УДК 004.724: 004.728

МОДЕРНИЗАЦИЯ ПРОЦЕССА ИЗМЕРЕНИЙ ИНТЕРВАЛОВ ВРЕМЕНИ В ОПЕРАЦИОННЫХ СИСТЕМАХ СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

И.П. Иванов, **А.Ю. Кондратьев**, **В.А. Лохтуров**

МГТУ им. Н.Э. Баумана, Москва
e-mail: ivanov@bmstu.ru; noc@bmstu.ru

Рассмотрена методика повышения точности измерений временных интервалов на примере утилиты ping, используемой при мониторинге локальных и глобальных компьютерных сетей.

Ключевые слова: компьютерная сеть, компьютерные технологии, утилита, модификация, операционная система, коммутатор, сервер.

MODERNIZATION OF PROCESS OF MEASURING TIME INTERVALS IN OPERATING SYSTEMS OF CONTEMPORARY COMPUTERS

I.P. Ivanov, **A.Yu. Kondratiev**, **V.A. Lokhturov**

Bauman Moscow State Technical University, Moscow
e-mail: ivanov@bmstu.ru; noc@bmstu.ru

A technique for improvement of accuracy of time-interval measurements is considered by example of the ping utility used in monitoring of local and global computer networks.

Keywords: computer network, computer technologies, utility, modification, operating system, switchboard, server.

Отличительной чертой современного состояния развития компьютерных технологий является экспоненциальное повышение производительности самих компьютеров и пропускной способности сетей, их объединяющих. Тактовые частоты процессоров лежат гигагерцовом диапазоне, а скорости передачи информации по сегментам компьютерных сетей достигают десятков гигабит в секунду [1, 2]. Этот факт требует адекватной модернизации технологий проводимых измерений интервалов времени, осуществляемых при мониторинге оконечных и транзитных узлов компьютерных сетей и их сегментов.

Одной из самых популярных утилит оценки качества работы сети вот уже на протяжении более 30 лет является утилита ping, использующая для измерений времени функцию gettimeofday( ) [3], описание которой приводится на рис. 1.

```
# man gettimeofday
…
int settimeofday(const struct timeval *tv, const struct timezone
*tz);
…
struct timeval {
   time_t      tv_sec;     /* seconds */
   suseconds_t tv_usec;    /* microseconds */
};
…
```

Рис. 1. Описание функции gettimeofday( )

Точность измерения времени этой функцией вычисляется в микросекундах, более того, в утилите ping время переводится в миллисекунды. В работе [4] отмечается недопустимость проведения измерений в компьютерных сетях в миллисекундном диапазоне уже для технологий Ethernet и Fast Ethernet, не говоря о пропускных способностях сетей Gigabit Ethernet и 10 Gigabit Ethernet. Там же предлагается модернизация утилиты ping, базирующаяся на измерении интервалов времени с точностью длительности одного интервала тактового генератора процессора, используемого для исследования компьютера. Измерения в тактах генератора конкретного процессора предполагают дальнейший пересчет в привычные единицы измерения, т.е. отсутствует свойство универсальности. Кроме того, эта модернизация утилиты возможна только для аппаратной платформы с архитектурой Intel на компьютере-измерителе и, наконец, модернизация не доведена до формы, привычной для самой утилиты ping, предполагающей статистическую обработку полученных при измерениях данных. В стандарте POSIX.1-2008 [5] функция gettimeofday( ) помечена как устаревшая и вместо нее рекомендуется использовать функцию clock_gettime( ), описанную стандартом POSIX.1-2001 [3]. На рис. 2 приведено описание этой функции.

```
# man clock_gettime
…
int clock_gettime(clockid_t clk_id, struct timespec *tp);
…
struct timespec {
   time_t   tv_sec;        /* seconds */
   long     tv_nsec;       /* nanoseconds */
};
```

Рис. 2. Описание функции clock_gettime( )

Функцией clock_gettime( ) время измеряется в наносекундах, что соизмеримо с длительностью тактов в генераторах современных про-

цессоров и длительностью бит-тайма технологии Gigabit Ethernet современных сетей.

