Network Address Translation (NAT): Cara lain menghemat IP *Address*

Tito Sugiharta
Laboratorium Sistem Informasi & Keputusan (LSIK)
Teknik Industri ITB
Tito@TI.ITB.ac.id

Misi awal Internet adalah sebagai jaringan komunikasi non-profit. Pada awalnya, Internet didesain tanpa memperhatikan dunia bisnis. Kemudian hal ini menjadi masalah sekarang dan di masa depan. Dengan semakin banyaknya penghuni Internet, baik pencari informasi maupun penyedia informasi, maka kebutuhan akan pengalamatan di Internet makin membengkak. Kebutuhan besar akan IP *address* biasanya terjadi di jaringan komputer perusahaan dan LAN-LAN di lembaga pendidikan.

IP *address* sebagai sarana pengalamatan di Internet semakin menjadi barang mewah dan ekslusif. Tidak sembarang orang sekarang ini bisa mendapatkan IP *address* yang valid dengan mudah. Oleh karena itulah dibutuhkan suatu mekanisme yang dapat menghemat IP *address*. Logika sederhana untuk penghematan IP *address* ialah dengan meng-*share* suatu nomor IP *address* valid ke beberapa *client* IP lainnya. Atau dengan kata lain beberapa komputer bisa mengakses Internet walau kita hanya memiliki satu IP *address* yang valid. Salah satu Mekanisme itu disediakan oleh *Network Address Translation* (NAT)

Beberapa Konsep Dasar

Sebelum kita membahas lebih lanjut ada baiknya kita urai kembali konsep-konsep dasar yang harus dipahami sebelum masuk ke NAT. Diantaranya adalah TCP/IP, *Gateway/Router*, dan *Firewall*.

TCP/IP

Protokol yang menjadi standar dan dipakai hampir oleh seluruh komunitas Internet adalah TCP/IP (*Transmission Control Protocol/Internet Protocol*). Agar komputer bisa berkomunikasi dengan komputer lainnya, maka menurut aturan TCP/IP, komputer tersebut harus memiliki suatu *address* yang unik. Alamat tersebut dinamakan IP *address*. IP *Address* memiliki format sbb: *aaa.bbb.ccc.ddd*. Contohnya: 167.205.19.33

Yang penting adalah bahwa untuk berkomunikasi di Internet, komputer harus memiliki IP *address* yang legal. Legal dalam hal ini artinya adalah bahwa alamat tersebut dikenali oleh semua *router* di dunia dan diketahui bahwa alamat tersebut tidak ada duplikatnya di tempat lain. IP *address* legal biasanya diperoleh dengan menghubungi InterNIC.

Suatu jaringan internal bisa saja menggunakan IP *address* sembarang. Namun untuk tersambung ke Internet, jaringan itu tetap harus menggunakan IP *address* legal. Jika masalah *routing* tidak dibereskan (tidak menggunakan IP *address* legal), maka saat sistem kita mengirim paket data ke sistem lain, sistem tujuan itu tidak akan bisa mengembalikan paket data tersebut, sehingga komunikasi tidak akan terjadi.

Dalam berkomunikasi di Internet/antar jaringan komputer dibutuhkan *gateway/router* sebagai jembatan yang menghubungkan simpul-simpul antar jaringan sehingga paket data bisa diantar sampai ke tujuan.

Gateway/Router

Gateway adalah komputer yang memiliki minimal 2 buah *network interface* untuk menghubungkan 2 buah jaringan atau lebih. Di Internet suatu alamat bisa ditempuh lewat *gateway-gateway* yang memberikan jalan/rute ke arah mana yang harus dilalui supaya paket data sampai ke tujuan. Kebanyakan *gateway* menjalankan *routing daemon* (program yang meng-*update* secara dinamis tabel *routing*). Karena itu *gateway* juga biasanya berfungsi sebagai *router*. *Gateway/router* bisa berbentuk *Router box* seperti yang di produksi Cisco, 3COM, dll atau bisa juga berupa komputer yang menjalankan *Network Operating System* plus *routing daemon*. Misalkan PC yang dipasang **Unix FreeBSD** dan menjalankan program `Routed` atau `Gated`. Namun dalam pemakaian `Natd`, *routing daemon* tidak perlu dijalankan, jadi cukup dipasang *gateway* saja.

