MATRIX TRANSFORMATIONS OF GENERALIZED ENTIRE DIRICHLET SERIES

Trinh Dao Chien
Gia Lai Education and Training Department

Abstract. *We consider matrix transformation of the class of generalized enti Dirichlet series with complex frequencies that define entire functions in* $\mathbf{C}$.

1. Introduction

Matrix transformations of power series of one complex variable has been studied [1]. Recently, Lê Hai Khôi [2] considered matrix transformation of the class of multip Dirichlet series with complex frequencies that define entire functions in $\mathbf{C}^n$. Note that tl techniques used in [1] are essentially one-dimensional, of power series and, therefore, (not work for Dirichlet series.

In this article, based on the ideas in [2], we study matrix transformation of the cla of generalized entire Dirichlet series of the form

$$\sum_{k=1}^{\infty} c_k E_\rho(\lambda_k z), \quad z \in \mathbf{C}, \tag{1.}$$

where $\rho > 0$, $\lambda_k \in \mathbf{C}$, $0 < |\lambda_k| \uparrow +\infty$, $c_k \in \mathbf{C}$ and $E_\rho(z)$ is the Mittag - Leffler function

$$E_\rho(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(\frac{n}{\rho}+1)} \quad \Big(\Gamma \text{ being the Gamma function}\Big).$$

2. Matrix transformation of generalized entire Dirichlet series

Given a sequence (λ_k) with $\lambda_k \in \mathbf{C}$ $0 < |\lambda_k| \uparrow +\infty$ and $\rho > 0$. Consider t generalized Dirichlet series (1.1).

In [4] we proved that, if the series (1.1) converges absolutely for all $z \in \mathbf{C}$, then

$$\limsup_{k\to\infty} \frac{\log |c_k|}{|\lambda_k|^\rho} = -\infty \tag{1.}$$

and conversely, if the coefficients of the series (1.1) satisfy condition (1.2) and if

$$\limsup_{k\to\infty} \frac{\log k}{|\lambda_k|^\rho} < +\infty, \tag{1.}$$

then series (1.1) converges absolutely for all $z \in \mathbf{C}$.

In the case (λ_k) satisfies the condition (1.3), we consider the following sequen space

$$\mathcal{A} = \{(c_k);\ c_k \text{ satisfies } (1.2)\} = \{(c_k);\ \limsup_{k\to\infty} |c_k|^{1/|\lambda_k|^\rho} = 0\}.$$

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-T

Denote by $\mathcal{A}^\alpha$ the Köthe dual space of $\mathcal{A}$, i.e.,

$$\mathcal{A}^\alpha = \Big\{(u_k);\ \sum_{n=1}^{\infty} c_k u_k \ \text{ converges absolutely for all } \ (c_k) \in \mathcal{A}\Big\}.$$

Also we introduce the following set

$$\mathcal{A}^\beta = \Big\{(u_k);\ \sum_{k=1}^{\infty} c_k u_k \ \text{ converges for all } \ (c_k) \in \mathcal{A}\Big\}.$$

In [4], we presented a characterization of the Köthe dual for the space $\mathcal{A}$.

Proposition 2.1. *If $(u_k) \in \mathcal{A}^\beta$, then*

$$\limsup_{k\to\infty} |u_k|^{1/|\lambda_k|^\rho} < +\infty\,. \tag{1.4}$$

Conversely, if the sequence (u_k) satisfies condition (1.4) and, in addition the sequence (λ_k) satisfies condition (1.3), then $(u_k) \in \mathcal{A}^\alpha$.

Corollary 2.1. *If (1.3) holds, then $(u_k) \in \mathcal{A}^\beta$ if and only if $(u_k) \in \mathcal{A}^\alpha$, i.e., $\mathcal{A}^\alpha = \mathcal{A}^\beta$. In this case these sequence spaces can be defined as follows*

$$\mathcal{A}^\beta = \mathcal{A}^\alpha = \Big\{(u_k);\ u_k \ \textit{satisfies condition (1.4)}\Big\}.$$

Throughout this note, we assume that the condition (1.3) holds.

We now proceed to the main result of our note on matrix transformations of generalized entire Dirichlet series.

