Inside netgraph, system tuning

Dmitry S. Luhtionov
dmitryluhtionov@gmail.com

KyivBSD, 2013

Создание простой netgraph ноды

Создаем ноду ng_mynode
Простейший заголовок ng_mynode.h

```
/* Node type name and magic cookie */
#define NG_MYNODE_NODE_TYPE     "mynode"
#define NGM_MYNODE_COOKIE    884298942

/* Hook names */
#define NG_MYNODE_HOOK_UPPER      "upper"
#define NG_MYNODE_HOOK_LOWER      "lower"
```

Тип ноды — используется для идентификации ноды.
Не должен содержать '.' или ':', и ограничен по длине до NG_NODESIZ символов (включая терминирующий символ '\0').
COOKIE — версия ноды. Получается из date -u +%s

Простейший файл ng_mynode.c

```
#include <sys/param.h>
#include <sys/systm.h>
#include <sys/kernel.h>
#include <sys/malloc.h>
#include <sys/mbuf.h>
#include <netgraph/ng_message.h>
#include <netgraph/netgraph.h>
#include <netgraph/ng_mynode.h>

/* Netgraph methods */
static ng_constructor_t ng_mynode_constructor;
static ng_rcvmsg_t      ng_mynode_rcvmsg;
static ng_shutdown_t    ng_mynode_shutdown;
static ng_newhook_t     ng_mynode_newhook;
static ng_rcvdata_t     ng_mynode_rcvdata;
static ng_disconnect_t  ng_mynode_disconnect;
```

Возможные методы netgraph

- ng_constructor_t — Конструктор ноды
- ng_rcvmsg_t — Принимает управляющие сообщения
- ng_close_t — Завершает соединения
- ng_shutdown_t — Деструктор ноды
- ng_newhook_t — Создание нового хука
- ng_findhook_t — Поиск хука (ng_ppp)
- ng_connect_t — Соединение хука
- ng_rcvdata_t — Прием данных
- ng_disconnect_t — Отсоединение хука

Методы вызываются при возникновении событий в системе netgraph

Очередность событий:
constuctor — newhook — connect — работа — close — disconnect — shutdown

Экспортируемая статистика

```
struct ng_mynode_stats {
    uint64_t        received;
    uint64_t        errors;
};
```

Внутренние переменные ноды

```
struct ng_mynode_priv {
    hook_p          upper;
    hook_p          lower;
    struct ng_mynode_stats stats;
};
typedef struct ng_mynode_priv *priv_p;
```

Заполняем поля структуры и регистрируем ноду в подсистеме netgraph

```
static struct ng_type typestruct = {
    .version =      NG_ABI_VERSION,
    .name =         NG_MYNODE_NODE_TYPE,
    .constructor =  ng_mynode_constructor,
    .rcvmsg =       ng_mynode_rcvmsg,
    .shutdown =     ng_mynode_shutdown,
    .newhook =      ng_mynode_newhook,
    .rcvdata =      ng_mynode_rcvdata,
    .disconnect =   ng_mynode_disconnect,
};
NETGRAPH_INIT(patch, &typestruct);
```

Конструктор ноды

```
static int
ng_mynode_constructor(node_p node)
{
    priv_p privdata;

    privdata = malloc(sizeof(*privdata),
M_NETGRAPH, M_WAITOK | M_ZERO);
    NG_NODE_SET_PRIVATE(node, privdata);
    privdata->upper = NULL;
    privdata->lower = NULL;
    privdata->stats.received = 0;
    privdata->stats.errors = 0;
    return (0);
}
```

Макрос устанавливает значение (node)->nd_private

Деструктор ноды

```
static int
ng_mynode_shutdown(node_p node)
{
   const priv_p privdata =
NG_NODE_PRIVATE(node);

   NG_NODE_SET_PRIVATE(node, NULL);
   NG_NODE_UNREF(node);
   free(privdata, M_NETGRAPH);
   return (0);
}
```

Макрос устанавливает значение (node)->nd_private

Новый хук

```
static int
ng_mynode_newhook(node_p node, hook_p hook, const
char *name)
{
    const priv_p privdata = NG_NODE_PRIVATE(node);

    if (strncmp(name, NG_MYNODE_HOOK_UPPER,
strlen(NG_MYNODE_HOOK_UPPER)) == 0) {
        privdata->upper = hook;
    } else if (strncmp(name, NG_MYNODE_HOOK_LOWER,
strlen(NG_MYNODE_HOOK_LOWER)) == 0) {
        privdata->lower = hook;
    } else
        return (EINVAL);
    return(0);
}
```