Karena *gateway/router* mengatur lalu lintas paket data antar jaringan, maka di dalamnya bisa dipasangi mekanisme pembatasan atau pengamanan (*filtering*) paket-paket data. Mekanisme ini disebut *Firewall*.

Firewall

Sebenarnya *Firewall* adalah suatu program yang dijalankan di *gateway/router* yang bertugas memeriksa setiap paket data yang lewat kemudian membandingkannya dengan *rule* yang diterapkan dan akhirnya memutuskan

apakah paket data tersebut boleh diteruskan atau ditolak. Tujuan dasarnya adalah sebagai *security* yang melindungi jaringan internal dari ancaman dari luar. Namun dalam tulisan ini *Firewall* digunakan sebagai basis untuk menjalankan *Network Address Translation* (NAT).

Dalam FreeBSD, program yang dijalankan sebagai *Firewall* adalah `ipfw`. Sebelum dapat menjalankan `ipfw`, *kernel* `GENERIC` harus dimodifikasi supaya mendukung fungsi *firewall*. `Ipfw` mengatur lalu lintas paket data berdasarkan IP asal, IP tujuan, nomor *port*, dan jenis *protocol*. Untuk menjalankan NAT, `option IPDIVERT` harus diaktifkan dalam *kernel*.

DIVERT (mekanisme diversi paket *kernel*)

Socket divert sebenarnya sama saja dengan *socket* IP biasa, kecuali bahwa *socket divert* bisa di *bind* ke *port divert* khusus lewat *bind system call*. IP *address* dalam *bind* tidak diperhatikan, hanya nomor *port*-nya yang diperhatikan. Sebuah *socket divert* yang di*bind* ke *port divert* akan menerima semua paket yang didiversikan pada *port* tersebut oleh mekanisme di *kernel* yang dijalankan oleh implementasi *filtering* dan program `ipfw`. Mekanisme ini yang dimanfaatkan nantinya oleh *Network Address Translator*.

Itulah beberapa bahasan awal yang akan mengantar kita ke pembahasan inti selanjutnya.

Network Address Translation (NAT)

Dalam FreeBSD, mekanisme *Network Address Translation* (NAT) dijalankan oleh program `Natd` yang bekerja sebagai *daemon*. *Network Address Translation Daemon* (`Natd`) menyediakan solusi untuk permasalahan penghematan ini dengan cara menyembunyikan IP *address* jaringan internal, dengan membuat paket yang di-*generate* di dalam terlihat seolah-olah dihasilkan dari mesin yang memiliki IP *address* legal. `Natd` memberikan konektivitas ke dunia luar tanpa harus menggunakan IP *address* legal dalam jaringan internal.

`Natd` menyediakan fasilitas *Network Address Translation* untuk digunakan dengan *socket divert*. `Natd` mengubah semua paket yang ditujukan ke *host* lain sedemikian sehingga *source* IP *address*nya berasal dari mesin `Natd`. Untuk setiap paket yang diubah berdasarkan aturan ini, dibuat tabel translasi untuk mencatat transaksi ini.

Dengan NAT, aturan bahwa untuk berkomunikasi harus menggunakan IP *address* legal, dilanggar.NAT bekerja dengan jalan mengkonversikan IP-IP *address* ke satu atau lebih IP *address* lain. IP *address* yang dikonversi adalah IP *address* yang diberikan untuk tiap mesin dalam jaringan internal (bisa sembarang IP). IP *address* yang menjadi hasil konversi terletak di luar jaringan internal tersebut dan merupakan IP *address* legal yang valid/*routable*.

Mekanisme NAT

Sebuah paket TCP terdiri dari *header* dan data. *Header* memiliki sejumlah *field* di dalamnya, salah satu *field* yang penting di sini adalah MAC (*Media Access Control*) *address* asal dan tujuan, IP *address* asal dan tujuan, dan nomor *port* asal dan tujuan.

Saat mesin A menghubungi mesin B, *header* paket berisi IP A sebagai IP *address* asal dan IP B sebagai IP *address* tujuan. *Header* ini juga berisi nomor *port* asal (biasanya dipilih oleh mesin pengirim dari sekumpulan nomor *port*) dan nomor *port* tujuan yang spesifik, misalnya *port* 80 (untuk *web*).