Denote by $\mathcal{A}_u$ the class of all matrices $(u_{jk})_{j,k=1}^{\infty}$ having the property that whenever the sequence $c = (c_k) \in \mathcal{A}$ the sequence of functions $\big(f_j(z)\big)_{j=1}^{\infty}$ given by

$$f_j(z) := \sum_{k=1}^{\infty} u_{jk} c_k E_\rho(\lambda_k z),\ \ j = 1, 2, ...,$$

converges uniformly on every compact subset of $\mathbf{C}$, each generalized Dirichlet series

$$\sum_{k=1}^{\infty} u_{jk} c_k E_\rho(\lambda_k z),\ \ j = 1, 2, ..., \tag{1.5}$$

being convergent in $\mathbf{C}$.

We have the following result.

Theorem 2.1. *A matrix $U = (u_{jk})$ belongs to the class $\mathcal{A}_u$ if and only if the following conditions hold*

$$\exists \lim_{j\to\infty} u_{jk} =: u_k,\ \ k = 1, 2, ...,$$

and

$$\sup_{j\geq 1, k\geq 1} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty\,.$$

Proof. Necessity. We assume that $U = (u_{jk}) \in \mathcal{A}_u$, i.e., for each $c = (c_k) \in \mathcal{A}$ the sequenc of function $\big(f_j(z)\big)$ converges uniformly on each closed disk $D_R = \{z \in \mathbf{C}; |z| \le R < +\infty\}$ each series (1.5) are convergent in $\mathbf{C}$.

In particular, the sequence

$$\Big(f_j(0)\Big)_{j=1}^{\infty} = \Big(\sum_{k=1}^{\infty} u_{jk}c_k\Big)_{j=1}^{\infty}$$

converges.

Choose (c_k) are unit vectors. It follows

$$\lim_{j\to\infty} u_{jk} =: u_k, \ \ k = 1, 2, ..., .$$

Now, we prove

$$\sup_{j\ge 1,\, k\ge 1} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty\,.$$

Indeed, since the series (1.5) converges in $\mathbf{C}$, by Theorem 2.1 in [4], we have

$$\limsup_{k\to\infty} |u_{jk}c_k|^{1/|\lambda_k|^\rho} = 0, \ \ j = 1, 2, \cdots .$$

On the other hand, since the series (1.5) converges for $z = 0$, we have the series

$$\sum_{k=1}^{\infty} u_{jk}c_k, \ \ j = 1, 2, ...,$$

also converges for $(c_k) \in \mathcal{A}$. Hence $(u_{jk})_{k=1}^{\infty} \in \mathcal{A}^\beta, \ \ j = 1, 2, \cdots$.

It follows from (1.4) that

$$\limsup_{k\to\infty} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty, \ \ j = 1, 2, \cdots .$$

Hence

$$M_j := \sup_{k\ge 1} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty, \ \ j = 1, 2, \cdots .$$

By the same method as was in [1], we can prove that

$$\sup_{j\ge 1,\, k\ge 1} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty\,.$$

Sufficiency. Note that, if (λ_k) satisfy condition (1.2), then there exists $\alpha > 0$ suc that

$$\sum_{k=1}^{\infty} \exp\ \big(-\alpha|\lambda_k|^\rho\big) < +\infty. \qquad (1.6$$

Let $c = (c_k) \in \mathcal{A}$, i.e.,

$$\limsup_{k\to\infty} |c_k|^{1/|\lambda_k|^\rho} = 0\,.$$

Since

$$\sup_{j\geq 1,\, k\geq 1} |u_{jk}|^{1/|\lambda_k|^\rho} < +\infty,$$

we have

$$\limsup_{k\to\infty} |u_{jk}c_k|^{1/|\lambda_k|^\rho} = 0, \quad j = 1, 2, \cdots. \tag{1.7}$$

By Theorem 2.1 in [4] of follows the series $\sum_{k=1}^{\infty} u_{jk}c_kE_\rho(\lambda_k z)$, $j = 1, 2, \ldots$, converges absolutely for all $z \in \mathbf{C}$.

