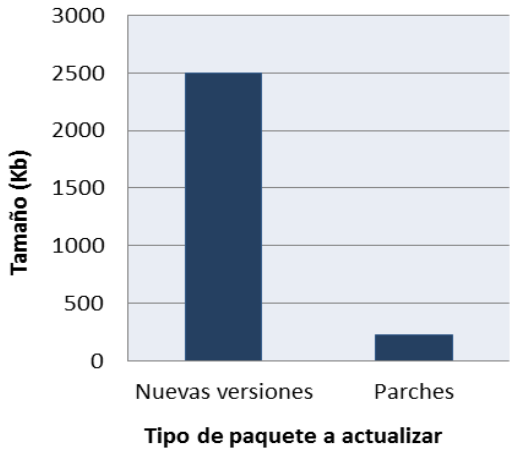
3.3 Validación de la integración de parches al sistema de Nova.

Una vez aplicadas las modificaciones a las herramientas debdellta y apt, se tiene la siguiente tabla 3.4 que expresa los tamaños de diferentes versiones de un paquete y su parche:

Paquete	Vieja versión (Kb)	Nueva versión (Kb)	Parche (Kb)
acl	49,9	41,7	20
adduser	116,4	129,5	69,9
alien	82,9	82,7	8,3
dbconfig-common	462,9	465	13
dblatex	1536	1536	10,2
dbs	24	24,8	6,6
debianutils	47,8	61,2	49,5
debootstrap	34,6	34,7	5,8
defoma	98,4	89,1	15,4
automoc	29,6	30,5	24,7
openoffice	2,9	2,9	3,1

Tabla 3.4 Comparación entre los tamaños de parches y las nuevas versiones de paquetes.

Los resultados tras la ejecución de la herramienta de parches para cada par de paquetes se ilustran en la gráfica 3.3:



Gráfica 3.3 Tamaños de repositorio de actualizaciones y sus parches.

Contar con un repositorio de parches para las actualizaciones, es más efectivo que almacenar sus nuevas versiones originales, pues representa solamente una capacidad de un 9 % con respecto al original, siendo un ahorro del 91% del tamaño original. Este hecho también es ventajoso para agilizar la descarga a través de herramientas para descargar archivos por el protocolo ftp, ya que por las herramientas tradicionales no se acepta, por ser un repositorio sin índices. Dichas comparaciones se indican a continuación:

Herramienta para descarga.	Tiempo para repositorio de actualizaciones de muestra.	Tiempo para repositorio de parches.	Diferencia de tiempo.	Por ciento de ahorro
wget	0,3	0,03	0,27	90,00%
curl	0,26	0,02	0,24	92.30%

Tabla 3.5 Comparación entre los tiempos de descarga del repositorio de parches y el de muestra.

Igualmente tales cálculos indican la reducción de demoras en la descarga, ya que se evidencia un promedio de ahorro de descargar a través de parches de un 91,15%.

3.4 Validación de la disminución de tiempo de las tres propuestas en el sistema.

Una vez probado la efectividad de los métodos propuestos en la distribución Nova, se tienen las representaciones siguientes de los tiempos reducidos en cuanto al total de tiempo que anteriormente ocupaba, para los casos de prueba que se validaron:

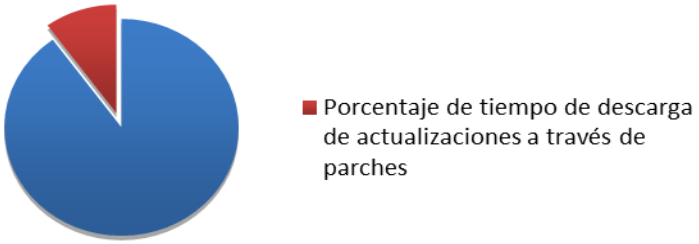
- Utilizando repositorio minimizado:

Gráfica 3.4 Por ciento de descarga para repositorio minimizado.

- Utilizando compresiones tar.7z en la compilación de paquetes:

Gráfica 3.5 Por ciento de descarga para repositorio comprimido en tar.7z.

- Utilizando parches para las descargas de actualizaciones:



Gráfica 3.6 Por ciento de descarga para repositorio de actualizaciones con parches.

