ISSN 2076-8184. Інформаційні технології і засоби навчання. 2010. №5 (19).
Режим доступу до журналу: http://www.ime.edu-ua.net/em.html

УДК 37:004

Каплун Олексій Олександрович, провідний інженер відділу електронних інформаційних ресурсів і мережних технологій Інституту інформаційних технологій і засобів навчання НАПН України, м. Київ

Підхід до модернізації серверного забезпечення та мережевої топології інформаційної системи

"Якщо ви знайшли помилку, краще виправити її самостійно, ніж просити кого-небудь зробити це. Якщо ви роздумуєте про підвищення продуктивності або про додавання нових можливостей, простіше зробити це самостійно. Якщо виявляється дірка в системі безпеки, «залатайте» її самостійно і не чекайте, поки цим займеться постачальник ОС."

Скотт Максвелл

Анотація

Модернізація мереж, оновлення програмного та апаратного забезпечення інформаційних систем у галузі освіти є актуальною проблемою в сучасних умовах швидкого розвитку інформаційних технологій. У статті подано аналіз серверного забезпечення та мережевої топології Інтернет-центру Інституту інформаційних технологій і засобів навчання НАПН України, запропоновано шляхи модернізації їх. Матеріали статті відображають результати роботи з модернізації мережі, проведені з метою підвищення ефективності й надійності мережі, скорочення часу на очікування результатів і даних обробки в інформаційних системах Інституту під час роботи в мережі. У статті наведено схеми мережевої топології до модернізації та після її проведення.

Ключові слова: серверна операційна система, Ubuntu, драйвер Linux, ядро Linux, мережева топологія.

Вступ. Постановка проблеми. Головною проблемою модернізації серверного забезпечення є зворотна сумісність функціоналу. Тобто нове обладнання та програмне забезпечення повинно виконувати всі функції старого забезпечення, причому набагато

якісніше, а також бути сумісним з тим обладнанням, яке не змінюється в процесі модернізації.

Метою статті є розгляд на практичному завданні підходу та методики модернізації серверного забезпечення, у якому зберігається частина обладнання, яке не потребує модернізації та успішно налаштовується функціонування цього обладнання під новим програмним та апаратним забезпеченням.

Поточна структура мережі. Аналіз топології та маршрутизації

Основною функцією мережі відділу інформаційних ресурсів і мережних технологій Інституту інформаційних технологій і засобів навчання НАПН України є забезпечення безперебійного доступу до мережі Інтернет співробітників відділу та забезпечення цілодобової роботи Інтернет-ресурсів відділу. Розглянемо схематичне відображення топології мережі до модернізації (рис. 1).