Целью настоящей работы является повышение точности измерений утилиты ping путем полной модификации ее исходного кода, с заменой функции gettimeofday( ) на clock_gettime( ).

Для модификации исходного кода утилиты ping была выбрана операционная система Linux в реализациях: Debian для платформы IBM PowerPC [6] и Ubuntu для платформы Intel [7]. В обеих реализациях ОС для управления программным обеспечением используются пакеты формата dpkg и менеджер управления пакетами APT (Advanced Packaging Tool). Для доступа к исходным кодам утилиты ping можно воспользоваться набором команд, представленным на рис. 3.

```
# dpkg -S /bin/ping
iputils-ping: /bin/ping
# apt-get source iputils-ping
# cd iputils-20100418/
```

Рис. 3. Набор команд доступа к исходным кодам утилиты ping

В данный пакет входят версии утилиты ping для протоколов IPv4 и IPv6. Для исключения из процесса сборки утилиты версии IPv6 и подключения библиотеки, реализующей функцию clock_gettime(), был модифицирован файл Makefile пакета (рис. 4).

```
# cat Makefile
…
LDLIBS=-lrt
…
TARGETS=$(IPV4_TARGETS)
…
```

Рис. 4. Модификация файла Makefile

Исходными файлами утилиты ping из пакета iputils являются файлы ping_common.h, ping_common.c и ping.c.

Для вычисления временных параметров сети в оригинальной версии утилиты ping временной штамп, полученный функцией gettimeofday( ), помещается в поле data пакета icmp. Поскольку размер структуры timeval, которой оперирует функция gettimeofday(), совпадает с размером структуры timespec, которой оперирует функция clock_gettime(), то можно регулярно проводить замены в исходном коде функции gettimejfday() на clock_gettime(), структур timeval на timespec и их элементов tv_usec на tv_nsec.

Для размещения в поле данных и вычисления статистических результатов в коде присутствуют операции приведения к общему знаменателю полей структуры timeval - tv_sec и tv_usec. Все эти приведения должны быть модифицированы с учетом замены структуры timeval на timespec и поля tv_usec, хранящего время в миллисекундах, на поле tv_nsec, хранящее время в наносекундах.

Результаты измерений в оригинальной утилите ping переводятся в миллисекунды, в модифицированной версии этот перевод устранен.

В процессе модификации исходных файлов утилиты ping использовалась система управления версиями RCS (Revision Control System). На рис. 5 представлена модификация файла ping_common.h пакета iputils.

```
# rcsdiff -r1.1 ping_common.h
===================================================================
RCS file: ping_common.h,v
retrieving revision 1.1
diff -r1.1 ping_common.h
96c96
< extern struct timeval start_time, cur_time;
---
> extern struct timespec start_time, cur_time;
131c131
<  *    Subtract 2 timeval structs:  out = out - in.  Out is assumed to
---
>  *    Subtract 2 timespec structs:  out = out - in.  Out is assumed to
134c134
< static inline void tvsub(struct timeval *out, struct timeval *in)
---
> static inline void tvsub(struct timespec *out, struct timespec *in)
136c136
<       if ((out->tv_usec -= in->tv_usec) < 0) {
---
>       if ((out->tv_nsec -= in->tv_nsec) < 0) {
138c138
<               out->tv_usec += 1000000;
---
>               out->tv_nsec += 1000000000;
194c194
< extern int parse_reply(struct msghdr *msg, int len, void *addr, struct
timeval *);
---
> extern int parse_reply(struct msghdr *msg, int len, void *addr, struct
timespec *);
206c206
<                 int csfailed, struct timeval *tv, char *from,
---
>                 int csfailed, struct timespec *tv, char *from,
```