De esta forma se visualizan los resultados por separados, pero si se integran todos, el resultado final sería el que se describe en las tablas 3.6 y 3.7, comenzando con el tiempo de descarga y tamaño del repositorio actual respectivamente:

Tiempos de descargas (min)	Propuesta aplicada	Resultado de tiempo de descarga (min)
78	Crear repositorio minimizado.	10
10	Compresión a paquetes en tar.7z.	8
8	Creación de parches para las actualizaciones.	1

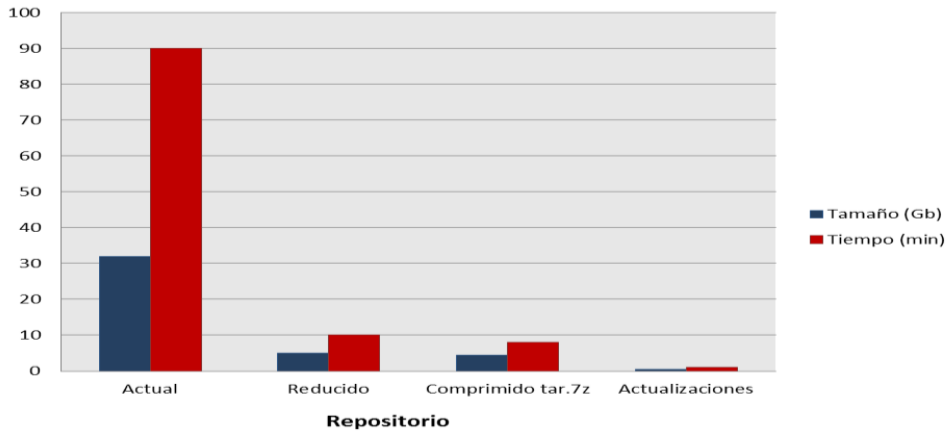
Tabla 3.6 Evolución del ahorro de tiempos aplicando propuestas en el repositorio.

De la misma forma se puede analizar los tamaños resultantes partiendo de 32Gb:

Tamaños de descargas(Gb)	Propuesta aplicada	Resultado de tamaño de descarga (Gb)
32	Crear repositorio minimizado.	5
4,7	Aplicar compresión a paquetes en tar.7z.	4,5
4,6	Aplicándole la creación de parches para la actualización.	0,41

Tabla 3.7 Evolución del ahorro de tamaños aplicando propuestas en el repositorio.

Tales resultados se detallan en la gráfica siguiente:



Gráfica 3.7 Tamaños y tiempos de evolución de reducción en volumen y tiempo de descarga del repositorio actual respectivamente.

A pesar de que son cálculos aproximados para un determinado caso de estudio se puede estimar, según los porcentajes obtenidos, que un repositorio actual de la distribución, se puede descargar en 8 min y sus actualizaciones en 1 min, lo que representa un 78 % del total de tiempo inicial de descarga y 12 % del tiempo para actualizaciones.

Estos indicadores demuestran que existe un ahorro de tiempo de 22 % para descargar el repositorio actual con un 88 % de tiempo para las actualizaciones.

3.5 Conclusiones del capítulo.

Debido a la complejidad de compilar paquetes y hacer un repositorio de parches para una muestra representativa para un repositorio de más de 38000 paquetes, se escoge una muestra más pequeña, obteniendo porcientajes que demuestran que efectivamente se reduce el tiempo de la descarga si se disminuye el tamaño de los paquetes, con la aplicación de cada una de las propuestas analizadas.

Este hecho da la idea de que si con los estudios estadísticos demostrados en el capítulo 1 el tamaño del repositorio disminuye un 55 % de su volumen total y un 43,9 % para sus actualizaciones, entonces aplicando el porcentaje de tiempo ahorrado, según las validaciones hechas, en dicha demora se estima una reducción del 78 % del tiempo total y un 12 % del tiempo para actualizaciones.

Tras la comprobación de la ejecución de las propuestas para casos particulares de estudio, se evidencian los resultados reales, que hacen que se pueda contar con una vía más eficiente para la descarga del repositorio de Nova en todo el país, ahorrando tanto en la descarga como en el almacenamiento del repositorio más del 50 % de los recursos tecnológicos que se utilizan hoy en día en este proceso.

CONCLUSIONES GENERALES

Al finalizar esta investigación, se tienen las siguientes conclusiones:

- Se realizó un análisis profundo de cada una de las propuestas de mejoras para disminuir los tiempos en la descarga del repositorio de Nova, apoyando dichas investigaciones en comparaciones a través de gráficos y tablas matemáticas para escoger las mejores y sostener las proposiciones hechas.
- Se implementaron prototipos para cada una de las funcionalidades que se relacionaban con las soluciones.
- Se aplicó cada solución investigada e implementada a la distribución GNU/Linux Nova, comprobando su buen funcionamiento en el sistema operativo.
- Se obtuvo una reducción del volumen del repositorio para un determinado caso de estudio a más del 80 % del tamaño total.
- Se cuenta actualmente con la integración de todos los resultados, llegando a obtener en la descarga del repositorio de la distribución, una reducción en su tiempo de transmisión de hasta un 78 % de lo que antes se descargaba con un 12 % para las actualizaciones para una determinada muestra, cumpliendo el objetivo general de disminuir en tiempo en la descarga del repositorio de la distribución.