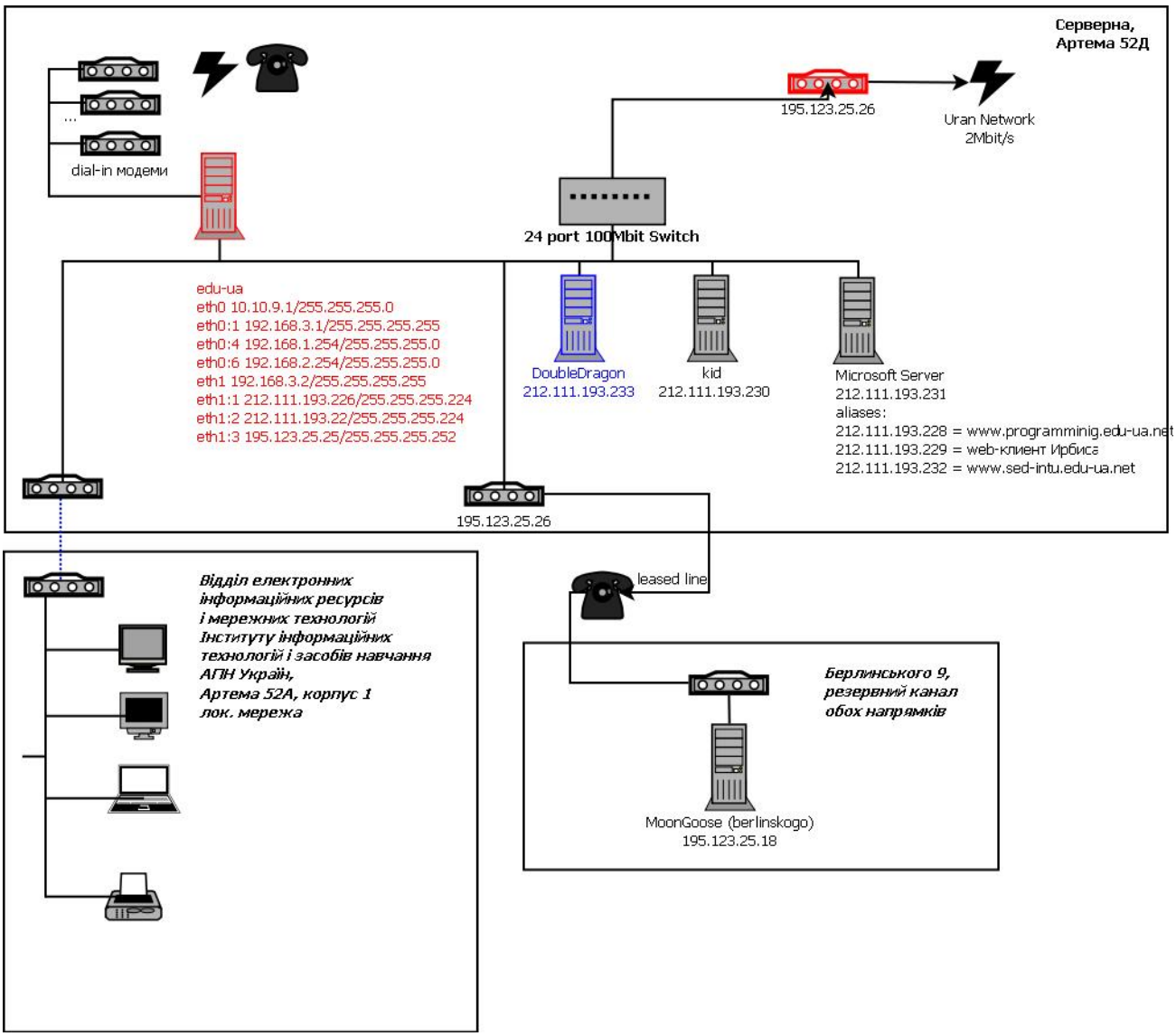

Рис. 1. Мережа відділу до початку модернізацій

На схематичному зображенні подано три основних сегмента мережі — це основний серверний сегмент, мережа відділу та серверний сегмент, який розташовано в окремому приміщенні і який є резервним каналом на випадок падіння основного каналу.

Головним серверним сегментом є чотири сервери та модеми-роутери, які фізично пов’язані між собою через один 100 Мбіт хаб. Головний канал, який забезпечує доступ усього сегмента до глобальної мережі Інтернет, організовано за допомогою модему (на рис. 1 позначено червоним кольором), підключеного через виділену лінію до Інтернет-провайдера УРАН. Максимальна пропускна спроможність каналу на даний час складає 2 Мбіт/с.

До головного серверного сегменту належать чотири сервери, це :

- **«Microsoft server»**, який працює під управлінням ОС Microsoft Windows Server та забезпечує роботу веб-ресурсів Інституту, забезпечує роботу пошти домену edu-ua.net;
- **«Kid»** – сервер, який працює під управлінням ОС Linux, забезпечує роботу електронної пошти, працює як веб-сервер, dns-сервер та проксі;
- «**EDU-UA**» – сервер, який працює під управлінням ОС Linux, працює центральним маршрутизатором мережі, проксі-сервером та сервером, який забезпечує «dial-up» доступ до мережі Інтернет через модемний пул.

Розглянувши топологію мережі, можна побачити, що одне фізичне середовище використовується одразу кількома логічними сегментами, центральним маршрутизатором яких є сервер EDU-UA, та центральним фізичним реплікатором — 100Мбіт хаб. Така організація не є оптимальною і призводить до частої появи колізій, що, у свою чергу, позначається на загальній швидкості та надійності мережі.

Проаналізувавши фізичну і логічну організацію і структуру мережі, можна зробити висновки, що головні елементи, від яких залежить швидкість та надійність мережі — це центральний хаб та сервер — маршрутизатор EDU-UA. Отже, модернізацію та вдосконалення мережі потрібно починати з них.

Модернізація апаратної складової мережі, перехід на новий сервер. Модернізація апаратної складової мережі, перехід на новий сервер

Як бачимо зі схеми топології мережі, основним сервером, який виконує роботу маршрутизації й обслуговує мережу, є сервер «edu-ua». Тому ефективність та надійність роботи мережі цілком та в основному залежить від спроможності та потужності цього серверу. Розглянемо його апаратну конфігурацію та проаналізуємо її.

Сервер «edu-ua» має таку апаратну конфігурацію:

Процесор: AMD-K6 233Mhz
Пам'ять: 62Mb 72pin simm
Материнська плата: ISA і PCI слоти, AT блок живлення
Жорсткий диск: 3.6 Gb інтерфейс IDE
Плата Digiboard PC/16e ISA для підтримки модемного пулу

Сервер працює під управлінням ОС Red Hat Linux 3.4.2-6.fc3 (Fedora Core 3) з ядром версії 2.6.9. Станом на нинішній час це дуже застаріла ОС і версія ядра. Застаріла

ОС та невеликі апаратні ресурси ведуть до неспроможності оновлювати та модернізувати програмне забезпечення на ньому, а це може призвести до потенційної небезпеки, злому зловмисниками та ненадійності роботи самого забезпечення в постійно зростаючих на сервер навантаженнях.

Як бачимо, така конфігурація, як апаратна, так і програмна, уже застаріла і потребує модернізації. Модернізація програмного забезпечення цього серверу не можлива, поки не буде проведено апаратну модернізацію, бо нове програмне забезпечення потребує більше пам'яті та потужності процесора. Тому спочатку розглянемо, як можна провести апаратну модернізацію сервера.

Є два шляхи модернізації – це апгрейд існуючої платформи або повний перехід на нову платформу. Апгрейд існуючої платформи має дуже малу ефективність та не дасть істотного приросту потужності сервера у зв'язку з тим, що материнська плата дуже застаріла і не підтримує достатньо потужні процесори та необхідну кількість пам'яті. Нова операційна система не буде задовільно працювати на старій платформі, навіть якщо поставити на неї центральний процесор, який максимально підтримує обсяг та розмір. Тому єдиний вихід із цієї ситуації — це заміна платформи. Перехід на нову платформу або новий комп'ютер — це революційний метод апгрейда, але він також має свої недоліки:

- висока ціна на нові комплектуючі, заміна платформи потребує заміни центрального процесора, пам'яті, блоку живлення та інших комплектуючих, що є достатньо коштовним;
- комплектуючі (наприклад, мультипортова плата), які переходять зі старої платформи з певних причин (у нашому випадку нова мультипортова плата коштує дуже дорого та її придбання недоцільне), повинні підтримуватись новою платформою.