Рис. 5. Модификация файла ping_common.h

Ниже приведен листинг модификации файла ping_common.c пакета iputils.

```
# rcsdiff -r1.1 ping_common.c
===================================================================
RCS file: ping_common.c,v
retrieving revision 1.1
diff -r1.1 ping_common.c
1c1
< #include "ping_common.h"
- - -
> #include "ping_common.h"
29c29
< struct timeval start_time, cur_time;
- - -
> struct timespec start_time, cur_time;
262c262
< struct itimerval it;
- - -
> struct itimerspec it;
278c278
< it.it_interval.tv_usec = 0;
- - -
> it.it_interval.tv_nsec = 0;
280c280
< it.it_value.tv_usec = waittime%1000000;
- - -
> it.it_value.tv_nsec = waittime%1000000000;
300,301c300,302
< struct timeval tv;
< gettimeofday(&tv, NULL);
- - -
> struct timespec tv;
> // gettimeofday(&tv, NULL);
> clock_gettime( CLOCK_REALTIME, &tv);
303c304
< (unsigned long)tv.tv_sec, (unsigned long)tv.tv_usec);
- - -
> (unsigned long)tv.tv_sec, (unsigned long)tv.tv_nsec);
312c313
< * of the data portion are used to hold a UNIX "timeval" struct in VAX
- - -
> * of the data portion are used to hold a UNIX "timespec" struct in VAX
327c328,329
< gettimeofday(&cur_time, NULL);
- - -
```

```
> // gettimeofday(&cur_time, NULL);
> clock_gettime( CLOCK_REALTIME, &cur_time);
331c333
< struct timeval tv;
- - -
> struct timespec tv;
333c335,336
< gettimeofday(&tv, NULL);
- - -
> // gettimeofday(&tv, NULL);
> clock_gettime( CLOCK_REALTIME, &tv);
335c338
< (tv.tv_usec-cur_time.tv_usec)/1000;
- - -
> (tv.tv_nsec-cur_time.tv_nsec)/1000000;
450c453
< struct timeval tv;
- - -
> struct timespec tv;
493c496
< tv.tv_usec = 0;
- - -
> tv.tv_nsec = 0;
496c499
< tv.tv_usec = 1000 * SCHINT(interval);
- - -
> tv.tv_nsec = 1000000 * SCHINT(interval);
503c506
< tv.tv_usec = 1000*(SCHINT(interval)%1000);
- - -
> tv.tv_nsec = 1000000*(SCHINT(interval)%1000000);
524c527,528
< gettimeofday(&start_time, NULL);
- - -
> // gettimeofday(&start_time, NULL);
> clock_gettime( CLOCK_REALTIME, &start_time);
527c531
< struct itimerval it;
- - -
> struct itimerspec it;
530c534
< it.it_interval.tv_usec = 0;
- - -
```

```
> it.it_interval.tv_nsec = 0;
532c536
< it.it_value.tv_usec = 0;
- - -
> it.it_value.tv_nsec = 0;
623,624c627,628
< struct timeval *recv_timep = NULL;
< struct timeval recv_time;
- - -
> struct timespec *recv_timep = NULL;
> struct timespec recv_time;
657c661
< if (c->cmsg_len < CMSG_LEN(sizeof(struct timeval)))
- - -
> if (c->cmsg_len < CMSG_LEN(sizeof(struct timespec)))
659c663
< recv_timep = (struct timeval*)CMSG_DATA(c);
- - -
> recv_timep = (struct timespec*)CMSG_DATA(c);
666c670,671
< gettimeofday(&recv_time, NULL);
- - -
> // gettimeofday(&recv_time, NULL);
> clock_gettime( CLOCK_REALTIME, &recv_time);
692c697
< int csfailed, struct timeval *tv, char *from,
- - -
> int csfailed, struct timespec *tv, char *from,
703,704c708,709
< if (timing && cc >= 8+sizeof(struct timeval)) {
< struct timeval tmp_tv;
- - -
> if (timing && cc >= 8+sizeof(struct timespec)) {
> struct timespec tmp_tv;
709c714
< triptime = tv->tv_sec * 1000000 + tv->tv_usec;
- - -
> triptime = tv->tv_sec * 1000000000 + tv->tv_nsec;
714c719,720
< gettimeofday(tv, NULL);
- - -
> // gettimeofday(tv, NULL);
> clock_gettime( CLOCK_REALTIME, tv);
```

```
773a780,781
> printf(" time=%ld ns", triptime);
> /*
784a793
> */
792,794c801,803
< cp = ((u_char*)ptr) + sizeof(struct timeval);
< dp = &outpack[8 + sizeof(struct timeval)];
< for (i = sizeof(struct timeval); i < datalen; ++i, ++cp, ++dp) {
- - -
> cp = ((u_char*)ptr) + sizeof(struct timespec);
> dp = &outpack[8 + sizeof(struct timespec)];
> for (i = sizeof(struct timespec); i < datalen; ++i, ++cp, ++dp) {
798,800c807,809
< cp = (u_char*)ptr + sizeof(struct timeval);
< for (i = sizeof(struct timeval); i < datalen; ++i, ++cp) {
< if ((i % 32) == sizeof(struct timeval))
- - -
> cp = (u_char*)ptr + sizeof(struct timespec);
> for (i = sizeof(struct timespec); i < datalen; ++i, ++cp) {
> if ((i % 32) == sizeof(struct timespec))
832c841
< struct timeval tv = cur_time;
- - -
> struct timespec tv = cur_time;
852c861
< printf(", time %ldms", 1000*tv.tv_sec+tv.tv_usec/1000);
- - -
> printf(", time %ldms", 1000*tv.tv_sec+tv.tv_nsec/1000);
862c871
<
- - -
> /*
868a878,884
> */
> printf("rtt min/avg/max/mdev = %ld/%lu/%ld/%ld ns",
> (long)tmin,
> (unsigned long)(tsum),
> (long)tmax,
> (long)tmdev
> );
877c893
< int ipg = (1000000*(long
```

```
long)tv.tv_sec+tv.tv_usec)/(ntransmitted-1);
- - -
> int ipg = (1000000000*(long
long)tv.tv_sec+tv.tv_nsec)/(ntransmitted-1).
```