Let $R > 0$. We prove that, for any $\varepsilon > 0$ there exists $j_0 = j_0(\varepsilon)$ such that for all $p \geq j_0$, $q \geq j_0$ we have

$$\sup_{z\in D_R} |f_p(z) - f(f_q(z)| < \varepsilon\,.$$

Indeed, since $(c_k) \in \mathcal{A}$ there exists $k_0 = k_0(\varepsilon)$ such that for all $k \geq k_0$ we have

$$|c_k| < \varepsilon^{|\lambda_k|^\rho}\,. \tag{1.8}$$

Since the entire function $E_\rho(z)$ is order $\rho > 0$ and type $\sigma = 1$, for $\varepsilon_0 \in (0, 1)$ there exists $C = C(\rho, \varepsilon_0) > 0$ and $k_1 \in \mathbf{N}$ such that for all $k \geq k_1$ (see, e.g, [3]) we have

$$\sup_{z\in D_R} |E_\rho(\lambda_k z) < Ce^{(1+\varepsilon_0)|\lambda_k|^\rho|z|^\rho} < Ce^{2|\lambda_k|^\rho R^\rho}\,. \tag{1.9}$$

Moreover, by (1.6), there exists $k_2 = k_2(\varepsilon)$ such that for all $k \geq k_2$ we have

$$\sum_{k=k_2+1}^{\infty} e^{-\alpha|\lambda_k|^\rho} < \frac{\varepsilon}{4C}\,. \tag{1.10}$$

Denote $M = \sup_{j\geq 1,\, k\geq 1} |u_{jk}|^{1/\lambda_k|^\rho} < +\infty$. Then, for all $P \in \mathbf{N}$, $q \in \mathbf{N}$, we have

$$|u_{pk} - u_{qk}| \leq |u_{pk}| + |u_{qk}| \leq 2M^{|\lambda_k|^\rho}, \text{ for all } k \geq 1\,. \tag{1.11}$$

Hence, by (1.8), (1.9), (1.10), (1.11), for $N = \max\{k_0, k_1, k_2\}$ we have

$$\sum_{k=N+1} |u_{pk} - u_{qk}|\,|c_k| \sup_{z\in D_R} |E_\rho(\lambda_k z)| <$$

$$< 2C \sum_{k=N+1}^{\infty} M^{|\lambda_k|^\rho} \varepsilon^{|\lambda_k|^\rho} e^{2|\lambda_k|R^\rho} = 2C \sum_{k=N+1}^{\infty} e^{[\log(M\varepsilon)+2R^\rho]|\lambda_k|^\rho}$$

$$< 2C \sum_{k=N+1}^{\infty} e^{-\alpha|\lambda_k|^\rho} < 2C\frac{\varepsilon}{4C} = \frac{\varepsilon}{2} \tag{1.12}$$

by choosing $\varepsilon < (Me^{\alpha+2R^\rho})^{-1} = \delta$.

On the other hand, since $\lim_{j\to\infty} u_{jk} = U_k$, $k = 1, 2, \ldots, N$, there exists $j_0 = j_0(\varepsilon)$ such that for all $p \geq j_0$, $q \geq j_0$, we have

$$|u_{pk} - u_{qk}| < \frac{\varepsilon}{2Q}, \text{ for all } k = 1, 2, \ldots, N\,,$$

where $Q = \sum\limits_{k=1}^{N} |c_k| \sup\limits_{z\in D_R} |E_\rho(\lambda_k z)|$.

Hence, for all $p \geq j_0$, $q \geq j_0$, we have

$$\sum_{k=1}^{N} |u_{pk} - u_{qk}|\,|c_k| \sup_{z\in D_R} |E_\rho(\lambda_k z)| < \frac{\varepsilon}{2Q}.Q = \frac{\varepsilon}{2}. \quad (1.1$$

By (1.12), (1.13), for any $\varepsilon > 0$ (satisfying $\varepsilon < \delta$) there exists $j_0 = j_0(\varepsilon)$ such th for all $p \geq j_0$, $q \geq j_0$, we have

$$\sup_{z\in D_R} |f_p(z) - f_q(z)| \leq \sum_{k=1}^{N} |u_{pk} - u_{qk}|\,|c_k| \sup_{z\in D_R} |E_\rho(\lambda_k z)|$$

$$+ \sum_{k=N+1}^{\infty} |u_{pk} - u_{qk}|\,|c_k| \sup_{z\in D_R} |E_\rho(\lambda_k z)| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

This implies that the sequence of function $(f_j(z))$ is uniformly convergent on co pact subset of **C**.

The theorem is proved.

Acknowledgment: The author would like to express his deep gratitude to Prof. Nguy Van Mau and Dr. Lê Hai Khôi for helpful suggestion in the preparation of this article.

REFERENCES

1. D. Borwein and A. Jakimovski. Matrix transformations of power series, *Pro. Am Math. Soc.* **122** (2) (1994), pp 511-523.
2. Lê Hai Khôi. Matrix transformations of entire Dirichlet series, *Vietnam J. Ma* **24** (1) (1996), pp 109-112.
3. A. F. Leontiev. *Entire function and series of exponential* (Russian), Nauka, Mosco 1983.
4. Trinh Dao Chien. Sequence space of coefficients of generalized entire Dirichlet seri *VNU, Journal of Science*, Nat. Sci., T. XIV, N.1 (1998), pp 8-15.