Отсоединение хука

```
ng_mynode_disconnect(hook_p hook)
{
    priv_p privdata;

    privdata = NG_NODE_PRIVATE(NG_HOOK_NODE(hook));
    if (hook == privdata->upper)
        privdata->upper = NULL;
    if (hook == privdata->lower)
        privdata->lower = NULL;
    if (NG_NODE_NUMHOOKS(NG_HOOK_NODE(hook)) == 0
    /* already shutting down? */
    && NG_NODE_IS_VALID(NG_HOOK_NODE(hook)))
        ng_rmnode_self(NG_HOOK_NODE(hook));
    return (0);
}
```

Макрос проверяет (node)->nd_flags & NGF_INVALID

Прием и отправка пакета

```
static int
ng_mynode_rcvdata(hook_p hook, item_p item)
{
    const priv_p privdata =
NG_NODE_PRIVATE(NG_HOOK_NODE(hook));
    int error;

    privdata->stats.received++;
    NG_FWD_ITEM_HOOK(error, item, hook);
    if (error != 0)
        privdata->stats.errors++;
    return (error);
}
```

Макрос пересылает данные. Для новых данных используем макрос NG_FWD_NEW_DATA(error, item, hook, m);
Где m — *mbuf

Прием и отправка сообщений

```
static int
ng_mynode_rcvmsg(node_p node, item_p item, hook_p
hook)
{
    struct ng_mesg *msg, *resp = NULL;
    const priv_p privdata = NG_NODE_PRIVATE(node);
    int error = NULL;

    NGI_GET_MSG(item, msg);
    switch (msg->header.typecookie) {
    case NGM_GENERIC_COOKIE:
        switch (msg->header.cmd) {
        case NGM_TEXT_STATUS:
```

В структуре item_p есть union «body», в котором могут передаваться mbuf (данные), msg (сообщение), fn (пользовательская функция)

NGM_GENERIC_COOKIE — стандартные сообщения
- NGM_SHUTDOWN — Удалить ноду
- NGM_MKPEER — Создать и присоединить соседнюю ноду
- NGM_CONNECT — Соединить две ноды
- NGM_NAME — Дать ноде имя
- NGM_RMHOOK — Разорвать соединение между нодами
- NGM_NODEINFO — Вывести информацию о ноде
- NGM_LISTHOOK — Вывести список хуков у ноды
- NGM_LISTNAMES — Вывести список нод с именем
- NGM_LISTNODES — Вывести список всех нод
- NGM_LISTTYPES — Вывести типы установленных нод
- NGM_TEXT_STATUS — Статус ноды в текстовом виде
- NGM_BINARY2ASCII — Преобразовать ng_mesg в ASCII
- NGM_ASCII2BINARY — Преобразовать ASCII в ng_mesg
- NGM_TEXT_CONFIG — Получить/установить конфиг

NGM_FLOW_COOKIE — управление данными/потоком
Структуры сообщений описаны в файле ng_message.h

```
{
    char *arg;
    int pos;

    NG_MKRESPONSE(resp, msg, NG_TEXTRESPONSE,
M_NOWAIT);
    if (resp == NULL) {
        error = ENOMEM;
        break;
    }
    arg = (char *) resp->data;
    pos = sprintf(arg, "recv: %llu, errs: %llu",
privdata->stats.received, privdata->stats.errors);
    resp->header.arglen = pos + 1;
    break;
}
NG_RESPOND_MSG(error, node, item, resp);
NG_FREE_MSG(msg);
return (error);
```

Создаем свою собственную команду
Файл ng_mynode.h

```
/* Netgraph commands understood by this node type */
enum {
    NGM_MYNODE_GET_STATS = 1,
    NGM_MYNODE_CLR_STATS,
    NGM_MYNODE_GETCLR_STATS
};

#define NG_MYNODE_STATS_TYPE_INFO {   \
    { "received", &ng_parse_uint64_type }, \
    { "errors", &ng_parse_uint64_type }, \
    { NULL } \
}
```

