SNATI
Seminar Nasional
Aplikasi Teknologi Informasi 2004
Yogyakarta, 19 Juni 2004

Aplikasi Virtual Driver NDIS (*Network Device Interface Specification*) untuk Pengembangan *Network Analyzer*

Sujoko Sumaryono
Jur. Teknik Elektro, Fak. Teknik, UGM
Jl. Grafika 2 Bulaksumur Yogyakarta, Telp. (0274) 902201, Faks (0274) 902200
e-mail: sujoko@mti.gadjahmada.edu

Abstract

The network analyzer instruments are needed for monitoring, and fault identifications in a local network. An application program in PC based can realize this instrument. The information MAC (Medium Access Control) frames are needed to develops these applications. In Microsoft Windows environment, there are three methods to capture the MAC frames in the local network that are direct NIC (Network Interface Controller) chip access, real NIC driver access, and virtual driver NDIS access. Virtual driver NDIS is used in this research, because the methods are independent from kinds of NIC. The result of the research shows: the application can to identify MAC address, IP address, protocol transport, services, the packet content, and statistically to analyze every packet has passed in the local network. The monitoring report can be store in a file. The application has successfully analyzed some local network technology, a UTP-Ethernet LAN, a wireless LAN, and a dial-up connection.

Keywords: *Network analyzer, MAC (Medium Access Control), NIC (Network Interface Controller), and Virtual driver NDIS*

1. Pendahuluan

1.1 Latar Belakang

Perkembangan infrastruktur teknologi informasi dewasa ini sangat pesat, jaringan komputer adalah salah satu aspek diantaranya. Jaringan yang berskala interprise dibangun berdasar jaringan lokal jaringan lokal yang saling terkoneksi. Dalam membangun dan memelihara suatu jaringan lokal kadang diperlukan alat untuk membantu dalam melakukan pengujian-pengujian, monitoring, serta mencari kegagalan-kegagalan pada sistem jaringan. Bahkan lebih jauh diperlukan untuk menganalisa suatu keadaan yang terjadi pada jaringan.

1.2 Perumusan Masalah

Pada dasarnya suatu aplikasi pengamat jaringan (*Network Analyzer*) memerlukan informasi dari lapisan MAC (*Medium Access Control*). Dalam pengembangan aplikasi untuk jaringan dapat dilaksanakan beberapa metode, yaitu akses langsung ke kontroler jaringan (NIC), akses melalui real driver (Dlink, 3Com, Micronet, Compex, dll), akses melalui *virtual driver* (NDIS) , dan pemrograman socket (WinSock, dan Berkeley Socket).

Pemrograman melalui akses langsung ke kontroler jaringan diperlukan informasi chip yang digunakan (misalnya: realtex, Atmel, Intersil PRISM, dll), dan mengenai aspek aspek antarmukanya dengan komputer (misalnya: I/O address, saluran Interupsi, dan kanal DMA), kelebihannya adalah diperoleh data hingga aras perangkat keras jaringan dan

memanipulasinya. Kelemahannya adalah suatu aplikasi yang dibangun hanya dipersembahkan untuk salah satu dari jenis *chip* yang digunakan.

Aplikasi yang dibangun berdasar akses ke real driver, kelebihannya hampir sama dengan akses langsung ke chip, akan tetapi lebih mudah dalam pemrogramannya. Informasi mengenai spesifikasi real driver diperlukan dalam pemrograman, hal ini tidak mudah diperolehnya. Sisi kelema-hannya juga hanya dapat digunakan untuk kartu kontroler jaringan tertentu.