У нашому випадку сервер виступає також як dial-in сервер та має у своєму складі 16-ти портову плату DigiBoard, до якої підключені модеми для телефонного Інтернет-доступу. Ця плата має ISA інтерфейс, її під'єднано до материнської плати серверу через ISA роз'єм. Це є великою проблемою, тому що такий інтерфейс дуже давно не використовується та є морально застарілим. Кількість нових портових плат DigiBoard на ринку обмежена, і коштують вони дуже дорого у зв'язку з тим, що модемний Інтернет

використовується нині дуже рідко. Тому було прийнято рішення залишити цю плату та підібрати платформу з підтримкою ISA інтерфейсу.

Така платформа була знайдена. Це intel-based сервер, побудований на материнський платі чипсету i440bx. Такий сервер має потужний tower-корпус з потужним блоком живлення, пристрій для гарячої заміни SCSI жорстких дисків з активним охолодженням. Сервер підтримує роботу з двома процесорами, але він містив тільки один 300Mhz Intel Pentium 2 процесор, три SCSI диски ємністю 3.6Gb. Так, цей сервер не є цілком сучасним, але він підтримує ISA інтерфейс, що дає можливість використання в ньому плати DigiBoard ISA, модернізації, та має підтримку таких технологій, як PCI, USB, SCSI. Материнська плата серверу має також у своєму складі один мережевий порт швидкістю 1Gbit, а також можливості моніторингу та віддаленого управління, що є дуже суттєвим для адміністрування серверною платформою.

Конфігурація нового сервера в початковому стані:

Процесор 1x300Mhz Pentium 2 (можливість встановити Pentium 3)
Пам'ять 128Mb DIMM
Плата Intel 440BX
Жорстки диски 3x SCSI 3.2 Gb
Інши комплектуючі

Після отримання сервер було модернізовано: встановлено два 450Mhz Pentium-2 процесори, пам'ять збільшено до 768Мб, встановлено додатковий жорсткий диск на 40Gb та плату DigiBoard зі старої платформи.

Конфігурація серверу після «апгрейду»:

Процесор 2x350Mhz Pentium 2
Пам'ять 528Mb DIMM
Плата Intel 440BX
Жорстки диски 3x SCSI 3.2 Gb + 1xIDE 40Gb
Інши комплектуючі DigiBoard PC/16e, 2x 100Mbit Network Card

Якщо порівнювати цю платформу з конфігурацією серверу edu-ua, то можна побачити, що потужність ЦПУ зросла понад 300%, розмір пам'яті — на 825%. Така конфігурація вже достатньо потужна, щоб обслуговувати нашу мережу під управлінням сучасної мережевої ОС та нести додаткове навантаження, таке як робота проксі-серверу, ДНС-серверу та інші.

Слід зауважити, що метою апгрейда серверу «EDU-UA» не є його заміна на надсучасну і потужну платформу. Метою модернізації є забезпечення за мінімальних

витрат виконання всіх тих функцій, які виконував старий сервер, відновити програмне забезпечення та створити можливість збільшення навантаження на сервер. Отже, отримуємо, можливо, не надсучасний, але потужніший, ніж попередній сервер, який зміг обслуговувати старе апаратне забезпечення (Digiboard PC/16e ISA) і має більше можливостей для апгрейду.

Підбір та установка ОС на модернізований сервер

Чому вибір пав саме на дистрибутив Ubuntu? Ubuntu — операційну систему для робочих станцій, лептопів і серверів, і є найпопулярнішим у світі дистрибутивом Linux. Серед основних завдань Ubuntu — надання сучасного і водночас стабільного програмного забезпечення для пересічного користувача із сильним акцентом на простоту встановлення і користування [2].

Ubuntu надає користувачеві мінімальний набір програм загального призначення: багатовіконне середовище організовано у вигляді робочого столу, є засоби для Інтернету, організації електронної пошти, офісні програми з можливістю читати і записувати файли у форматі Microsoft Office, редактор зображень, програвач компакт-дисків тощо. Спеціалізоване програмне забезпечення, необхідне досвідченішим користувачам, можна отримати з відповідних репозиторіїв. Серверний варіант системи включає також засоби, потрібні для організації веб-серверу, серверу електронної пошти тощо [2].

Інсталяційні диски Ubuntu не тільки безкоштовні, але й можуть бути безкоштовно доставлені практично у будь-яку точку світу [3].

Для інсталяції на сервер було вибрано серверну версію Ubuntu 4.2.4 з ядром версії 2.6.24 як більш стабільної і гнучкої. На сервер спочатку встановлюється основна частина системи, після чого всі необхідні пакети (архівні файлі і метадані для інсталяції і видалення архівних файлів) можуть завантажуватися і встановлюватися з Інтернету репозиторію. Така схема дуже гнучка та забезпечує користувача найновішими версіями пакетів на момент їх встановлення.

Було встановлено ядро для підтримки двох процесорів у режимі «symmetric multiprocessing».