Con la culminación exitosa de la investigación, se responde a la necesidad de que los usuarios que utilicen el sistema operativo Nova en sus versiones venideras cuenten con el aporte práctico y teórico de aplicar las soluciones que se proponen en aras de acotar los tiempos en las descargas de los repositorios de Nova. De tal forma el trabajo con la red será fructífero, pues disminuirá la espera de los usuarios y la congestión de la red mejorando así la adaptabilidad a las condiciones tecnológicas del país. De esta forma se favorecerá al crecimiento de usuarios que se beneficien del sistema operativo cubano, haciendo que la migración en el país continúe.

RECOMENDACIONES

Al finalizar la investigación se recomiendan un conjunto de mejoras:

- Integrar las soluciones en la próxima versión del sistema operativo Nova.
- Compilar el repositorio de la distribución con la modificación de dpkg, utilizando el futuro sistema de compilación para Nova.
- Crear un repositorio de parches para aplicar en la nueva versión de Nova.
- Demostrar con muestras representativas del 25% del repositorio actual, la reducción de tamaños y tiempos de descargas del mismo.
- Realizar estudio de las herramientas aceleradoras de descargas, en aras de incluirlas a las integraciones hechas en la investigación favoreciendo a que la transferencia sea más rápida.
- Determinar en próximos estudios, qué herramienta es la más favorable para las descarga de repositorios locales.
- Adaptar a los gestores de paquetes para que puedan interactuar con paquetes binarios comprimidos internamente con tar.7z.
- Incluir las nuevas funcionalidades de debdelta, apt y dpkg en sus respectivos manuales (*man*).
- Para la creación de un repositorio reducido, añadir además de los paquetes básicos, una selección de los más usados, seleccionándolos a través de un sistema integrado al repositorio para determinar cuáles son los que más descarga la comunidad de software libre.
- Elaborar encuestas a la comunidad de software libre, que indiquen qué paquetes adicionales incluir en el repositorio reducido, además de los que vienen por defecto.

BIBLIOGRAFÍA REFERENCIADA

[1] PIERRA FUENTES, Allan. *Nova Distribución cubana de GNU/Linux: soberanía tecnológica, seguridad, criollo.* Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2011. p. 9, 18, 19, 29.

[2] UBUNTU. *Instalar software.* [en línea]. Consultado el: 12 diciembre 2011. Disponible en la Web: <http://doc.ubuntu-es.org/Instalar_software>.

[3] AOKI, Osamu y ECHARRI, Walter. Debian Reference (version 1) - Fundamentos de Debian. [en línea]. Consultado el: 12 de Diciembre 2011. Disponible en la Web: <http://www.debian.org/doc/manuals/debian-reference/ch-system.es.html#s-deb-format>.

[4] ELLIS, Stuart y FRIELDS, Paul. Fedora Core 4: Administrando Software con yum. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://docs.fedoraproject.org/es-ES/Fedora_Core/4/pdf/Software_Management_Guide/Fedora_Core-4-Software_Management_Guide-es-ES.pdf>. p 4.

[5] DEBIAN Proyect. deb man (5). Manual Linux. 27 de febrero de 2009.

[6] ALBO, Castro Mónica. *Sistema para automatizar la generación de paquetes binarios de la distribución Nova*. [en línea]. Universidad de las Ciencias Informáticas. Ciudad de la Habana Junio 2008. p 7-9.

[7] ISOTTON, Aaron. Debian Repository HOWTO. [en línea]. Consultado el 5 abril 2012. Disponible en la Web: <http://www.isotton.com/software/debian/docs/repository-howto/repository-howto.html>.

[8] OBREGON, Alejandra. Repositories - Community Ubuntu Documentation. Repositories. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <https://help.ubuntu.com/community/Repositories>.

[9] GÓMEZ, Savino Guillermo. Manual para la gestión de software: Guía definitiva para la gestión. Edición. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://docs.fedoraproject.org/es-ES/Fedora_15/0.1/html/Software_Management_Guide/ch02s02.html>.

[10] UBUNTU. Componentes de los repositorios. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://doc.ubuntu-es.org/Componentes_de_los_repositorios>.

[11] PARCHE. En: Diccionario de informática e Internet de Microsoft. Sánchez González Carmelo, ed. Mc Graw Hill. Ediciones profesionales de Microsoft. 2000. ISBN 48-481-2893-1. p 434.