Файл ng_mynode.c

```
static const struct ng_parse_struct_field
ng_mynode_stats_fields[]
    = NG_MYNODE_STATS_TYPE_INFO;
static const struct ng_parse_type
ng_mynode_stats_type = {
    &ng_parse_struct_type,
    &ng_mynode_stats_fields
};
static const struct ng_cmdlist ng_mynode_cmdlist[] =
{
    {
        NGM_MYNODE_COOKIE,
        NGM_MYNODE_GET_STATS,
        "getstats",
        NULL,
        &ng_mynode_stats_type
    }
...
```

Файл ng_mynode.c

```
static struct ng_type typestruct = {
    .version =      NG_ABI_VERSION,
...
    .disconnect =   ng_mynode_disconnect,
    .cmdlist =      ng_mynode_cmdlist,
};
...
static int
ng_mynode_rcvmsg(node_p node, item_p item, hook_p
hook)
{
...
    case NGM_MYNODE_COOKIE:
        switch (msg->header.cmd) {
        case NGM_MYNODE_GET_STATS:
        case NGM_MYNODE_GETCLR_STATS:
        {
```

Безопасно удаляем ноду
Файл ng_mynode.c

```
static struct ng_type typestruct = {
    .version =      NG_ABI_VERSION,
...
    .cmdlist =      ng_mynode_cmdlist,
    .close =        ng_mynode_close,
};
...
ng_mynode_close(node_p node)
{
    const priv_p privdata = NG_NODE_PRIVATE(node);

    if (privdata->upper && privdata->lower)
        ng_bypass(privdata->upper, privdata->lower);
    return (0);
}
```

Tuning

Выделение памяти
Смотрим колонку FAILURES

```
# vmstat -z
ITEM                   SIZE     LIMIT       USED
...
NetGraph items:          36,      4134,         11,
NetGraph data items:     36,       546,          0,
```

Увеличиваем размер выделяемой памяти

```
kern.ipc.nmbclusters: 128000
kern.ipc.maxsockbuf: 2097152
net.graph.maxdata: 65536
net.graph.maxalloc: 65536
net.graph.recvspace: 65536
net.graph.maxdgram: 65536
```

Уменьшаем лишнюю обработку при прохождении сетевых пакетов. Убираем процессы ng_queue с лишних CPU

```
kern.random.sys.harvest.ethernet: 0
kern.random.sys.harvest.point_to_point: 0
kern.random.sys.harvest.interrupt: 0
net.graph.threads: 4

/usr/bin/cpuset -l 0-3 -p 12
/usr/bin/cpuset -l 4 -x 256
/usr/bin/cpuset -l 5 -x 257
/usr/bin/cpuset -l 6 -x 259
/usr/bin/cpuset -l 7 -x 260
```

Pid 12 — ng_queue
Irq256 — em0:rx
Irq257 — em0:tx
Irq259 — em1:rx
Irq260 — em1:tx

Средняя нагрузка на сервер

```
CPU 0:  1.6% user,  0.0% nice, 12.6% system,  0.4% interrupt, 85.4% idle
CPU 1:  2.0% user,  0.0% nice, 12.6% system,  0.0% interrupt, 85.4% idle
CPU 2:  1.2% user,  0.0% nice, 14.6% system,  0.0% interrupt, 84.3% idle
CPU 3:  0.8% user,  0.0% nice, 10.6% system,  0.0% interrupt, 88.6% idle
CPU 4:  0.4% user,  0.0% nice,  1.6% system, 15.7% interrupt, 82.3% idle
CPU 5:  1.2% user,  0.0% nice,  2.4% system,  3.5% interrupt, 92.9% idle
CPU 6:  0.8% user,  0.0% nice,  0.8% system, 26.4% interrupt, 72.0% idle
CPU 7:  0.4% user,  0.0% nice,  2.0% system,  3.5% interrupt, 94.1% idle


root@117-14:/home/mitya # netstat -w 1 -h
            input         (Total)           output
   packets  errs idrops      bytes    packets  errs      bytes colls
      129k     0     0        86M       134k     0       117M     0
      114k     0     0        81M       122k     0       108M     0
      108k     0     0        75M       114k     0        97M     0
```

Вопросы?