TẠP CHÍ KHOA HỌC ĐHQGHN, KHTN, t.XVI, n°3 - 2000

PHÉP BIẾN ĐỔI MA TRẬN CỦA CHUỖI DIRICHLET NGUYÊN SUY RỘNG

Trịnh Đào Chiến

Sở Giáo dục và Đào tạo Gia Lai

Trong bài báo này chúng tôi đề cập đến phép biến đổi ma trận của chuỗi Dirichl nguyên suy rộng dạng

$$\sum_{k=1}^{\infty} c_k E_\rho(\lambda_k z)$$

trong đó $E_\rho(z)$ là hàm Mittag - Leggler.

NU JOURNAL OF SCIENCE, Nat. Sci., t.XVI, n^{0}3- 2000

AN IMPLEMENTATION OF ADDRESS RESOLUTION PROTOCOL ALGORITHM OF CLASSICAL IP OVER ATM

Hoang Vinh Dien
Faculty of Mathematics, College of Natural Sciences - VNU

Nguyen Thuc Hai
Faculty of Mathematics, Hanoi University of Technology

Abstract: *This article describes Address Resolution Protocol Algorithm of Classical IP Over ATM (from now on we call ATMARP algorithm) in RFC 1577. We then present an implementation of this algorithm on TCP/IP networks.*

. Introduction

Asynchronous Transfer Mode (from now on we call ATM) is a new and fast de-eloping networking architecture and technology. It is obvious that ATM technology will lay an important role in the development of current networks. ATM delivers important dvantages over existing LAN and WAN technologies, including the promise of scalable andwidths at unprecedented price and performance points and Quality of Service (QoS) uarantees, which facilitate new classes of applications such as multimedia application.

This is the reason why people want to accelerate the ATM deploy and as a result, 'lassical IP Over ATM comes into being.

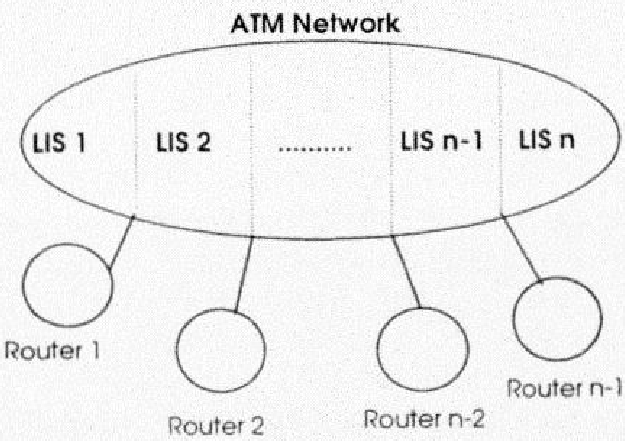


Fig 1: Classical IP Over ATM

Figure 1 shows the configuration of Classical IP Over ATM. As the name indicates, iis model treats the ATM network as a number of separate IP subnets connected through outers. Such an IP subnet is called a logical subnet (LIS). A LIS has the following roperties [1]

i. End systems in a LIS share the same IP prefix and address mask

ii. End systems in a LIS communicates with each other through end-to-end ATM onnections.

iii. End systems in deference LISs communicates with each other through a router.

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX

As in (ii), when an end systems A want to communicates with end systems I it musts to know B's ATM address. Because there is not any way to calculate B's AT address from its IP address, A must ask a Server which have a <ATM address, IP address map entry of B. From now on we call ATMARP Server because its function likes AR Server on traditional networks. This process is illustrated in ATMARP Algorithm.

2. ATMARP Algorithm

2.1 ATMARP Server Operational Requirements

The ATMARP Server accepts ATM calls/connections from other ATM end point At call setup and if the VC supports LLC/SNAP encapsulation, the ATMARP Server wi transmit to the originating ATM station an InATMARP request (InARP-REQUEST) f each logical IP subnet the Server is configured to serve. After receiving an InATMAR reply (InARP-REPLY), the Server will examine the IP address and the ATM address. Tl Server will add (or update) the <ATM address, IP address> map entry and timestam into its ATMARP table. If the InATMARP IP address duplicates a table entry IP addre and the InATMARP ATM address does not match the table entry ATM address and the is an open VC associated with that table entry, the InATMARP information is discard and no modifications to the table are made. ATMARP table entries persist until aged invalidated. VC call tear down does not remove ATMARP table entries.