[12] SUEL, Torsten y MEMON, Nasir. *Algorithms for Delta Compression and Remote File Synchronization.* CIS Department Polytechnic University Brooklyn. p. 1-2.

[13] NOVA. Nova » [tdnova'12] Sesión Introductoria. [en línea]. Consultado el 3 abril 2012. Disponible en la Web: <http://comunidades.uci.cu/blogs/nova/?p=698>.

[14] UBUNTU. 7-Zip. [en línea]. Consultado el 21 marzo 2012. Disponible en la Web: <http://www.guia-ubuntu.org/index.php?title=7-zip>.

[15] UBUNTU. P7zip [en línea]. Consultado el 26 marzo 2012. Disponible en la Web: <http://www.guia-ubuntu.org/index.php?title=P7zip>.

[16] PAVLOV, Igor. 7-Zip. [en línea]. Consultado el 9 abril 2012a. Disponible en la Web: <http://www.7-zip.com.mx/>.

[17] ECURED. Lz77. [en línea]. Consultado el 9 abril 2012c. Disponible en la Web: <http://www.ecured.cu/index.php/Lz77>.

[18] SUEL, Torsten y MEMON, Nasir. *Algorithms for Delta Compression and Remote File Synchronization.* CIS Department Polytechnic University Brooklyn. p. 4-5.

[19] PERCIVAL, Colin. *Naive Differences of Executable Code.* Computing Lab, Oxford University. p. 2-3.

[20] MENNUCCI, Andrea. The debdelta suite. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://debdelta.debian.net/html/index.html>.

[21] MENNUCCI, Andrea y PIERREYVES, Jegou. *Debdelta.* Página Manual Linux. Agosto 2009.

[22] GUNTHORPE, Jason y BURROWS, Daniel. Manual apt.conf. Página Manual Linux. Enero 2009.

BIBLIOGRAFÍA CONSULTADA

- AGARWAL Ramesh y AMALAPURAPU Suchitra. *An Approximation to the Greedy Algorithm for Differential Compression of Very Large Files.* Department of Electrical Engineering and Computer Sciences. 2000.
- ALBO CASTRO Mónica. *Sistema para automatizar la generación de paquetes binarios de la distribución Nova*. [en línea]. Universidad de las Ciencias Informáticas. Ciudad de la Habana Junio 2008. p. 7-9.
- AOKI Osamu y ECHARRI Walter. Debian Reference (version 1) - Fundamentos de Debian. [en línea]. Consultado el: 12 de Diciembre 2011]. Disponible en la Web: <http://www.debian.org/doc/manuals/debian-reference/ch-system.es.html#s-deb-format>.
- COMPRESIÓN DE DATOS. En: Diccionario de informática e Internet de Microsoft. Sánchez González Carmelo, ed. Mc Graw Hill. Ediciones profesionales de Microsoft. 2000. ISBN 48-481-2893-1. p 138.
- COMPRESION.ES. Compresión de Datos - Compresión Compresores de archivos, ficheros y carpetas. [en línea]. Consultado el 15 marzo 2012a. Disponible en la Web: <http://www.compresion.es/compresion-de-datos/>.
- DEBIAN Proyect. deb man (5). Manual Linux. 27 de febrero de 2009.
- ECURED MD5. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://www.ecured.cu/index.php/MD5>.
- ECURED. Compresión de datos. [en línea]. Consultado el 15 marzo 2012b. Disponible en la Web: <http://www.ecured.cu/index.php/Compresi%C3%B3n_de_datos>.
- ECURED. Lz77. [en línea]. Consultado el 9 abril 2012c. Disponible en la Web: <http://www.ecured.cu/index.php/Lz77>.
- ECURED. SHA [en línea]. Consultado el 15 marzo 2012. Disponible en la Web: <http://www.ecured.cu/index.php/SHA>.
- ELLIS Stuart y FRIELDS Paul. Fedora Core 4: Administrando Software con yum. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://docs.fedoraproject.org/es-ES/Fedora_Core/4/pdf/Software_Management_Guide/Fedora_Core-4-Software_Management_Guide-es-ES.pdf>. p. 4.