За допомогою менеджера пакетів apt-get були доставлені всі необхідні пакети для повноцінної роботи системи в мережі.

Проблема підтримки мультипортової плати DigiBoard у новій операційній системі

Одним із ключових моментів апгрейда є підтримка старої мультипортової ISA плати DigiBoard PC/16e. Якщо з боку апаратної структури ця проблема була вирішена підтримкою шини ISA і формфактору плати новою платформою, то з боку програмного забезпечення виникли труднощі.

Один з основних недоліків плати DigiBoard PC/16e — це моральна застарілість і відмова від підтримки її виробником. Тому офіційних драйверів/модулів для цієї плати під сучасні опреційниі системи просто не існує. Але є два модулі, написані ентузіастами та розробниками ОС Linux для серії плат, до якої належить наша плата. Це модулі **pcxx.c** та **epca.c**:

Pcxx.c – Digiboard PC/X{i,e,eve} driver v1.6.3_e драйвер, перша версія якого з'явилась у 1995 році та була написана Христофором Ламетером. Драйвер працює тільки з ISA картами та має вбудовану можливість ініціювання bios-програми плати.

Epca.c – це офіційний драйвер Digiboard, який був змінений програмістами для підтримки нових ядер версій 2.6.X. Драйвер підтримує старі ISA і сучасніші PCI плати, але не має можливості ініціювання bios-програми ISA-плат. Для цього використовувалась окрема програма, яка до кодів ядра не входить.

Ці модулі підтримувались ядром до версії 2.6.11. Після цього Лінусом Товальдсом (розробник ядра Linux та засновник операційної системи Linux) було прийнято рішення видалити модулі з початкових кодів ядра у зв'язку з відсутністю підтримки з боку програмістів в адаптуванні цих модулів під нові версії ядра. Відсутність підтримки пояснюється тим, що самі плати, для яких були написані модулі, дуже застаріли та dial-in метод доступу до Інтернету, для яких вони використовувались, стає все менш популярним. Також ці модулі не підтримували роботу ядра в мультипроцесорному режимі, необхідному для використання потужності двох процесорів.

Встановлення старого ядра версії, у якому ці модулі були присутні, робить багато сервісів системи недієспроможними, оскільки вони використовують можливості сучасніших ядер. Попри це, ці модулі не розраховано на роботу в мультипроцесорному

режимі ядра системи. Тому було прийнято рішення вести доробку одного з цих модулів для роботи з поточною версією ядра. Модуль *epca*, не зважаючи на те, що він є сучаснішим, у зв'язку з відсутністю вбудованої можливості ініціювати *біос* карти, а без цього карта функціювати не буде, є менш прийнятним для доробки, тому було прийнято рішення доробити модуль *pcxx.c*.

Доробка драйвера pcxx.c для роботи в симетричному мультипроцесорному режимі ядра версії 2.6.24

Для роботи над драйвером було встановлено початкові коди ядра та пакети, які необхідні для компіляції й інсталяції ядра з початкових кодів. Ядро було сконструйовано для роботи в симетричному мультипроцесорному режимі і максимально налаштовано під апаратну конфігурацію платформи [1]. Компіляція ядра з початкових кодів — це дуже довгий процес, який на двох процесорах тривав приблизно 6 годин. Тому питанню конфігурації та налаштування ядра приділено особливу увагу. Після компіляції ядро було протестовано та встановлено як робоче для операційної системи.

Після модернізації ядра завантажено останню версію початкових кодів модуля pcxx з версії ядра, де цей модуль ще підтримувався (2.6.11). Файли модуля були зібрані в окремий каталог, і за допомогою файлу Makefile створено середовище для компілювання драйверу в режимі модуля окремо, без повторного компілювання всього ядра системи, що дало змогу заощадити час та максимально швидко проводити тестування роботи модуля [7]:

```
obj-m := pcxx.o
KDIR := ./../../linux-2.6.24/debian/build/build-generic/
PWD := $(shell pwd)
default:
 $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
```

Після створення цього файлу для компіляції драйвера достатньо було лише набрати команду make, та, якщо компіляція проходила без помилок, ми отримували готовий модуль-файл pcxx.ko. Цей файл для запуску драйвера необхідно переписати в каталог модулів поточного ядра:

```
/lib/modules/2.6.24-25-generic/kernel/drivers/char
```

і виконати команду

```
modprobe –v pcxx
```

Якщо драйвер запрацює, то в лог-файлі ми побачимо приблизно таке:

```
[ 120.936331] Digiboard PC/X{i,e,eve} driver v1.6.3_e
```

[121.589170] PC/Xx: PC/Xe (64k) I/O=0x200 Mem=0xd0000 Ports=16

Але драйвер не був готовий для роботи з новим ядром та у разі спроби компіляції видав помилки:

/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘pcxe_cleanup’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:212: error: implicit declaration of function ‘save_flags’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:213: error: implicit declaration of function ‘cli’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:222: error: implicit declaration of function ‘restore_flags’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘pcxx_waitcarrier’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:331: error: implicit declaration of function ‘sti’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: At top level:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:971: warning: initialization from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘pcxe_init’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1093: error: ‘struct tty_driver’ has no member named ‘devfs_name’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1387:41: error: macro "INIT_WORK" passed 3 arguments, but takes just 2
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1387: error: ‘INIT_WORK’ undeclared (first use in this function)
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1387: error: (Each undeclared identifier is reported only once
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1387: error: for each function it appears in.)
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘doevent’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1607: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1608: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1609: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘pcxxparam’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1769: warning: assignment from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function ‘receive_data’:
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1856: warning: assignment from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1884: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1895: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1896: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1917: error: ‘struct tty_struct’ has no member named ‘flip’
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1918: error: ‘struct tty_struct’ has no member named ‘flip’.
*make[4]: *** [/home/alexk/edu-ua.laboratory/pcxx/pcxx.o] Error 1*
*make[3]: *** [_module_/home/alexk/edu-ua.laboratory/pcxx] Error 2*
*make[2]: *** [sub-make] Error 2*
*make[1]: *** [all] Error 2*
make[1]: Leaving directory `/home/alexk/linux-2.6.24/debian/build/build-generic'
*make: *** [default] Error 2*

За допомогою пошукової системи було знайдено, що ці помилки викликані переходом нового ядра на новий тип «блокування» (операцій процесора для захисту даних, які він обробляє у поточний період, від спроб роботи з ними іншим процесором) під час роботи в мультипроцесорному режимі, і знайдено декілька документів стосовно інших схожих модулів (drivers/char/riscom8.c, drivers/char/.moxa.c тощо) про те, як цю помилку обійти та примусити старий модуль повноцінно функціонувати з новим ядром. Завдяки посиланням у цих документах в початкових кодах ядра було знайдено файл з описом, як здійснити перехід від використання механізмів блокування старого ядра до використання механізмів нового ядра, написаного Ingo Molnar [5]. Користуючись цією документацію і за аналогією з іншими модулями, драйвер *pcxx.c* для DigiBoard було перероблено з використанням механізмів блокування нового ядра.

Після модернізації драйвера залишились ще помилки, але помилки стосовно блокування cli/sti механізмом зникли:

/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:974: warning: initialization from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function 'pcxe_init':
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1096: error: 'struct tty_driver' has no member named 'devfs_name'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1390:41: error: macro "INIT_WORK" passed 3 arguments, but takes just 2
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1390: error: 'INIT_WORK' undeclared (first use in this function)
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1390: error: (Each undeclared identifier is reported only once
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1390: error: for each function it appears in.)
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function 'doevent':
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1610: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1611: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1612: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function 'pcxxparam':
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1772: warning: assignment from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c: In function 'receive_data':
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1859: warning: assignment from incompatible pointer type
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1887: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1898: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1899: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1920: error: 'struct tty_struct' has no member named 'flip'
/home/alexk/edu-ua.laboratory/pcxx/pcxx.c:1921: error: 'struct tty_struct' has no member named 'flip'
*make[4]: *** [/home/alexk/edu-ua.laboratory/pcxx/pcxx.o] Error 1*
*make[3]: *** [_module_/home/alexk/edu-ua.laboratory/pcxx] Error 2*
*make[2]: *** [sub-make] Error 2*
*make[1]: *** [all] Error 2*
make[1]: Leaving directory `/home/alexk/linux-2.6.24/debian/build/build-generic'
*make: *** [default] Error 2*

Ці помилки вже пов'язані зі структурою 'struct tty_struct' та використанням у драйвері її старої версії. Рішення цієї проблеми почалося з аналізу іншого сучаснішого модуля *epca.c*. Він уже розрахований на роботу з новою структурою 'struct tty_struct', і оскільки ці два модулі розроблено для одного обладнання, а також вони дуже схожі за структурою, то з *epca.c*. було запозичено деякі рядки коду роботи з 'struct tty_struct', а також повністю функція *receive_data*, яка ідентична до функції у pcxx за параметрами виклику та даними повернення. Компіляція з урахуванням вище наведених модифікацій успішно завершилася:

…
Building modules, stage 2.
MODPOST 1 modules
CC /home/alexk/edu-ua.laboratory/pcxx/pcxx.mod.o
LD [M] /home/alexk/edu-ua.laboratory/pcxx/pcxx.ko
…

Отже, ми отримали скомпільований під нове ядро модуль *pcxx.ko*. Після його запуску командою modprobe (як описано вище) в /var/log отримуємо такий запис:

[134.097150] Digiboard PC/X{i,e,eve} driver v1.6.3_e
[134.746852] PC/Xx: PC/Xe (64k) I/O=0x200 Mem=0xd0000 Ports=16

Цей запис показує, що драйвер правильно ідентифікував плату DigiBoard, розмір її пам’яті та кількість портів. Для того щоб остаточно переконатись в працездатності драйвера, підключаємо до портової плати DigiBoard декілька модемів, наприклад, на порти позначені 1,3,5. У системі їм будуть відповідати системні файли /dev/ttyD0, /dev/ttyD2, /dev/ttyD4 (нумерація портів у системі починається з нуля). Після підключення модемів спробуємо провести з ними обмін інформацією (на прикладі з /dev/ttyD0) за допомогою програми – терміналу «cu»:

```
alexk@doubledragon:/var/log$ cu -l ttyD0
Connected.
>ATZ
OK
>ATi9
($ZYX0304\0000DCAF\MODEM\\ U336E V 1.18 8B)

OK
```

Бачимо, що модем нам відповів — отже, обмін інформацією відбувається, а це свідчить про те, що драйвер працює.

Для того щоб драйвер запускався автоматично і максимально швидко під час завантаження системи, до параметрів ядра додаємо рядок:

```
digiepca=E,PC/Xe,D,16,200,D0000
```

Під час завантаження драйвер отримає цей рядок від ядра, розшифровує та використовує параметри замість автоматичного визначення, на яке витрачається час.