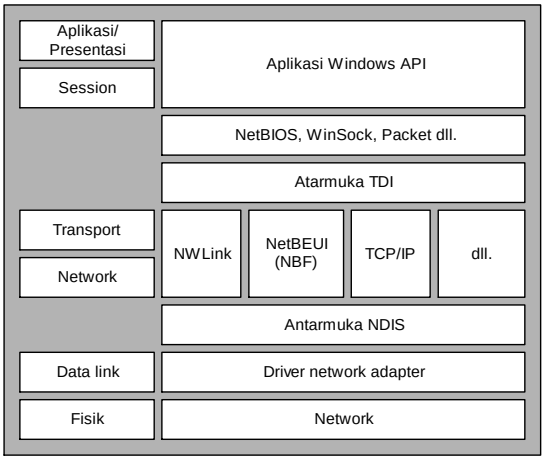
Virtual driver merupakan kelengkapan dari suatu sistem operasi, misalnya NDIS merupakan kelengkapan dari sistem operasi Microsoft Windows. Kelebihan aplikasi yang dibangun berdasar *virtual driver* ini adalah tidak tergantung dari jenis kartu kontroler jaringan yang digunakan. Ditinjau dari sisi kelemahannya, bahwa parameter-parameter jaringan yang dapat diakses tergantung dari fungsi-fungsi yang disediakan oleh virtual driver tersebut.

Ketiga metode tersebut diatas, sesungguhnya telah mencukupi untuk memba-ngun suatu aplikasi penganalisa jaringan, karena informasi mengenai lapisan MAC dapat diperolehnya. Pemrograman jaringan berdasar *Socket*, tidak dapat digunakan karena hanya menghasilkan paket dan memanipulasi pada lapisan IP saja.

Pada penelitian ini, digunakan metode akses melalui *virtual diver* (NDIS*). Source code* program untuk pengembangan aplikasi ini ditulis dalam bahasa C++, yang beberapa pustakanya telah disediakan oleh C++ compiler (Microsoft C++ V. 7.0) untuk akses NDIS.

1.3 Pembatasan Masalah

Pada dasarnya NDIS merupakan virtual driver untuk mendukung beberapa protokol jaringan, di samping TCP/IP, dapat juga digunakan untuk mendukung protokol, Net-BUI, NWLink, dan lainnya dapat bekerja pada lingkungan sistem operasi Microsoft. Pada penelitian ini aplikasi dibatasi hanya untuk menganalisa protokol IP pada jaringan lokal Ethernet.



Gambar 1. Arsitektur jaringan microsoft

1.4 Tujuan Penelitian

Membuat suatu aplikasi untuk menganalisa suatu jaringan Ethernet, sehingga dapat digunakan untuk mempelajari jaringan dengan mengamati mekanisme pertukaran data/frame antar komputer, dan skenarionya untuk masing-masing jenis protokol diatasnya (TCP/IP). Aplikasi dapat juga digunakan sebagai tools dalam uji jaringan, maupun mencari kegagalan dalam instalasi jaringan Ethernet.

2. Landasan Teori

2.1 Arsitektur Jaringan Microsoft

Sistem operasi Microsoft Windows me-ndukung berbagai macam protokol transport jaringan antara lain NetBIOS Frame Proto-col (NBF), Nwlink, TCP, DLC, dan lain-lain. Protokol-protokol tersebut diinte-grasikan dengan menggunakan dua tekno-logi yaitu *Network Driver Interface Spesifi-cation* (NDIS) dan *Transport Driver Interface* (TDI). Arsitektur jaringan Microsoft diperlihatkan Gambar 1.

NDIS dan TDI bertindak sebagai lapisan pemersatu yang memungkinkan komputer yang menjalankan jaringan Microsoft dapat mendukung berbagai kumpulan protokol hanya melalui sebuah antarmuka.

2.2 Transport Driver Interface(TDI)

Protokol ini menetapkan antarmuka protokol, yaitu antara protokol lapisan sesi dengan lapisan transport. Lapisan TDI pada jaringan Microsoft bertindak sebagai antar-muka yang mengintegrasikan bermacam protokol transport. Lapisan di atas TDI digunakan sebagai lapisan antarmuka pemrograman berbagai aplikasi (*Application Program Interface* /API) antara lain NetBIOS dan WinSock API. NetBIOS adalah suatu API yang merupakan produk awal dari jaringan Microsoft. Kebanyakan program yang berbasis jaringan yang dikembangkan dalam lingkungan Microsoft berdasar NetBIOS. WinSock juga suatu API yang standar untuk aplikasi TCP/IP, yang dikembangkan berdasar model socket Berkeley dalam sistem operasi UNIX.