- FEDORA. presto. [en línea]. Consultado el 31 Enero 2012. Disponible en la Web: <http://fedoraproject.org/wiki/Features/Presto>.
- GAILLY Jean-loup y ADLER Mark. The gzip home page. [en línea]. Consultado el 15 marzo 2012a. Disponible en la Web: <http://www.gzip.org/>.
- GNU. Tar - GNU Project - Free Software Foundation (FSF). [en línea]. Consultado el 15 marzo 2012a. Disponible en la Web: <http://www.gnu.org/software/tar/>.
- GÓMEZ SAVINO Guillermo. Manual para la gestión de software: Guía definitiva para la gestión. Edición. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://docs.fedoraproject.org/es-ES/Fedora_15/0.1/html/Software_Management_Guide/ch02s02.html>.
- GRIMANY López Elena. Conexiones en las empresas cubanas. Marzo 2012.
- GUNTHORPE Jason y BURROWS Daniel. Manual apt.conf. Página Manual Linux. enero 2009.
- GZIP. Ubuntu Manpage: gzip, gunzip, zcat. Página Manual Linux.
- HUNT James J y VO KIEM Phong. Delta Algorithms: An Empirical Analysis. University of Karls-ruhe, Germany. p 1-6.
- ISOTTON Aaron. Debian Repository HOWTO. [en línea]. Consultado el 5 abril 2012. Disponible en la Web: <http://www.isotton.com/software/debian/docs/repository-howto/repository-howto.html>.
- JEAN-LOUP Gailly y MARK Alder. ZLIB. [en línea]. Consultado el 21 Enero 2012b. Disponible en la Web: <zlib.org>.
- MACDONALD Josh. dcms-dev: Xdelta 2.0beta1. Xdelta. [en línea]. Consultado el 9 marzo 2012ª. Disponible en la Web: <http://www.eros-os.org/~majordomo/dcms-dev/0212.html>.
- MACDONALD Josh. xdelta. xdelta. [en línea]. Consultado el 21 Enero 2012. Disponible en la Web: <http://xdelta.org/>.
- MACDONALD Joshua P. *File System Support for Delta Compression*. University of California at Berkeley.2006
- MARZO Lázaro José Luis. *Control de tráfico en redes de altas prestaciones: ATM e Internet de nueva generación*. Departamento de Electrónica, Informática y Automática. Universidad de Giro-na. España. p 1-24.
- MENNUCCI Andrea y PIERREYVES Jegou. *Debdelta.* Página Manual Linux. Agosto 2009.

- MENNUCCI Andrea. The debdelta suite. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://debdelta.debian.net/html/index.html>.
- MILLER Webb y MYERS Eugene. *A File Comparison Program*. Department of Computer Science, Tucson. 1985.
- NOVA » [tdnova'12] Sesión Introductoria. [en línea]. Consultado el 3 abril 2012. Disponible en la Web: <http://comunidades.uci.cu/blogs/nova/?p=698>.
- OBREGON Alejandra. Repositories - Community Ubuntu Documentation. Repositories. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <https://help.ubuntu.com/community/Repositories>.
- PARCHE. En: Diccionario de informática e Internet de Microsoft. Sánchez González Carmelo, ed. Mc Graw Hill. Ediciones profesionales de Microsoft. 2000. ISBN 48-481-2893-1. p 434.
- PAVLOV. Igor. 7-Zip. [en línea]. Consultado el 9 abril 2012a. Disponible en la Web: <http://www.7-zip.com.mx/>.
- PERCIVAL Colin. *Naive Differences of Executable Code.* Computing Lab, Oxford University. p. 2-3.
- PIERRA FUENTES Allan. *Nova Distribución cubana de GNU/Linux: soberanía tecnológica, seguridad, criollo.* Universidad de las Ciencias Informáticas. Ciudad de la Habana. 2011. p 9, 18, 19, 29.
- RAMÓN Rodríguez Miguel Angel. Conectividad en Joven Clubs. Marzo 2012.
- RODRÍGUEZ Ávila Abel. *Iniciación a la Red Internet: Concepto, Funcionamiento, Servicios y Aplicaciones de Internet.* Ideaspropias Editorial. Vigo, 2007. ISBN: 978-84-9839-139-8. p. 1-7.
- SEWARD Julian. bzip2. Página manual Linux.
- SUEL Torsten y MEMON Nasir. *Algorithms for Delta Compression and Remote File Synchronization.* CIS Department Polytechnic University Brooklyn.
- UBUNTU. 7-Zip. [en línea]. Consultado el 21 marzo 2012. Disponible en la Web: <http://www.guia-ubuntu.org/index.php?title=7-zip>.
- UBUNTU. Componentes de los repositorios. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://doc.ubuntu-es.org/Componentes_de_los_repositorios>.
- UBUNTU. *Instalar software.* [en línea]. Consultado el: 12 diciembre 2011. Disponible en la Web: <http://doc.ubuntu-es.org/Instalar_software>

- UBUNTU. P7zip [en línea]. Consultado el 26 marzo 2012. Disponible en la Web: <http://www.guia-ubuntu.org/index.php?title=P7zip>.
- UBUNTU. Rar - Guía Ubuntu. [en línea]. Consultado el 26 marzo 2012b. Disponible en la Web: <http://www.guia-ubuntu.org/index.php?title=RAR>.
- VESGA Ferreira Juan Carlos. *Modelo empirico para predicción de hroughput en redes lan sobre power line communications.* junio 2009. ISBN: 2027-2646. p. 38-39.
- WERNER Koch. The GNU Privacy. [en línea]. Consultado el 9 marzo 2012. Disponible en la Web: <http://www.gnupg.org/>
- ZIP. zip(1): package/compress files. Página Manual Linux.