Перенесення конфігурації і налаштування сервісу «dial-in» на сервері «DoubleDragon»

Після того, як драйвер працює, можна починати організацію dial-in сервісу, який є симбіозом двох служб: це *mgetty* — служба, яка відповідає за початкове налаштування та запуск *pppd*, і сам *pppd*, який відповідає за авторизацію користувачів і забезпечує обмін інформацією між користувачем і сервером за протоколом PPP.

Установлюється стандартний пакет *mgetty*, який завантажується з репозиторію Ubuntu і встановлюється за допомогою пакет-менеджера apt-get.

З пакетом *pppd* виникає проблема, оскільки у стандартній версії пакета не реалізовано функцію call-back (зворотного дзвінка), у зв’язку з цим початкові коди пакета *pppd* були завантажені з репозиторію, налаштовані й установлені самостійно [6].

Базова версія *pppd* підтримувала тільки клієнтську версію протоколу *cbcp*, який був необхідним для забезпечення зворотного дзвінка. Для підтримки серверної версії протоколу до початкових кодів *pppd* було застосовано «патчи» (у буквальному перекладі – «латочка», заміна у вказаному(их) файлі(ах) деякої послідовності байтів) із системи ASPLinux [4]. Після цього сервіс було скомпільовано та встановлено в системі.

Після встановлення сервісів налаштовуємо їх. Пересвідчуємось, що в файлі конфігурації *mgetty* (/etc/mgetty/login.conf) є рядок:

```
/AutoPPP/ - a_ppp /usr/sbin/pppd auth -chap +pap login debug
```

Він автоматично запускає сервіс *pppd* у разі додзвону до модему та використанні протоколу ppp на клієнтській стороні. У файл /etc/inittab прописуємо команди запуску сервісів *mgetty*, які будуть очікувати на терміналах додзвону клієнтів:

```
T0:23:respawn:/sbin/mgetty -x0 -s 57600 ttyD0
T2:23:respawn:/sbin/mgetty -x0 -s 57600 ttyD2
T6:23:respawn:/sbin/mgetty -x0 -s 57600 ttyD6
```

Виконаємо команду init –Q, щоб зміни вступили в дію, і перевіримо, чи запустились необхідні процеси:

```
root@doubledragon:/etc# ps ax|grep mgetty
 4923 ? Ss 1:08 /sbin/mgetty -x0 -s 57600 ttyD2
 5098 ? Ss 0:59 /sbin/mgetty -x0 -s 57600 ttyD0
 7409 ? Ss 0:00 /sbin/mgetty -x0 -s 57600 ttyD6
```

Бачимо, що на першому (ttyD0), третьому (ttyD2) та сьомому портах (ttyD6) запущено процеси mgetty, які контролюють модеми на цих портах та чекають додзвону користувачів.

Для того щоб користувач після додзвону до сервера отримав підключення до Інтернету, необхідно налаштувати ppp (налаштування перенесені зі старого серверу «edu-ua»):

Файл /etc/ppp/options:

```
-chap
+pap
10.9.9.1:
debug
callback server
-detach
modem
crtscts
ms-dns 212.111.193.226
lock
noauth
mtu 576
mru 576
```

asyncmap 0
proxyarp
lcp-echo-failure 5
lcp-echo-interval 5

Файли callback-server (скрипт додзвону), pap-secrets (паролі користувачів), callback-users (користувачі, яким дозволено call-back) і denylist (список телефонів, на які додзвін заборонено) було перенесено із серверу edu-ua без змін, що дозволило зберегти всі налаштування і зробити оновлення серверу непомітним для користувачів.

Перевіряємо додзвін до серверу на один із модемів. Пересвідчуємось, що підключення відбувається, перевірка користувача (авторизація) проходить успішно і після підключення на клієнтській машині з'являється доступ до мережі Інтернет.

Після відключення в /var/log/messages є такі записи, які свідчать, що система працює правильно:

Mar 18 17:01:57 doubledragon pppd[11883]: pppd 2.4.4 started by a_ppp, uid 0
Mar 18 17:01:57 doubledragon pppd[11883]: using channel 22
Mar 18 17:01:57 doubledragon pppd[11883]: Using interface ppp0
Mar 18 17:01:57 doubledragon pppd[11883]: Connect: ppp0 <--> /dev/ttyD0
...
Mar 18 17:02:00 doubledragon pppd[11883]: lcp_reqci: rcvd CALLBACK
Mar 18 17:02:00 doubledragon pppd[11883]: sent [LCP ConfAck id=0x2 <asyncmap 0x0> <magic 0x6ab92b66> <pcomp> <accomp> <callback CBCP>]
Mar 18 17:02:00 doubledragon pppd[11883]: sent [LCP EchoReq id=0x0 magic=0x853f6c17]
Mar 18 17:02:00 doubledragon pppd[11883]: cbcp_lowerup
Mar 18 17:02:00 doubledragon pppd[11883]: want: 0
...
Mar 18 17:02:00 doubledragon pppd[11883]: rcvd [PAP AuthReq id=0xc user="alexkppp" password=<hidden>]
Mar 18 17:02:00 doubledragon pppd[11883]: sent [PAP AuthAck id=0xc "Login ok"]
Mar 18 17:02:00 doubledragon pppd[11883]: PAP peer authentication succeeded for alexkppp
Mar 18 17:02:00 doubledragon pppd[11883]: cbcp_open
Mar 18 17:02:00 doubledragon pppd[11883]: cbcp_sendreq cb_allowed=6
Mar 18 17:02:00 doubledragon pppd[11883]: cbcp_sendreq CONF_USER
Mar 18 17:02:00 doubledragon pppd[11883]: cbcp_sendreq CONF_NO
...
Mar 18 17:02:00 doubledragon pppd[11883]: rcvd [LCP EchoRep id=0x0 magic=0x6ab92b66]
Mar 18 17:02:02 doubledragon pppd[11883]: rcvd [CBCP Response id=0x1 < NoCallback>]
Mar 18 17:02:02 doubledragon pppd[11883]: cbcp_sendack cb_type=2
Mar 18 17:02:02 doubledragon pppd[11883]: cbcp_sendack CONF_NO
Mar 18 17:02:02 doubledragon pppd[11883]: sent [CBCP Ack id=0x1 < NoCallback>]
Mar 18 17:04:24 doubledragon pppd[11883]: Connect time 2.4 minutes.
Mar 18 17:04:24 doubledragon pppd[11883]: Sent 2749 bytes, received 16417 bytes.
Mar 18 17:04:24 doubledragon pppd[11883]: Script /etc/ppp/ip-down started (pid 11963)
Mar 18 17:04:24 doubledragon pppd[11883]: sent [LCP TermAck id=0xf]
Mar 18 17:04:24 doubledragon pppd[11883]: Script /etc/ppp/ip-down finished (pid 11963), status = 0x0
Mar 18 17:04:27 doubledragon pppd[11883]: Connection terminated.
Mar 18 17:04:27 doubledragon pppd[11883]: Modem hangup
Mar 18 17:04:27 doubledragon pppd[11883]: Exit.

Перенесення сервісів, служб і функціонала зі старого сервера на новий, фізична заміна сервера, зміни в топології мережі

На цьому етапі сервер DoubleDragon уже виконує функції «Edu-Ua» як dial-in серверу. Усі користувачі цієї послуги перенесені на новий сервер без зміни паролів. Із сервісів, які залишились на старому сервері, — це функція проксі-сервису та маршрутизація. Проксі-сервіс встановлюється стандартно за допомогою команди apt-get з репозіторію Ubuntu. Маршрутизація на старому сервері велась через один фізичний інтерфейс, який відповідає декільком логічним. Це не дуже вдала конфігурація, тому що:

- велике навантаження на інтерфейс, повільність роботи;
- пакети для різних логічних сегментів посилаються в одну фізичну мережу, що приводить до навантаження мережі та появи колізій;
- усі логічні сегменти на фізичному рівні залежать від надійності одного порту комутатора, який у разі виходу з ладу призведе до непрацездатності всіх сегментів одночасно.

Тому логічні сегменти мережі були розділені на різні фізичні середовища (див. рис. 2) за допомогою підключення до різних мережних інтерфейсів на новому сервері. Інтерфейс eth0 залишився основним та отримав адресу основного інтерфейсу старого серверу. Інтерфейси eth1 та eth2 отримали адреси логічних (віртуальних) інтерфейсів зі старого серверу.