Kemudian B menerima paket pada *port* 80 dan memilih nomor *port* balasan untuk digunakan sebagai nomor *port* asal menggantikan *port* 80 tadi. Mesin B lalu membalik IP *address* asal & tujuan dan nomor *port* asal & tujuan dalam *header* paket. Sehingga keadaan sekarang IP B adalah IP *address* asal dan IP A adalah IP *address* tujuan. Kemudian B mengirim paket itu kembali ke A. Selama *session* terbuka, paket data hilir mudik menggunakan nomor *port* yang dipilih.

Router (yang biasa – tanpa `Natd`) memodifikasi *field* MAC *address* asal & tujuan dalam *header* ketika me-*route* paket yang melewatinya. IP *address*, nomor *port*, dan nomor *sequence* asal & tujuan tidak disentuh sama sekali.

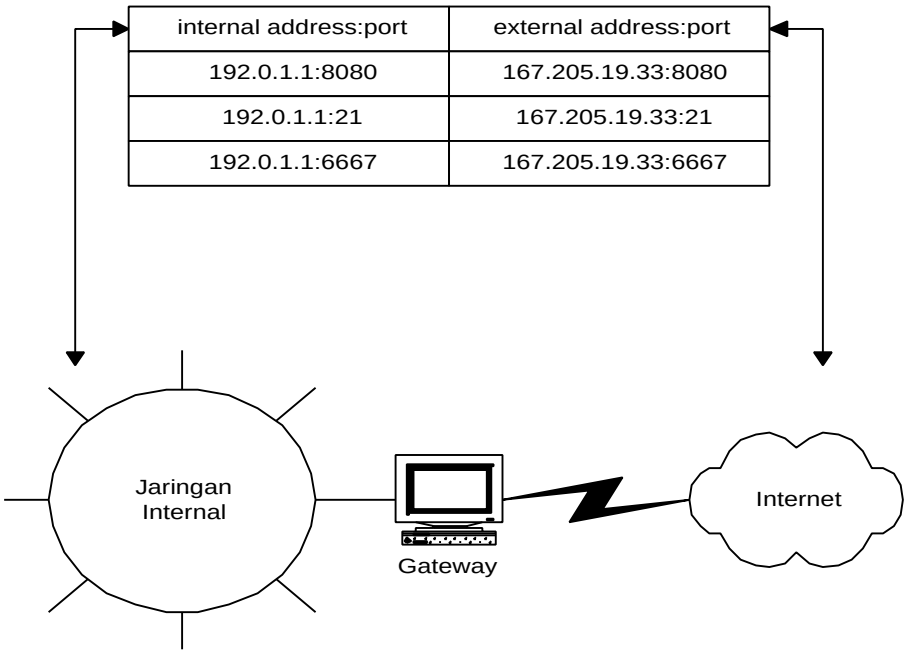
NAT juga bekerja atas dasar ini. Dimulai dengan membuat tabel translasi internal untuk semua IP *address* jaringan internal yang mengirim paket melewatinya. Lalu men-*set* tabel nomor *port* yang akan digunakan oleh IP *address* yang valid. Ketika paket dari jaringan internal dikirim ke `Natd` untuk disampaikan keluar, `Natd` melakukan hal-hal sebagai berikut:

1. Mencatat IP *address* dan *port* asal dalam tabel translasi
2. Menggantikan nomor IP asal paket dengan nomor IP dirinya yang valid
3. Menetapkan nomor *port* khusus untuk paket yang dikirim keluar, memasukkannya dalam tabel translasi dan menggantikan nomor *port* asal tersebut dengan nomor *port* khusus ini.

Ketika paket balasan datang kembali, `Natd` mengecek nomor *port* tujuannya. Jika ini cocok dengan nomor *port* yang khusus telah ditetapkan sebelumnya, maka dia akan melihat tabel translasi dan mencari mesin mana di

jaringan internal yang sesuai. Setelah ditemukan, ia akan menulis kembali nomor *port* dan IP *address* tujuan dengan IP *address* dan nomor *port* asal yang asli yang digunakan dulu untuk memulai koneksi. Lalu mengirim paket ini ke mesin di jaringan internal yang dituju. `Natd` memelihara isi tabel translasi selama koneksi masih terbuka.