ANEXOS

Anexo 1. **Entrevista realizada en instituciones.**

Preguntas:

1. ¿En la institución siguen los planes de migración de la informatización cubana?
2. ¿Usan actualmente el Sistema Operativo Nova?
3. ¿Cuáles son los principales problemas que tienen con el Sistema Operativo Nova?
4. ¿Cómo acceden a los repositorios de la distribución cubana?
5. ¿Cuál es la conectividad que actualmente se tiene en el centro?
6. ¿Qué equipos de conexión tienen?
7. ¿Tiempo aproximado en la descarga del repositorio entero?

Anexo 2. **Script para unir el listado de paquetes de distintas distribuciones de Nova.**

1. Para obtener de cada imagen el listado de paquetes:

```
mount –t iso9660 –o loop nova-ligth-2010-r2-i386.iso /mnt
cd mnt
find pool/ -type d | cut –d/ -f4 > ligero.txt
```

2. Ejecutar script para unir listas de cada imagen de línea de desarrollo de Nova.

```
#!/usr/bin/python
l=[]
f= open("ligero.txt", "r")
f1= open("escritorio.txt", "r")
f2= open("servidores.txt", "r")

while True:
    leer=f.readline()
    if not leer:
        break
    l.append(leer)

while True:
    leer=f1.readline()
    if not leer:
        break
    l.append(leer)

while True:
    leer=f2.readline()
    if not leer:
        break
    l.append(leer)
c=0

for i in l:
    if l.count(i) !=1:
        c=l.count(i)-1
        while (c!=0):
            l.remove(i)
            c=c-1

l.sort()
```

Anexo 3. **Código bash para crear repositorio minimizado.**

```
#!/bin/bash

DIST="2011"
REPO="/mnt/repo/new-extracted-repo/"
GNUPG_HOME=/root/.gnupg
SOURCE="/mnt/repo/nova2011/build/www/nova/"
ARCH=i386

set -x

insert_into_repo()
{
    COMPONENT=$( apt-cache showsrc $1 | grep Directory | cut -d/ -f2 | head -1 )
    FOLDER=$( apt-cache showsrc $1 | grep Directory | cut -d' ' -f2 | head -1 )/
    if [ -z "$(find $REPO | grep $FOLDER )" ]
    then
        reprepro -b $REPO --gnupghome $GNUPG_HOME -A $ARCH -C $COMPONENT includedeb $DIST $(find $SOURCE$FOLDER | grep .deb$ | grep -v udeb | grep -E "($ARCH|all)")
    fi
}

for i in $(cat $1)
do
    insert_into_repo $i
done
```