The ATMARP Server, upon receiving an ATMARP request (ARP-REQUEST will generate the corresponding ATMARP reply (ARP-REPLY) if it has an entry in i ATMARP table. Otherwise it will generate a negative ATMARP reply (ARP-NAK). Tl ARP-NAK response is an extension to the ARMARP protocol and is used to improve tl robustness of the ATMARP Server mechanism. With ARP-NAK, a Client can determi the difference between a catastrophic Server failure and an ATMARP table lookup failur The ARP-NAK packet format is the same as the received ARP-REQUEST packet form with the operation code set to ARP-NAK, i.e., the ARP-REQUEST packet data is merel copied for transmission with the ARP-REQUEST operation code reset to ARP-NAK.

Updating the ATMARP table information timeout, the short form: when the Serv receives an ATMARP request over a VC, where the source IP and ATM address match tl association already in the ATMARP table and the ATM address matches that associat with the VC, the Server may update the timeout on the source ATMARP table entry: i.e if the Client is sending ATMARP requests to the Server over the same VC that it used t register its ATMARP entry, the Server should examine the ATMARP requests and no that the Client is still "alive" by updating the timeout on the Client's ATMARP tab entry.

2.2 ATMARP Client Operational Requirements

The ATMARP Client is responsible for contacting the ATMARP Server to regist its own ATMARP information and to gain and refresh its own ATMARP entry/informati about other IP members. This means, as noted above, that ATMARP Clients MUST l configured with the ATM address of the ATMARP Server. ATMARP Clients MUST:

1. Initiate the VC connection to the ATMARP Server for transmitting and receivi ATMARP and InATMARP packets.

2. Respond to ARP-REQUEST and InARP-REQUEST packets received on ar

VC appropriately.

3. Generate and transmit ARP-REQUEST packets to the ATMARP Server and to process ARP-REPLY and ARP-NAK packets from the Server appropriately. ARP-REPLY packets should be used to build/refresh its own Client ATMARP table entries.

4. Generate and transmit InARP-REQUEST packets as needed and to process InARP-REPLY packets appropriately. InARP-REPLY packets should be used to build/refresh its own Client ATMARP table entries.

5. Provide an ATMARP table aging function to remove its own old Client ATMARP tables entries after a convenient period of time.

Note: if the Client does not maintain an open VC to the Server, the Client MUST refresh its ATMARP information with the Server at least once every 20 minutes. This is done by opening a VC to the Server and exchanging the initial InATMARP packets.

2.3 ATMARP Table Aging

An ATMARP Client or Server MUST have knowledge of any open VCs it has (permanent or switched), their association with an ATMARP table entry, and in particular, which VCs support LLC/SNAP encapsulation.

Client ATMARP table entries are valid for a maximum time of 15 minutes.

Server ATMARP table entries are valid for a minimum time of 20 minutes.

Prior to aging an ATMARP table entry, an ATMARP Server MUST generate an InARP-REQUEST on any open VC associated with that entry. If an InARP-REPLY is received, that table entry is updated and not deleted.

If there is no open VC associated with the table entry, the entry is deleted.

When an ATMARP table entry ages, an ATMARP Client MUST invalidate the table entry. If there is no open VC associated with the invalidated entry, that entry is deleted. In the case of an invalidated entry and an open VC, the ATMARP Client must revalidate the entry prior to transmitting any non address resolution traffic on that VC. In the case of a PVC, the Client validates the entry by transmitting an InARP-REQUEST and updating the entry on receipt of an InARP-REPLY. In the case of an SVC, the Client validates the entry by transmitting an ARP-REQUEST to the ATMARP Server and updating the entry on receipt of an ARP-REPLY. If a VC with an associated invalidated ATMARP table entry is closed, that table entry is removed.

Following is two programs illustrating the above algorithm in TCP/IP network.

3. Implementing ATMARP Algorithm

3.1 ATMARP Server program

Illustration ATMARP Server and ATMARP Client programs are built by Delphi 3.0 of Borland Inc with ServerSocket and ClientSocket Components.

ATMARP Server was configured with unreal ATM address and a IP subnet mask which are served by this Server. These parameters can be modify during Server's operation and are stored on INI file of Windows 95 system. ATMARP Server program uses a timer which count down 2 seconds each times. If there is any map entry that requires to refresh this timer will run a procedure to refresh it.