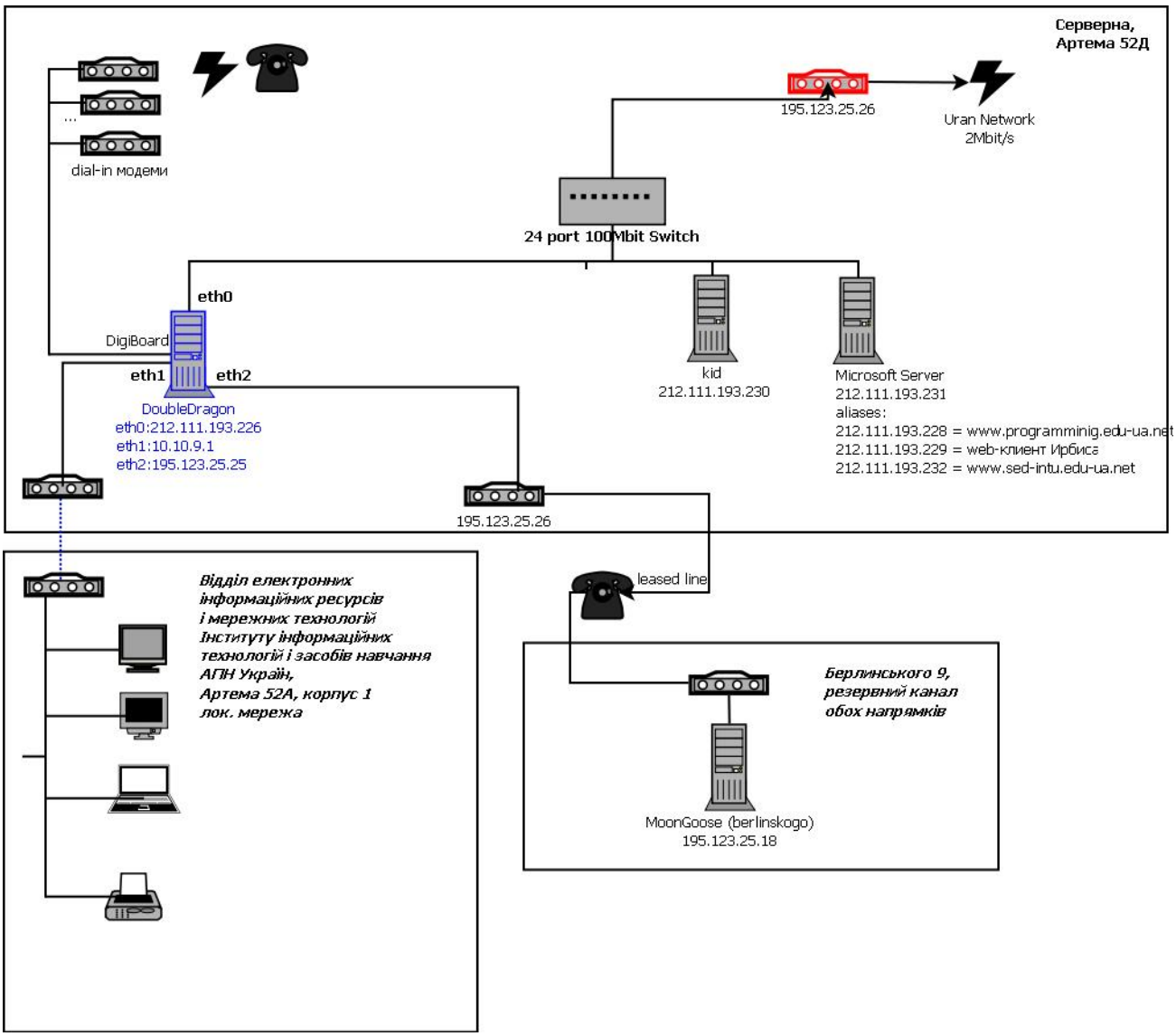

Рис. 2. Мережа після модернізації

Висновки. Проведена модернізація торкнулась як фізичної складової мережі, так і логічної її структури. Фізична модернізація або апаратний апгрейд серверу дає можливість швидше і якісніше виконувати вже покладені на нього завдання і виконувати набагато потужніші завдання. Логічна зміна в структурі мережі призвела до більш логічної її організації, що як наслідок проведеної модернізації підвищить її надійність і стійкість до аварій (якщо відмовить один сегмент, то інші будуть працювати).

Описані у статті підхід і методика модернізації серверного забезпечення і мережевої топології можуть використовуватися для модернізації та вдосконалення мереж у навчальних та наукових закладах, а також як навчальні матеріали і приклади

реальних розробок під час вивчення теорії і проведення практичних занять з таких дисциплін, як інформаційно-комп'ютерні мережі, архітектура обчислювальних систем, операційні системи.

Список використаних джерел

1. Scott Maxwell, Linux Core Kernel Commentary. — Coriolis Press, 1999.
2. Ubuntu (Матеріал з Вікіпедії — вільної енциклопедії). — [Електронний ресурс]. — Режим доступу: http://uk.wikipedia.org/wiki/Ubuntu.
3. Ubuntu Home Page. — [Електронний ресурс]. — Режим доступу: http://www.ubuntu.com/.
4. ASPLinux — разработка дистрибутива Linux, настройка Linux серверов, программы Linux, технический консалтинг. — [Електронний ресурс]. — Режим доступу: http://www.asplinux.ru/.
5. The LXR Cross Referencer, cli()/sti() removal guide, started by Ingo Molnar. — [Електронний ресурс]. — Режим доступу: http://lxr.ncu.cc/source/Documentation/cli-sti-removal.txt?v=linux-2.6.24.
6. Проект OpenNet — портал по открытому ПО, Linux, BSD и Unix системам. — [Електронний ресурс]. — http://www.opennet.ru/.
7. HOWTO compile kernel modules for the kernel 2.6. — [Електронний ресурс]. — Режим доступу: http://www.captain.at/programming/kernel-2.6/.

ПОДХОД К МОДЕРНИЗАЦИИ СЕРВЕРНОГО ОБЕСПЕЧЕНИЯ И СЕТЕВОЙ ТОПОЛОГИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Каплун А. А.

Аннотация

Проблема модернизации сетей, обновления программного и аппаратного обеспечения информационных систем в сфере образования является актуальной в современных условиях быстрого развития информационных технологий. В статье представлен анализ серверного обеспечения и сетевой топологии Интернет-центра Института информационных технологий и средств обучения НАПН Украины, предложены пути их усовершенствования. Материалы статьи отображают результаты

работы по модернизации, проведенные с целью повышения эффективности и надёжности сети, сокращения времени ожидания результатов и данных обработки в информационных системах Института при работе в сети. В статье приведены схемы сетевой топологии до модернизации и после ее проведения.

Ключевые слова: серверная операционная система, Ubuntu, драйвер Linux, ядро Linux, сетевая топология.

Upgrade for hardware/software server and network topology in information systems

Kaplun A.

Resume

The network modernization, educational information systems software and hardware updates problem is actual in modern term of information technologies prompt development. There are server applications and network topology of Institute of Information Technology and Learning Tools of National Academy of Pedagogical Sciences of Ukraine analysis and their improvement methods expound in the article. The article materials represent modernization results implemented to increase network efficiency and reliability, decrease response time in Institute's network information systems. The article gives diagrams of network topology before upgrading and after finish of optimization and upgrading processes.

Keywords: network operation systems, Ubuntu, Linux driver, Linux kernel, network topology.