Далее представлен листинг модификации файла ping.c пакета iputils.

```
# rcsdiff -r1.1 ping.c
===================================================================
RCS file: ping.c,v
retrieving revision 1.1
diff -r1.1 ping.c
2c2
< * Copyright (c) 1989 The Regents of the University of California.
- - -
> * Copyright (c) 1989 The Regents of the University of California.
499c499
< if (datalen >= sizeof(struct timeval)) /* can we time transfer */
- - -
> if (datalen >= sizeof(struct timespec)) /* can we time transfer */
616c616
< * of the data portion are used to hold a UNIX "timeval" struct in VAX
- - -
> * of the data portion are used to hold a UNIX "timespec" struct in VAX
636,638c636,639
< static volatile int fake_fucked_egcs = sizeof(struct timeval);
< struct timeval tmp_tv;
< gettimeofday(&tmp_tv, NULL);
- - -
> static volatile int fake_fucked_egcs = sizeof(struct timespec);
> struct timespec tmp_tv;
> // gettimeofday(&tmp_tv, NULL);
> clock_gettime( CLOCK_REALTIME, &tmp_tv);
643c644
< memset(icp+1, 0, sizeof(struct timeval));
- - -
> memset(icp+1, 0, sizeof(struct timespec));
653,655c654,657
< static volatile int fake_fucked_egcs = sizeof(struct timeval);
< struct timeval tmp_tv;
< gettimeofday(&tmp_tv, NULL);
- - -
> static volatile int fake_fucked_egcs = sizeof(struct timespec);
```

```
> struct timespec tmp_tv;
> // gettimeofday(&tmp_tv, NULL);
> clock_gettime( CLOCK_REALTIME, &tmp_tv);
690c692
< parse_reply(struct msghdr *msg, int cc, void *addr, struct timeval *tv)
- - -
> parse_reply(struct msghdr *msg, int cc, void *addr, struct timespec *tv)
796,798c798,801
< struct timeval recv_time;
< gettimeofday(&recv_time, NULL);
< printf("%lu.%06lu ", (unsigned long)recv_time.tv_sec,
(unsigned long)recv_time.tv_usec);
- - -
> struct timespec recv_time;
> // gettimeofday(&recv_time, NULL);
> clock_gettime( CLOCK_REALTIME, &recv_time);
> printf("%lu.%06lu ", (unsigned long)recv_time.tv_sec,
(unsigned long)recv_time.tv_nsec);
```

Собранные для обеих платформ утилиты были названы nanoping и могут запускаться с ключами и спецификациями, идентичными таковым для утилиты ping.