Gambar Contoh Mekanisme `Natd`



Perbedaan dengan sistem *Proxy*

Hampir mirip dengan NAT, suatu jaringan kecil dengan *proxy* bisa menempatkan beberapa mesin untuk mengakses *web* dibelakang sebuah mesin yang memiliki IP *address* valid. Ini juga merupakan langkah penghematan biaya dibanding harus menyewa beberapa account dari ISP dan memasang modem & sambungan telepon pada tiap mesin.

Namun demikian, *proxy* server ini tidak sesuai untuk jaringan yang lebih besar. Bagaimanapun, menambah *hard disk* dan RAM pada server *proxy* supaya *proxy* berjalan efisien tidak selalu dapat dilakukan (karena *constraint* biaya). Lagi pula, persentase *web page* yang bisa dilayani oleh *cache proxy* akan makin menurun sejalan dengan semakin menipisnya ruang kosong di *hard disk*, sehingga penggunaan *cache proxy* menjadi tidak lebih baik dari pada sambungan langsung. Tambahan lagi, tiap koneksi bersamaan akan meng-*generate* proses tambahan dalam *proxy*. Tiap proses ini harus menggunakan *disk I/O channel* yang sama, dan saat *disk I/O channel* jenuh, maka terjadilah *bottle neck*.

NAT menawarkan solusi yang lebih fleksibel dan *scalable*. NAT menghilangkan keharusan mengkonfigurasi *proxy/sock* dalam tiap *client*. NAT lebih cepat dan mampu menangani trafik *network* untuk beribu-ribu *user* secara simultan.

Selain itu, translasi alamat yang diterapkan dalam NAT, membuat para *cracker* di Internet tidak mungkin menyerang langsung sistem-sistem di dalam jaringan internal. *Intruder* harus menyerang dan memperoleh akses ke mesin NAT dulu sebelum menyiapkan serangan ke mesin-mesin di jaringan internal. Penting di ketahui bahwa, sementara dengan NAT jaringan internal terproteksi, namun untuk masalah *security*, tetap saja diperlukan paket *filtering* dan metoda pengamanan lainnya dalam mesin NAT.

Contoh Kasus Installasi `Natd`

Sebuah perusahaan kecil memiliki sejumlah komputer dan sambungan ke Internet. Komputer-komputer itu saat ini telah membentuk suatu LAN. Sambungan Internet-nya diasumskan berupa *dedicated T1 link*

Langkah-langkah yang harus dilakukan

1. **Installasi FreeBSD**

Sediakan satu komputer untuk dijadikan *Gateway*. Penulis menyarankan penggunaan **FreeBSD RELEASE 2.2.6** (`Natd` hanya jalan di FreeBSD 2.2.1 ke atas), karena selain gratis juga *requirement hardware*-nya tidak terlalu boros. PC 486 dengan 16 MB *memory* dan HD 850 MB juga sudah cukup mewah.

Untuk mengetahui proses installasi FreeBSD, silahkan baca kembali tulisan-tulisan di Infokomputer sebelumnya dan manual FreeBSD sendiri.

2. **Installasi *Gateway***

Pasang 2 *network interface* agar mesin ini menjadi *gateway*. *Network Card* (misal NE2000 atau 3COM) satu dihubungkan ke jaringan internal dan satu lagi untuk koneksi ke ISP. Misalnya dua-duanya NE2000 *Compatible*. maka *nick* untuk *card* yang menghadap ke dalam adalah `ed0` dan untuk card yang menghadap keluar adalah `ed1.`

Pastikan juga `option gateway = "YES"` tertulis dengan benar dalam *file* `rc.conf`. Atau bisa juga dengan mengetik perintah: `sysctl -w net.inet.ip.forwarding=1`

3. **Installasi *Firewall***

Pasang IP *firewall* di mesin FreeBSD ini. Caranya adalah :

a. Edit *kernel source* di `/usr/src/sys/i386/conf`
Tambahkan *option-option* berikut ini pada file *kernel*.

```
options         IPFIREWALL
options         IPFIREWALL_VERBOSE
options         "IPFIREWALL_VERBOSE_LIMIT=100"
options         IPDIVERT
```

b. Compile *kernel* tersebut
c. Aktifkan *firewall* di `rc.conf` dengan menambahkan

```
firewall="YES"
firewall_type="OPEN"
```