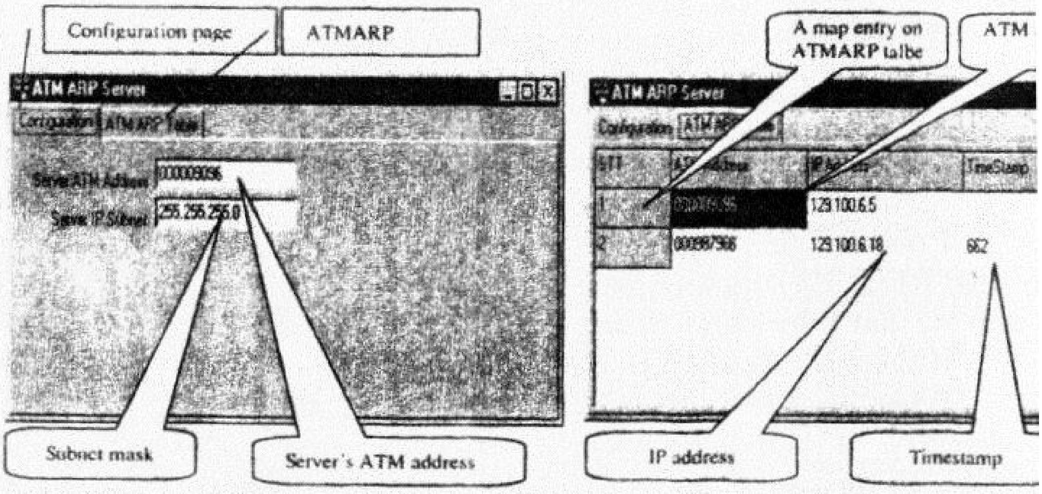
On initializing, the program will listen to port 1998. If there is a connection from network it will add ATM address, IP address and timestamp that are sent by Client.

Timestamp in this case is a variable counting down to zero. If timestamp equals zero t associated map entry will be refreshed. ATMARP Server refresh it map entry by sendi a refresh request to its Client.

If there is a ARP-REQUEST, Server program will examine ATMARRP talbe, it found a map entry it will send that ATM address to Client, otherwise a "Not foun string will be sent.

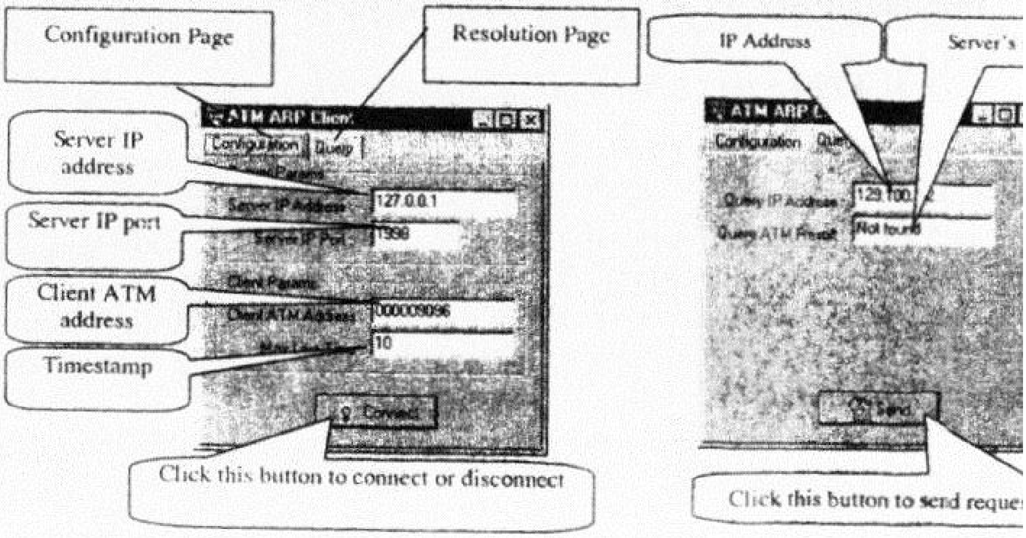
Because full program text is larger than 20 A4 pages, We have to present he a reduce program with the most important codes which are initializing process, Serv socket's event handlers. This is the most important part of ATMARP Server progra Full version of these programs can be found in master thesis "Asynchronous Transfer Mo and Internetworking methods". We use TStringGrid type for ATMARP Table to sho ATMARP table on the screen.