2.3 NDIS

NDIS dikembangkan oleh Microsoft bersama 3Com, yang merupakan antarmuka standar antara protokol lapisan MAC dengan network. Pada lapisan MAC, NDIS menye-diakan fungsi-fungsi antarmuka, yaitu ter-dapat pada file pustaka NDIS.SYS/VXD. Pustaka ini mengekspor semua fungsi-fungsi yang diperlukan untuk pengembangan driver NIC dan menangani semua tugas khusus sistem operasi yang berhubungan dengan semua driver yang berada pada lapisan dibawahnya. NDIS dapat menangani beberapa protokol secara bersama-sama TCP/IP, Nwlink, dan NETBUI.

2.4 Jenis-Jenis Driver yang didukung oleh NDIS

Tiga macam driver yang didukung oleh NDIS dalam sistem operasi yang dike-mbangkan oleh Microsoft (Windows 95/98/NT, 2k, Melenium, XP) yaitu driver NIC, Intermediate Protocol Driver, dan Upper-level Protocol Driver.

2.5 Real Driver NIC

Real Driver NIC berfungsi untuk menga-tur NIC. Driver NIC merupakan antarmuka langsung dengan perangkat keras (*chip controller*) dengan lapisan-lapisan di atasnya untuk melakukan fungsi-fungsi: mengirim dan menerima paket, mereset NIC, menghentikan kerja NIC, mencari NIC dan mengatur karakteristik operasional NIC.

Ada dua macam driver NIC yang oleh NDIS yaitu *Miniport Drivers dan Full NIC Drivers.Miniport Drivers* menetapkan operasi-operasi khusus perangkat keras yang diperlukan untuk mengontrol NIC, termasuk mengirim dan menerima paket. *Miniport* tidak memanggil rutin-rutin sistem operasi secara langsung tetapi memanggil fungsi-fungsi yang diekspor oleh NDIS.

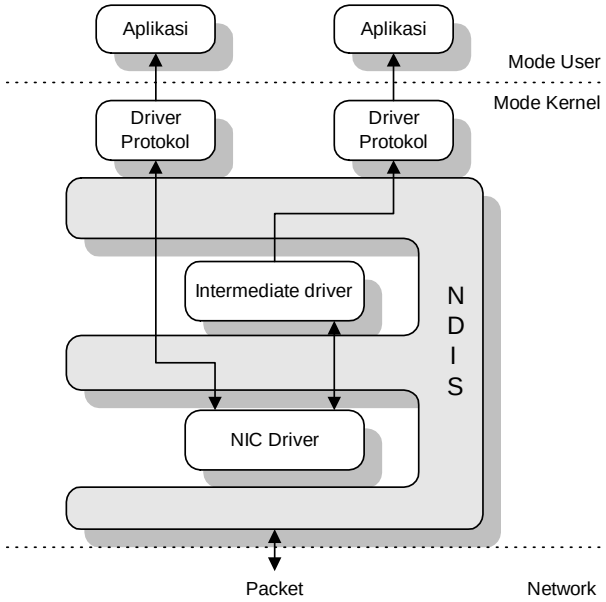
Full NIC Drivers, mengatur tugas-tugas yang bersifat spasifik perangkat keras (*hardware-specific*) yang biasa dikerjakan NDIS dan mengatur tugas spesifik dari sistem operasi. Full NIC Drivers juga memelihara informasi Binding untuk indikasi penerimaan data. Binding digunakan oleh NIC untuk dapat menggunakan beberapa tumpukan (*stack*) protokol sekaligus.