3. Installasi `Natd`

Langkah-langkahnya adalah sbb:

a. *Download source* nya di `ftp://ftp.suutari.iki.fi/pub/natd`
b. *Unzip* dan *untar archive* tersebut dengan perintah
```
gzip -dc natd_1.12.tar.gz | tar -xvf -
```
c. Lakukan *make* dan *make install* di direktori yang dihasilkan. Ketikkan perintah berikut:
```
cd natd_1.12
make
make install
```
d. Edit *startup file* supaya `Natd` berjalan secara otomatis
Buat file `natd.sh` di `/usr/local/etc/rc.d`. Isi file tersebut adalah

```
#!/bin/sh
        /sbin/ipfw -f flush
        /sbin/ipfw add divert 13494 ip from any to any via ed0
        /sbin/ipfw add pass all from 127.0.0.1 to 127.0.0.1
        /sbin/ipfw add pass ip from any to any
        /usr/local/sbin/natd -port 13494  -interface ed0
```

Arti dari file ini adalah:

- Hapuskan semua rule *firewall*
- Tambahkan feature *divert* di *port* 13494 (Anda bisa mengganti ini dengan *port* yang Anda inginkan) untuk mendiversi paket dari dan ke *gateway* lewat *interface* ed0
- Bolehkan semua paket lewat di atas local host
- Bolehkan semua paket IP lewat semua *interface*
- Jalankan `Natd` dengan menjadi *daemon* yang menunggu di *port* 13494 via *interface* ed0.

e. Reboot mesin FreeBSD-nya supaya setting bisa diaktifkan.

4. Konfigurasikan TCP/IP *Client*.

Jadikan nomor IP *card* `ed0` di FreeBSD sebagai *gateway* dari tiap *workstation*, IP tiap-tiap *work station* harus berada dalam *network* yang sama dengan *card* `ed0` yang ada di mesin *gateway*. Misal `ed0` di-beri nomor IP 192.168.1.1 dan `ed1` 167.205.19.5, maka *workstation* diberi nomor IP 192.168.1.2 s/d 192.168.1.14 jika digunakan *mask* 16 atau 255.255.255.240. `ed1` adalah *interface* yang memiliki IP *address* valid

Setelah semuanya langkah-langkah di atas dijalankan dengan baik maka, applikasi Internet di *client* siap dijalankan via NAT.

Untuk kasus lain misalnya sambungan ke Internet-nya menggunakan modem, maka mekanismenya sama saja, tinggal diganti *interface* di *gateway* yang menghadap keluar dengan *interface* modem (`tun0`) dan jalankan program `ppp` untuk men-*dial* ISP-nya. Khusus untuk *dial-out*, `ppp` sebenarnya memiliki mekanisme sendiri untuk kasus ini yaitu dengan `option -alias`. Jadi jika kita menjalankan `ppp` dengan `option -alias` maka kita tidak perlu menjalankan `Natd`, karena option ini menyediakan fasilitas yang sama dengan `Natd` khusus untuk `dial-out`.

`Natd` hanyalah salah satu cara untuk menghemat persediaan IP *address* yang semakin menipis. Dengan adanya fakta bahwa untuk bergabung ke Internet, *host* pencari informasi (*Client*) sebenarnya tidak perlu memiliki IP *address* legal, maka IP *address* legal tersebut bisa dicadangkan untuk *host-host* penyedia informasi (*Server*). Penelitian untuk terus memperbaiki performansi Internet ini masih terus dikembangkan. Sekarang ini juga sedang dikembangkan model IP versi baru yaitu IP versi 6 (IPv6), yang bisa menampung lebih banyak lagi komputer-komputer di Internet. Namun demikian untuk kondisi sekarang, `Natd` masih merupakan solusi ampuh sebelum IPv6 diterapkan.

Referensi

Douba, Salim. *Networking UNIX, The Complete Reference for UNIX networks*. SAMS Publishing. 1995
Unix Integration to WAN: Applied Computer Internetworking. CNRG ITB
FreeBSD Handbook. FreeBSD Inc.