Anexo 4. **Código más relevante del parche realizado para dpkg para que utilice el método de compresión lzma.**

```
--- dpkg-1.16.1.2ubuntu7.orig/scripts/Dpkg/Compression.pm	2011-10-14 06:26:48.000000000 -0400
+++ dpkg-1.16.1.2ubuntu7/scripts/Dpkg/Compression.pm	2012-05-17 21:51:02.000000000 -0400
@@ -50,6 +50,12 @@ interact with the set of supported compr
 =cut

 my $COMP = {
+    "lzma" => {
+        "file_ext" => "lzma",
+        "comp_prog" => [ 'xz', '--format=lzma' ],
+        "decomp_prog" => [ 'unxz', '--format=lzma' ],
+        "default_level" => 6,
+    },
     "gzip" => {
         "file_ext" => "gz",
         "comp_prog" => [ "gzip", "--no-name", "--rsyncable" ],
@@ -62,12 +68,6 @@ my $COMP = {
         "decomp_prog" => [ "bunzip2" ],
         "default_level" => 9,
     },
-    "lzma" => {
-        "file_ext" => "lzma",
-        "comp_prog" => [ 'xz', '--format=lzma' ],
-        "decomp_prog" => [ 'unxz', '--format=lzma' ],
-        "default_level" => 6,
-    },
     "xz" => {
         "file_ext" => "xz",
         "comp_prog" => [ "xz" ],
@@ -76,7 +76,7 @@ my $COMP = {
     },
 };

-our $default_compression = "gzip";
+our $default_compression = "lzma";
 our $default_compression_level = undef;
```

```
diff -Nurp dpkg-1.16.1.2ubuntu7.orig/debian/source/options dpkg-1.16.1.2ubuntu7/debian/source/options
--- dpkg-1.16.1.2ubuntu7.orig/debian/source/options	2010-05-20 09:20:05.000000000 -0400
+++ dpkg-1.16.1.2ubuntu7/debian/source/options	2012-05-17 21:50:39.000000000 -0400
@@ -1,2 +1,2 @@
 # Use bzip2 compression by default, we save 2.5Mb
-compression = "bzip2"
+compression = "lzma"
diff -Nurp dpkg-1.16.1.2ubuntu7.orig/dpkg-deb/build.c dpkg-1.16.1.2ubuntu7/dpkg-deb/build.c
--- dpkg-1.16.1.2ubuntu7.orig/dpkg-deb/build.c	2011-10-14 06:26:48.000000000 -0400
+++ dpkg-1.16.1.2ubuntu7/dpkg-deb/build.c	2012-05-17 21:50:52.000000000 -0400
@@ -475,11 +475,11 @@ do_build(const char *const *argv)
     /* And run gzip to compress our control archive. */
     c2 = subproc_fork();
     if (!c2) {
-	compress_filter(&compressor_gzip, p1[0], gzfd, 9, _("control member"));
+	compress_filter(&compressor_lzma, p1[0], gzfd, 9, _("control member"));
 	exit(0);
     }
     close(p1[0]);
-    subproc_wait_check(c2, "gzip -9c", 0);
+    subproc_wait_check(c2, "lzma -9c", 0);
     subproc_wait_check(c1, "tar -cf", 0);

     if (lseek(gzfd, 0, SEEK_SET))
diff -Nurp dpkg-1.16.1.2ubuntu7.orig/dpkg-deb/dpkg-deb.h dpkg-1.16.1.2ubuntu7/dpkg-deb/dpkg-deb.h
--- dpkg-1.16.1.2ubuntu7.orig/dpkg-deb/dpkg-deb.h	2011-10-14 06:26:48.000000000 -0400
+++ dpkg-1.16.1.2ubuntu7/dpkg-deb/dpkg-deb.h	2012-05-17 21:50:52.000000000 -0400
@@ -58,7 +58,7 @@ extern int compress_level;
 #define OLDOLDDEBDIR	".DEBIAN"

 #define DEBMAGIC	"debian-binary"
-#define ADMINMEMBER	"control.tar.gz"
+#define ADMINMEMBER	"control.tar.lzma"
 #define DATAMEMBER	"data.tar"
```

Anexo 5. **Código más relevante del parche realizado a debdelta para que descargue solamente el o los paquetes especificados.**

```
+	cache=apt.Cache()
+	try:
+
+		pkg_upg=[]
+
+		if(len(argv)==1):
+			pkg = cache[argv[0]] # buscar el paquete en la cache
+
+			# si el paquete no esta instalado se obtienen las dependencias para actualizarlas con deltas
+			# y luego instalar el paquete
+
+			if ( pkg.is_installed != True): # si el paquete no esta instalado
+
+				print "the package " + pkg.name +" is not installed"
+				list = pkg.candidate.dependencies # obtener dependencias
+
+
+				for i in list: # para cada dependencia obtener datos
+					for j in i.or_dependencies:
+						nom_dep = j.name
+						#print nom_dep
+						pkg_dep = cache[j.name]
+						#print pkg_dep
+
+						if(pkg_dep.is_installed): # si la dependencia esta instalada, ver si se puede actualizar
+							print pkg_dep.name + ' is installed'
+
+							if(pkg_dep.is_upgradable):
+								print pkg_dep.name + ' is upgradable'
+								pkg_dep.mark_upgrade
+								pkg_upg.append(pkg_dep) # guardar en variable que actualizara con deltas
+
+						elif(pkg_dep.is_installed != true): # si la dependencia no esta instalada
+							pkg_dep.mark_install		# marcar para instalar
```

```
                if(pkg_dep.is_installed):
                    print pkg_dep.name + ' is installed'

                    if(pkg_dep.is_upgradable):
                        print pkg_dep.name + ' is upgradable'
                        pkg_dep.mark_upgrade
                        pkg_upg.append(pkg_dep)

                    elif(pkg_dep.is_installed != true):
                        pkg_dep.mark_install
                pkg.mark_install
                print pkg.name + ' has been installed'

            if ( pkg.is_installed):
                print pkg.name + ' is intalled'

                if(pkg.is_upgradable):
                    print pkg.name + ' has been upgrade'
                    pkg.mark_upgrade
                    pkg_upg.append(pkg)
                else:
                    print "the package " + pkg.name + " is not posible update with deltas"

            pkg.mark_install
            print pkg.name + ' has been installed'

#si el paquete ya esta instalado, verificar solamente que se pueda actualizar
        if ( pkg.is_installed):
            print pkg.name + ' is intalled'

            if(pkg.is_upgradable):
                print pkg.name + ' has been upgrade'
                pkg.mark_upgrade
                pkg_upg.append(pkg)
            else:
                print "the package " + pkg.name + " is not posible update with deltas"

    # si hay mas de 2 paquetes a actualizar hacer el mismo proceso con cada uno
    if(len(argv)!=1):

        # recorrer cada paquete
        for l in argv:
            pkg = cache[l]
            #print pkg.name+'...'

            if ( pkg.is_installed != True): #si no esta instalado ver las dependencias
                print "the package " + pkg.name +" is not installed" #si se pueden actualizar para luego instalar el paquete
                list = pkg.candidate.dependencies

                for i in list:
                    for j in i.or_dependencies:

                        nom_dep = j.name
                        #print nom_dep
                        pkg_dep = cache[j.name]
                        #print pkg_dep
```