2.6 Intermediate Protocol Driver (IPD)

IPD terletak antara driver protokol dan driver miniport. IPD tampak sebagai *mini-port driver* oleh sebuah protokol transport dan tampak seperti transport protokol oleh *miniport driver*. Alasan utama penggunaan IPD adalah sebagai media penerjemah antara driver transport yang telah ada dengan *miniport driver* yang mengatur media baru yang tidak dikenal jenisnya oleh driver transport. Sebagai contoh IPD dapat mengubah protokol LAN ke protokol ATM (*Asynchronous Transfer Mode*) dalam ATM-LANE (*Local Area Network Emula-tor*). IPD tidak dirancang untuk berkomunikasi dengan aplikasi langsung (*user-mode*), dan hanya dapat berkomunikasi dengan driver NDIS.

2.7 Upper-level Protocol Driver/Transport Driver/Protocol Driver

Upper-level Protocol Driver adalah bagian driver NDIS yang menetapkan suatu atarmuka TDI atau antarmuka khusus untuk suatu aplikasi (application specific) yang digunakan untuk menghasilkan layanan yang diperlukan oleh pengguna. Struktur NDIS dengan dua stack protokol diperlihatkan pada Gambar 2.



Gambar 2. Struktur NDIS dengan dua stack protokol

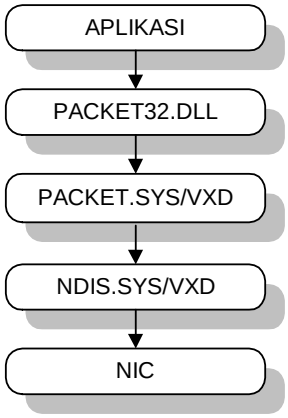
2.8 Packet Virtual Device Driver dan Pustaka Packet32.dll

Komunikasi antar aplikasi dapat dilihat pada Gambar 3. Aplikasi network analyzer dapat berkomunikasi dengan NIC melalui PACKET32.DLL, PACKET.SYS/VXD, dan NDIS.SYS/VXD. Versi NDIS untuk Windows98 adalah NDIS 3.1, Windows NT menggunakan 3.0 – 4.0, dan Windows 2000 menggunakan versi NDIS 5.0.

3. Metode Penelitian

Metode dalam penelitian ini terbagi atas tiga bagian, yaitu: menangkap semua frame yang melewati jaringan, menganalisa, dan *logging* (menyimpan hasil analisa ke suatu file).

Sebuah komputer yang bertindak sebagai penganalisa jaringan (*network analyzer*), dipasang sebuah NIC jenis Ethernet (EIC = *Ethernet Interface Controller*) untuk menangkap setiap frame yang lewat di suatu jaringan. Hal ini dapat dilakukan karena EIC dalam mengirim paket selalu menggunakan mode *broadcastin*g. Setiap frame akan dapat diterima oleh semua komputer yang berada pada jaringan tersebut, dan setiap komputer yang memerimanya akan memeriksa alamat tujuan paket tersebut, jika alamat paket tersebut dialamatkan padanya maka akan menerimanya untuk diproses lebih lanjut. Sebaliknya jika alamat tidak sesuai akan ditolak (dibiarkan). Oleh karena sifat tersebut maka EIC dapat digunakan untuk menangkap setiap paket yang lewat pada jaringan, apakah alamat paket ditujukan padanya maupun tidak.



Gambar 3. Komunikasi antara aplikasi dengan NIC

Mode operasi EIC dapat dikontrol atau ditetapkan melalui driver, yaitu mode *broadcast*, mode *multicast*, mode *direct*, dan mode *promiscuous*.

Pada operasi dengan mode *broadcast*, maka EIC akan mengirimkan suatu paket ke seluruh mesin/komputer yang berada pada jaringan. Alamat broadcast pada ethernet adalah 0xFFFFFFFFFFFF. Operasi *multi-cast,* suatu EIC akan mengirimkan frame ke beberapa mesin yang lain. Sejumlah mesin yang menerimanya dinamakan *group multicast*. Mode *direct* digunakan untuk mengirim frame ke suatu EIC dengan alamat fisik tujuan yang khusus, sesuati dengan alamat yang termuat dalam frame tersebut. Pada operasi *mode promiscuous,* setiap EIC dalam mode ini akan menerima semua frame yang lewat pada jaringan. Pada penelitian ini komputer yang bertindak sebagai network analyzer ditetapkan dalam operasi mode *promiscuous*.