ATMARP Server program's Interface are (refer to appendix for reduced sour code):



3.2 ATMARP Client program

ATMARP Client was configured with unreal ATM address, timestamp and Server port and IP address. These parameters can be modify during Client's operation a are stored on INI file of Windows 95 system. As stated above, We have to present he a reduce program with the most important codes which are initializing process, Clie socket's event handles, "send" and "connect" button process. ATMARP Client program Interface are (refer to appendix for reduced source code) :



4. Conclusion

Although these programs are not full function versions of ATMARP algorithm due to network environment, they show detail operations of ATMARP algorithm. One advantage of this algorithm is the resolution IP ↔ATM address does not require another resolution-IP ↔ MAC ↔ ATM so that processing time, network complexity and network traffics can be reduce. LANE and MPOA haven't made use of this advantage yet.

5. Appendix

5.1 Reduced ATMARP Server source code

```
// This is initializing process of ATMARP Server program
procedure TfmMain.FormCreate(Sender: TObject);
var
  str : String;
  Inifile : TIniFile;
  i : Integer;
begin
  str := Application.Exename;
  Appdir := ExtractFilePath(Str);
  Str := ExtractFilename(Str);
  Appname := copy(str, 1, pos('.', str)-1);
// listen to port 1998
  PortNumber := 1998;
  ServerSocket.Port := PortNumber;
// Start to listen
  ServerSocket.Open;
// Label ATMARP talbe on the stringgrid
  ATMARPTable.cells[0, 0] := 'STT';
  ATMARPTable.cells[1, 0] := 'ATM Address';
  ATMARPTable.cells[2, 0] := 'IP Address';
  ATMARPTable.cells[3, 0] := 'TimeStamp';
// Read Server parameters on INI file
  try
    Inifile : = TIniFile.Create(Appdir + appname + '.Ini');
    ATMAddress := Inifile.ReadString('Configuration',
    'ATMaddress','0000000000');
    IPSubnet := Inifile.ReadString('Configuration', 'IPSubnet',
    '255.255.0.0');
    IPAddress := ServerSocket.Socket.LocalAddress;
    edATMAddress.Text := ATMAddress;
    edIPSubnet.Text := IPSubnet;
// The first map entry is Server's address itself
    ARPEntry := 1;
    ATMARPTable.cells[0, ARPEntry] := IntToStr(ARPEntry);
    ATMARPTable.cells[1, ARPEntry] := ATMAddress;
    ATMARPTable.cells[2, ARPEntry] := IPAddress;
  finally
    IniFile.Free;
  end;
end;
// This is Server socket accepting connection event's event handle from Client
procedure TfmMain.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
```

```
  ATMARPTable.cells[2, 1] := Socket.LocalAddress;
end;
// This is Server socket disconnect event's event handle when Client disconnects
procedure TfmMain.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
var
  I, J : Integer;
begin
// Erase associate map entry of Client on ATMARP table
  IPAddress := Socket.RemoteAddress;
  I := 1;
  while (ATMARPTable.cells[2, I] <> IPAddress)
        and (I <= ATMARPTable.RowCount - 1) do Inc(I);
  if I <= ATMARPTable.RowCount - 1 then
  for J := I to ATMARPTable.Rowcount - 2 do
  begin
    ATMARPTable.cells[0, J] := IntToStr(I);
    ATMARPTable.cells[1, J] := ATMARPTable.cells[1, J + 1];
    ATMARPTable.cells[2, J] := ATMARPTable.cells[2, J + 1];
    ATMARPTable.cells[3, J] := ATMARPTable.cells[3, J + 1];
  end;
  ATMARPTable.Rowcount := ATMARPTable.Rowcount - 1;
end;
// This is Server socket's event handle for messages from Client
procedure TfmMain.ServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  I : Integer;
  IP, Subnet : LongInt;
begin
  Value := Socket.ReceiveText;
// if messages is new Client registration then register Client
  if ReturnStr(#9, Value, 1) = 'New Client Registration' then
  begin
    ATMAddress := ReturnStr(#9, Value, 2);
    IPAddress := ReturnStr(#9, Value, 3);
    Maxlivetime := StrToInt(ReturnStr(#9, Value, 4));
    ATMARPTable.RowCount := ATMARPTable.RowCount + 1;
    ARPEntry := ATMARPTable.Rowcount - 1;
    ATMARPTable.cells[0, ARPEntry] := IntToStr(ARPEntry);
    ATMARPTable.cells[1, ARPEntry] := ATMAddress;
    ATMARPTable.cells[2, ARPEntry] := IPAddress;
    ATMARPTable.cells[3, ARPEntry] := IntToStr(MaxLiveTime)
  end;
// if message is refresh request's reply then refresh map entry
  if ReturnStr(#9, Value, 1) = 'Refresh Reply' then
  begin
    I := 1;
    while (ATMARPTable.cells[2, I] <> IPAddress)
          and (I <= ATMARPTable.RowCount - 1) do Inc(I);
    if I > ATMARPTable.RowCount - 1 then exit;
    ATMAddress := ReturnStr(#9, Value, 2);
    IPAddress := ReturnStr(#9, Value, 3);
    Maxlivetime := StrToInt(ReturnStr(#9, Value, 4));
```

```
    ATMARPTable.cells[1, I] := ATMAddress;
    ATMARPTable.cells[2, I] := IPAddress;
    ATMARPTable.cells[3, I] := IntToStr(MaxLiveTime);
  end;
// if message is ARP query then answer
  if ReturnStr(#9, Value, 1) = 'Client ARP Querry' then
  begin
    IPAddress := ReturnStr(#9, Value, 3);
    if (StrIPtoHexIP(IPAddress) and StrIPtoHexIP(IPSubnet))
       <> (StrIPtoHexIP(ATMARPTable.cells[2, 1])
       and StrIPtoHexIP(IPSubnet)) then
    begin
      Socket.SendText('Server Answer Client ARP Querry :'+ #9 +
                      'Wrong IP Subnet');
      exit;
    end;
    I := 1;
    while (ATMARPTable.cells[2, I] <> IPAddress)
          and (I <= ATMARPTable.RowCount - 1) do Inc(I);
    if I > ATMARPTable.RowCount - 1 then
      Socket.SendText('Server Answer Client ARP Querry :'+ #9
      + 'Not Found')
    else Socket.SendText('Server Answer Client ARP Querry :'+ #9
                         + ATMARPTable.Cells[1, I]);
  end;
end;

// This is timer's event to refresh map entry by sending message to Client
procedure TfmMain.TimerTimer(Sender: TObject);
var
  I : Integer;
begin
  For I := 2 to ATMARPTable.Rowcount - 1 do
  begin
    ATMARPTable.cells[3, ARPEntry] := IntToStr(
                       StrToInt(ATMARPTable.cells[3, ARPEntry]) - 2);
    if StrToInt(ATMARPTable.cells[3, ARPEntry]) <= 0 then
    begin
      if StrToInt(ATMARPTable.cells[3, ARPEntry]) <= - 120 then
         ServerSocket.Socket.Connections[I - 2].Close
      Else
          ServerSocket.Socket.Connections[I - 2].SendText(
         'Refresh Please');
    end;
  end;
end;
```