Anexo 6. **Código más relevante del parche realizado a apt-get para que se integre en la actualización e instalación de paquetes con debdelta.**

```
diff -Nurp apt-0.8.16~exp12ubuntu10_orig/cmdline/apt-get.cc apt-0.8.16~exp12ubuntu10/cmdline/apt-get.cc
--- apt-0.8.16~exp12ubuntu10_orig/cmdline/apt-get.cc	2012-04-19 04:56:33.000000000 -0400
+++ apt-0.8.16~exp12ubuntu10/cmdline/apt-get.cc	2012-05-19 05:57:03.000000000 -0400
@@ -68,6 +68,8 @@
 #include <regex.h>
 #include <sys/wait.h>
 #include <sstream>
+#include <string.h>
+#include <string>
 
 #include <apti18n.h>
-	/*}}}*/
@@ -391,8 +393,21 @@ void ShowNew(ostream &out,CacheFile &Cac
 	    VersionsList += string(Cache[I].CandVersion) + "\n";
 	 }
       }
-
       ShowList(out,_("The following NEW packages will be installed:"),List,VersionsList);
+
+      char *i;
+      char *k = strdup(List.c_str());
+      string ListDeltas;
+      string e=" ";
+
+      for ( i = strtok(k," "); i != NULL; i = strtok(NULL, " ") )
+      {
+	 ListDeltas += i + e;
+	 execlp("debdelta-upgrade", i, 0);
+      }
+
+      ShowList(out,_("The following packages will be upgraded by deltas:"),List,VersionsList);
+
    }
    // ShowDowngraded - Show downgraded packages	/*{{{*/
@@ -802,6 +833,9 @@ struct TryToInstall {
 	 Fix->Clear(Pkg);
 	 Fix->Protect(Pkg);
       }
+      /*char *a = Pkg;
+      execlp("debdelta-upgarde",a,0 );
+      */
       Cache->GetDepCache()->MarkInstall(Pkg,false);
 
       if (State.Install() == false) {
@@ -1601,6 +1635,7 @@ pkgSrcRecords::Parser *FindSrc(const cha
 /* */
 bool DoUpdate(CommandLine &CmdL)
 {
+   execlp("debdelta-upgrade", " ",0);
    if (CmdL.FileSize() != 1)
       return _error->Error(_("The update command takes no arguments"));
 
@@ -1800,6 +1835,7 @@ bool DoAutomaticRemove(CacheFile &Cache)
    packages */
 bool DoUpgrade(CommandLine &CmdL)
 {
+   execlp("debdelta-upgrade", " ",0);
    CacheFile Cache;
    if (Cache.OpenForInstall() == false || Cache.CheckDeps() == false)
       return false;
@@ -1974,6 +2010,19 @@ bool DoInstall(CommandLine &CmdL)
       }
```

```
@@ -1974,6 +2010,19 @@ bool DoInstall(CommandLine &CmdL)
      }

      ShowList(c1out,_("The following extra packages will be installed:"),List,VersionsList);
+     char *i;
+     char *k = strdup(List.c_str());
+     string ListDeltas;
+     string e = " ";
+
+     for ( i = strtok(k," "); i != NULL; i = strtok(NULL, " ") )
+     {
+         ListDeltas += i + e;
+         execlp("debdelta-upgrade", i, 0);
+     }
+     ShowList(c1out,_("The following packages will be upgraded by deltas:"),List,VersionsList);
+
+
   }
```