Semua frame yang diterima akan diidentifikasi berdasar frame Ethernet II, yang diambil berdasar standar RFC 1700. Frame Ethernet II terdiri atas : Ethernet header (14 byte), dan diikuti oleh frame protokol IP. Total panjang frame maksimum adalah 1520 byte.

Ethetnet header (14 byte) terdiri atas : alamat mesin (MAC) tujuan (6 byte), alamat mesin asal (6 byte), dan nilai tipe jenis Ethernet (2 byte). Identifikasi frame IP.

Frame IP terdiri atas header IP (20 byte), dan diikuti segmet TCP/UDP.

Header IP (20 byte) terdiri atas: versi IP, *IP header Length, Type of service, Type of service, Total IP length, Identification number, Flag, Fragmentation offset, Time To Live, Protocol, Checksum, Source address, Destination Address, Option,* dan *Padding*.

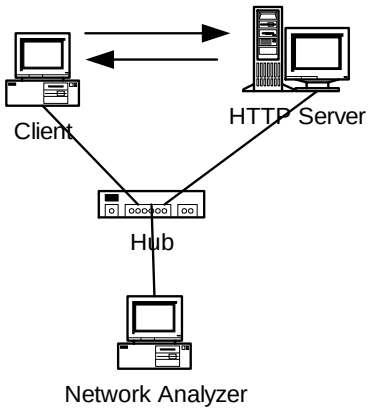
3.1 Perancangan aplikasi network analyzer

Program aplikasi dirancang berdasar Pemrograman Aplikasi Packet32 API. Program terdiri atas:

a. Pembuatan struktur data, untuk struktur frame Ethernet, IP, TCP, UDP, dan protkol lain yang akan diamati.
b. Inisialisasi *WinMain*(), berfungsi sebagai titik masuk dan keluar program dengan pemanggilan fungsi *dlg_Utama*() dan inisialisasi WinSock (fungsi *WSAStartUp*()) sebagai awal operasi WinSock.
c. Menetapkan memori sebagai buffer aplikasi untuk melewatkan data dari paket yang diterima 1520 byte.
d. Mengambil informasi dari komputer host, tentang adapter yang terinstal, dengan fungsi *PacketGetAdapterNames*(), perintah ini mengambil informasi dari file registry yang berisi tentang adapter yang terpasang, yaitu pada kunci: *HKEY_LOCAL_MACHINE\\System\\CurrentControlSet\\Services\\Class\\Net\\0000\\DriverDesc*. Kunci tersebut menyimpan diskripsi driver adapter yang terpasang.
e. Membuka adapter yang telah dipilih dengan fungsi *PacketOpenAdapter*(). Fungsi ini berdasar urutan adapter yang terinstalasi sebagai parameter untuk membuka adapter (*HKEY_LOCAL_MACHINE\\System\\CurrentControlSet\\Services\\Class\\Net\\0000*).
f. Mengambil informasi tentang alamat IP dan Subnet Mask adapter yang telah dibuka dengan mengambil informasi dari registry pada kunci *HKEY_LOCAL_MACHINE\\System-\\CurrentControlSet\\Services\\Class\\Net\\Trans*. Pengambilan informasi dengan mengguna-kan parameter yang sama dengan pembukaan adapter. Alamat IP dan Subnet Mask diubah dari bentuk string ke biner dengan menggunakan fungsi WinSock, yaitu *inet_addr*().
g. Menetapkan netid dari adapter tersebut dengan menggunakan operasi AND bit per bit pada alamat IP dengan Subnet Mask.
h. Mengambil alamat Medium Access Control dari adapter dengan fungsi *PacketGetAddress*().
i. Pemilihan filter yang digunakan untuk menangkap paket dengan fungsi *PacketSetFilter*(). Aplikasi ini menggunakan filter Promiscuous untuk menangkap setiap filter yang lewat dalam jaringan.
j. Inisialisasi buffer untuk packet dengan fungsi *PacketInitPacket*() sebesar memori yang telah dialokasikan diatas.
k. Menangkap setiap packet yang lewat dalam jaringan dengan fungsi *PacketReceivePacket*(). Penangkapan ini menggunakan suatu proses tersendiri, yaitu dengan fungsi *multithreading* atau fungsi *Thread*(). Fungsi *multithreading* digunakan agar proses penangkapan berlang-sung terus menerus.
l. Mengekstraks paket dengan fungsi-fungsi khusus yang telah dibuat yaitu fungsi *CheckEtherType*() digunakan untuk memeriksa protokol yang digunakan dalam paket tersebut dengan memeriksa protokol yang digunakan dalam paket tersebut dengan memeriksa nilai Ethertype pada header Ethernet. Fungsi *ParseIP*() akan dipanggil jika nilai pada Ethertype pada paket tersebut menunjukkan bahwa paket menggunakan protokol IP. Fungsi *ParseARP*() akan dipanggil jika nilai pada Ethertype pada paket tersebut menunjukkan bahwa paket tersebut menggunakan protokol ARP.
m. Pendeteksian host lokal yang sedang aktif akan melakukan operasi AND antara alamat IP yang tertangkap dengan Subnet Maks. Jika alamat-IP yang tertangkap dalam suatu jaringan maka akan ditampilkan pada ListBox.
n. Menghitung kecepatan lalulintas data per detik untuk masing-masing jenis protokol dengan menggunakan fungsi *CounterProc*().
o. Proses komunikasi dapat disimpan dalam suatu file (*aplikasi.log*)
p. Program ini diakhiri dengan membe-baskan paket dengan fungsi *PacketFreePacket*(), kemudian menutup adapter yang telah dibuka dengan fungsi *PacketCloseAdapter*().