Подключение систем Intel и PowerPC к корпоративной сети МГТУ им. Н.Э. Баумана осуществлялось с использованием коммутатора Cisco Catalyst, описание которого приведено ниже.

```
#show version
Cisco Internetwork Operating System Software
IOS (tm) C2950 Software (C2950-I6K2L2Q4-M), Version 12.1(22)EA13, RELEASE
SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2009 by cisco Systems, Inc.
Compiled Fri 27-Feb-09 22:20 by amvarma
Image text-base: 0x80010000, data-base: 0x80680000
ROM: Bootstrap program is C2950 boot loader

switch74_AIS uptime is 31 weeks, 3 days, 20 hours, 39 minutes
System returned to ROM by power-on
System restarted at 16:59:20 MSK Sun Aug 21 2011
System image file is "flash:/c2950-i6k2l2q4-mz.121-22.EA13.bin"
. . .
cisco WS-C2950G-48-EI (RC32300) processor (revision P0) with 19912K bytes of
memory.
Processor board ID FOC0834Y42N
```

Last reset from system-reset
Running Enhanced Image
48 FastEthernet/IEEE 802.3 interface(s)
2 Gigabit Ethernet/IEEE 802.3 interface(s)
32K bytes of flash-simulated non-volatile configuration memory.
Base ethernet MAC Address: 00:12:00:6E:AA:00
Motherboard assembly number: 73-7409-12
Power supply part number: 34-0965-01
Motherboard serial number: FOC08341RUF
Power supply serial number: DAB0833DAUL
Model revision number: P0
Motherboard revision number: A0
Model number: WS-C2950G-48-EI
System serial number: FOC0834Y42N
Configuration register is 0xF

Проверка модифицированной утилиты nanoping для платформы IBM PowerPC выполнялась на сервере, конфигурация которого имеет следующий вид:

lshw
p550-2
description: Computer
product: IBM,9133-55A
serial: IBM,030645B9H
width: 32 bits
*-core
description: Motherboard
physical id: 0
clock: 1056MHz
capabilities: ibm_9133-55a
*-firmware
product: IBM,SF240_320
physical id: 0
logical name: /proc/device-tree
*-cpu:0
description: CPU
product: POWER5+ (gs)
physical id: 2
bus info: cpu@0
version: 2.1 (pvr 003b 0201)
size: 1900MHz
clock: 1056MHz
capabilities: performance-monitor

```
*-cache
description: L1 Cache
physical id: 0
size: 32KiB
*-cpu:1
. . .
*-cpu:2
. . .
*-cpu:3
. . .
*-memory
description: System memory
physical id: 6
size: 15GiB
```

Системное программное обеспечение сервера описано на рис. 6.

```
# uname -a
Linux p550-2 2.6.32-5-powerpc64 #1 SMP Tue Mar 8 02:01:42 UTC 2011 ppc64 GNU/Linux
# lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 6.0.1 (squeeze)
Release: 6.0.1
Codename: squeeze
```