3.2 Pengujian

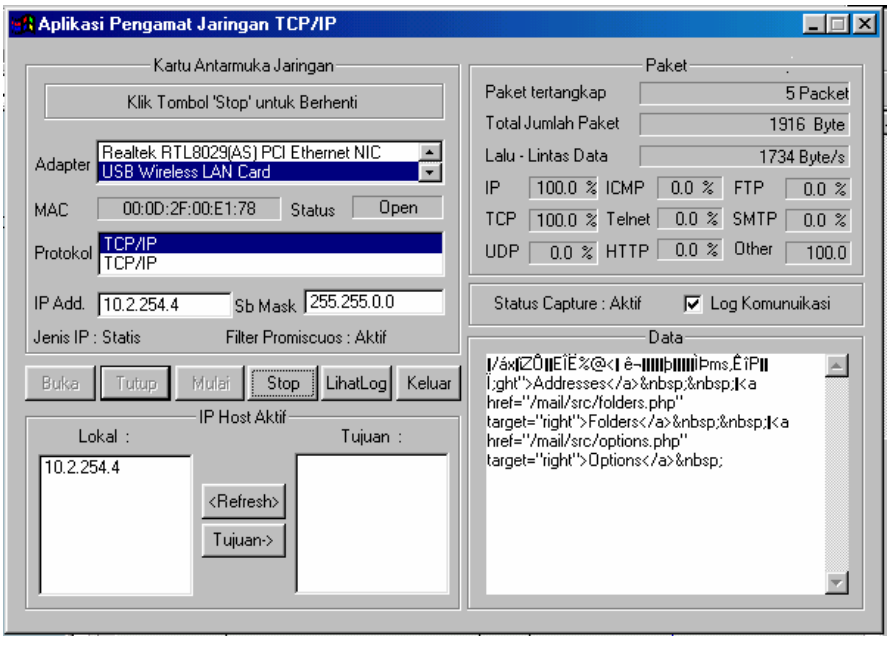
Pengujian dilakukan pada suatu jaringan lokal Ethernet, yang terdiri atas server, client, dan komputer yang menjalankan aplikasi *Network Analyzer*, konfigurasi seperti diperlihatkan pada Gambar 4.



Gambar 4. Konfigurasi jaringan Ethernet dengan sebuah hub

Tampilan aplikasi Network Analyzer diperlihatkan pada Gambar 5. Tampilan terdiri atas spesifikasi kartu antarmuka jaringan yang terpasang, jendela yang menampilkan alapat IP asal dan tujuan dari host-host yang aktif pada jaringan, statistik hasil tangkapan, dan isi data dari seluruh frame yang tertangkap.

Gambar 6 memperlihatkan rekaman hasil analisa yang tersimpan dalam file logger (aplikasi.log), dapat dilihat dengan text editor (ASCII): *Note pad*, *Text pad*, dll..



Gambar 5. Tampilan aplikasi Network Analyzer

```
Packet ke-1  , Jumlah packet 60
MAC Source       : 00:07:EC:5A:D4:0A (Hex)
MAC Destination  : 00:0D:2F:00:E1:78 (Hex)
HWType        : 0.1
Protokol      : 8.0
HWLength      : 6
ProtoAdd      : 4
OpCode      : 0.2
 Source      : 00:07:EC:5A:D4:0A
(ARP Packet) IP Source      : 10.2.1.254
 Destination      : 00:0D:2F:00:E1:78
(ARP Packet) IP Destination : 10.2.254.4

 Packet ke-2  , Jumlah packet 62
MAC Source        : 00:07:EC:5A:D4:0A (Hex)
MAC Destination   : 00:0D:2F:00:E1:78 (Hex)
IP Source      : 172.16.30.3
IP Destination : 10.2.254.4
Protokol Host ke Host : TCP(6)
TCP Seq : 3437127614
TCP Ack : 2935001
TCP FLAG : 18(ACK PUSH)
TCP Window : 5840
Port Service : OTHER
Port Service : OTHER
…
```

Gambar 6. Isi file rekaman (aplikasi.log)

4. Hasil dan Pembahasan

Aplikasi Network Analyzer yang untuk jaringan TCP/IP dapat digunakan untuk melakukan pengamatan:

a. Jumlah total paket yang diterima
b. Alamat fisik (MAC) dan mendes-kripsikan adapter EIC yang terpasang pada komputer.
c. Pengamatan terhadap paket ARP yang merupakan paket protokol yang sangat diperlukan dalam komunikasi dengan protokol TCP/IP
d. Menampilkan detil proses komunikasi yang terjadi.

5. Kesimpulan

Berdasar hasil pembahasan dapat disim-pulkan sebagai berikut:

a. Virtual driver NDIS berjalan dalam sistem operasi MS Windows, sebagai antarmuka antara berbagai jenis adapter jaringan dengan berbagai protokol jaringan (*protocol suite*)
b. Packet32 Virtual Device Driver, secara khusus digunakan untuk menangkap setiap paket MAC yang dikirimkan oleh suatu Ethernet Interface Card, dan tidak tergantung pada jenis chip yang digunakan.
c. Dalam frame MAC yang ditangkap oleh NDIS ini merupakan frame yang memuat MAC header dan frame-frame protokol jaringan yang digunakan, sehingga semua parameter fisik dan jaringan dapat dianalisa.
d. Aplikasi ini dapat digunakan untuk menganalisa berberapa teknologi jaringan lokal, yaitu Ethernet-UTP, WLAN (terbatas pada suatu host), dan koneksi dial-up.

Daftar Pustaka

Apparma, R., Premkumar, B., (1999), *Monitoring Ethernet Network Activity With NDIS Driver,* California Software Laboratories.

Dumas, A., 1995, *Programming WinSock*, Sams Publishing.

Petzold, C., (1996), *Programming Windows95*, Microsoft Press.

Quinn, B., Shunte, D., 1995, *Windows Socket Network Programming*, Prentice Hall, Inc.

---- , (1998-1999), *NDIS Specification*, Microsoft Corporation.