Рис. 6. Системное программное обеспечение сервера IBM PowerPC

На рис. 7 представлены результаты обмена пакетами ICMP PING с сервером ya.ru размером 64 байта с интервалом 0,1 с в количестве 1000 штук для платформы IBM PowerPC.

```
powerpc# nanoping -i 0.1 -c 1000 ya.ru
PING ya.ru (93.158.134.3) 56(84) bytes of data.
64 bytes from ya.ru (93.158.134.3): icmp_req=1 ttl=57 time=2296031 ns
64 bytes from ya.ru (93.158.134.3): icmp_req=2 ttl=57 time=2496424 ns
. . .
64 bytes from ya.ru (93.158.134.3): icmp_req=999 ttl=57 time=2297043 ns
64 bytes from ya.ru (93.158.134.3): icmp_req=1000 ttl=57 time=3800034 ns
- - - ya.ru ping statistics - - -
1000 packets transmitted, 1000 received, 0% packet loss, time 100568ms
rtt min/avg/max/mdev = 1293628/2116858/5042580/504256 ns
```

Рис. 7. Пример использования утилиты nanoping для сервера IBM PowerPC

Для сравнения на рис. 8 приведены результаты обмена пакетами ICMP PING с этим же сервером обычной утилитой ping с теми же опциями, что и nanoping.

```
powerpc# ping -i 0.1 -c 1000 ya.ru
PING ya.ru (93.158.134.3) 56(84) bytes of data.
64 bytes from ya.ru (93.158.134.3): icmp_req=1 ttl=57 time=2.63 ms
64 bytes from ya.ru (93.158.134.3): icmp_req=2 ttl=57 time=3.00 ms
. . .
64 bytes from ya.ru (93.158.134.3): icmp_req=999 ttl=57 time=2.53 ms
64 bytes from ya.ru (93.158.134.3): icmp_req=1000 ttl=57 time=2.04 ms
- - - ya.ru ping statistics - - -
1000 packets transmitted, 997 received, 0% packet loss, time 100490ms
rtt min/avg/max/mdev = 1.283/2.351/5.041/0.686 ms
```

Рис. 8. Пример использования утилиты ping для сервера IBM PowerPC

Проверка модифицированной утилиты nanoping для платформы Intel осуществлялась на компьютере (PC), конфигурация которого имеет следующий вид:

```
intell# lshw
ms.bmstu.ru
description: Desktop Computer
product: GEG
vendor: KRAFTWAY
serial: 10610023
width: 32 bits
capabilities: smbios-2.4 dmi-2.4 smp-1.4 smp
configuration: boot=normal chassis=desktop cpus=4
uuid=00000000-0000-0000-0000-001D7D0C062F
*-core
description: Motherboard
product: X48-DS4
vendor: Gigabyte Technology Co., Ltd.
physical id: 0
version: x.x
*-firmware
description: BIOS
vendor: Award Software International, Inc.
physical id: 0
version: F2 (07/17/2008)
size: 128KiB
capacity: 960KiB
capabilities: pci pnp apm upgrade shadowing cdboot bootselect edd int13floppy360
int13floppy1200 int13floppy720 int13floppy2880 int5printscreen int9keyboard int14serial
```

int17printer int10video acpi usb ls120boot zipboot biosbootspecification
*-cpu:0
description: CPU
product: Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz
vendor: Intel Corp.
physical id: 4
bus info: cpu@0
version: 6.15.11
serial: 0000-06FB-0000-0000-0000-0000
slot: Socket 775
size: 1600MHz
capacity: 4GHz
width: 64 bits
clock: 266MHz
capabilities: boot fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe nx x86-64 constant_tsc arch_perfmon pebs bts aperfmperf pni dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr pdcm lahf_lm tpr_shadow vnmi flexpriority cpufreq
configuration: id=3
*-cache:0
description: L1 cache
physical id: a
slot: Internal Cache
size: 64KiB
capacity: 64KiB
capabilities: synchronous internal write-back
*-cache:1
description: L2 cache
physical id: b
slot: External Cache
size: 4MiB
capabilities: synchronous internal write-back
*-logicalcpu:0
description: Logical CPU
physical id: 3.1
width: 64 bits
capabilities: logical
*-logicalcpu:1
. . .
*-logicalcpu:2
. . .
*-logicalcpu:3
. . .

*-memory
description: System Memory
physical id: 19
slot: System board or motherboard
size: 4GiB . . .

Описание системного программного обеспечения этого компьютера приведено на рис. 9.

intell# uname -a
Linux ms.bmstu.ru 2.6.32-21-generic-pae #32-Ubuntu SMP Fri Apr 16 09:39:35 UTC 2010 i686 GNU/Linux
intell# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 10.04.3 LTS
Release: 10.04
Codename: lucid

Рис. 9. Системное ПО для PC платформы Intel

Результаты обмена пакетами ICMP PING с сервером ya.ru утилитой nanoping для PC платформы Intel представлена на рис. 10.

intell# nanoping -i 0.1 -c 1000 ya.ru
PING ya.ru (93.158.134.3) 56(84) bytes of data.
64 bytes from ya.ru (93.158.134.3): icmp_req=1 ttl=58 time=2310666 ns
64 bytes from ya.ru (93.158.134.3): icmp_req=2 ttl=58 time=2059204 ns
. . .
64 bytes from ya.ru (93.158.134.3): icmp_req=999 ttl=58 time=2298258 ns
64 bytes from ya.ru (93.158.134.3): icmp_req=1000 ttl=58 time=2179567 ns
- - - ya.ru ping statistics - - -
1000 packets transmitted, 1000 received, 0% packet loss, time 100546ms
rtt min/avg/max/mdev = 1201137/2003100/4636744/481802 ns

Рис. 10. Пример использования утилиты nanoping для Intel PC

Для сравнения ниже приведены результаты обмена пакетами ICMP PING с этим же сервером обычной утилитой ping. В обоих случаях опции утилит выбирались идентичными таковым для платформы IBM PowerPC.

intell# ping -i 0.1 -c 1000 ya.ru
PING ya.ru (93.158.134.3) 56(84) bytes of data.
64 bytes from ya.ru (93.158.134.3): icmp_seq=1 ttl=58 time=2.05 ms
64 bytes from ya.ru (93.158.134.3): icmp_seq=2 ttl=58 time=3.85 ms
. . .
64 bytes from ya.ru (93.158.134.3): icmp_seq=999 ttl=58 time=1.95 ms

64 bytes from ya.ru (93.158.134.3): icmp_seq=1000 ttl=58 time=2.96 ms
- - - ya.ru ping statistics - - -
1000 packets transmitted, 1000 received, 0% packet loss, time 100510ms
rtt min/avg/max/mdev = 1.152/2.230/5.864/0.672 ms

Сравнение результатов измерений времени обмена ICMP пакетами для утилит nanoping и ping свидетельствует о полной работоспособности проведенных модернизаций исходных кодов файлов ping_common.h, ping_common.c и ping.c. Использование предлагаемой утилиты nanoping дает возможность повысить точность измерений интервалов времени RTT (Round Trip Time) минимум на 3 порядка, что весьма желательно при мониторинге современных локальных вычислительных сетей.

Применение функции clock_gettime( ) может быть рекомендовано и для модернизации других утилит в ОС современных компьютеров при замерах интервалов времени.

СПИСОК ЛИТЕРАТУРЫ

1. Таненбаум Э. Компьютерные сети. – СПб.: Питер, 2009. – 992 с.
2. Олифер В. Г., Олифер Н. А. Компьютерные сети. – СПб.: Питер, 2011. – 944 с.
3. 1003.1-2001 — IEEE Standard for Information Technology — Portable Operating System Interface (POSIX(R)).
4. Иванов И. П. Математические модели коммутаторов локальных вычислительных сетей // Вестник МГТУ. Сер. Приборостроение. – 2009. – № 2. – С. 84–92.
5. 1003.1-2008 — IEEE Standard for Information Technology — Portable Operating System Interface (POSIX(R)).
6. Бойченко М. К., Иванов И. П., Кондратьев А. Ю. Доступность ресурсов транспортных подсистем корпоративных сетей // Вестник МГТУ. Сер. Приборостроение. – 2010. – № 3. – С. 103–118.
7. Филимонов А. Ю. Построение мультисервисных сетей Ethernet. – СПб.: БХВ, 2007. – 592 с.

Статья поступила в редакцию 30.05.2012