2 Reduced ATMARP Client source code

```
procedure TfmMain.FormCreate(Sender: TObject);
var
  str     : String;
  Inifile : TIniFile;
  i       : Integer;
begin
```

```
  str := Application.Exename;
  Appdir := ExtractFilePath(Str);
  Str := ExtractFilename(Str);
  Appname := copy(str, 1, pos('.', str)-1);
// Read cleint program's parameters in INI file
  try
    Inifile := TIniFile.Create(Appdir + appname + '.Ini');
    ServerIPAddress := Inifile.ReadString('Configuration',
                          'ServerIPaddress', '129.100.6.5');
    ServerIPPort := Inifile.ReadInteger('Configuration',
                          'ServerIPPort', 1998);
    ClientATMAddress := Inifile.ReadString('Configuration',
                          'ClientATMaddress', '0000000000');
    MaxliveTime := Inifile.ReadInteger('Configuration',
                          'MaxLiveTime', 900);
    edServerIPAddress.Text := ServerIPaddress;
    edServerIPPort.Text := IntToStr(ServerIPPort);
    edClientATMAddress.Text := ClientATMaddress;
    edMaxliveTime.Text := IntToStr(MaxLiveTime);
  finally
    IniFile.Free;
  end;
end;
// This is Client socket's event handle for Server acceptance event
procedure TfmMain.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
Begin
// Register Client IP, ATM address and timestamp...
  Socket.SendText('New Client Registration'+#9 + ClientATMAddress +
                  #9 + ClientSocket.Socket.LocalAddress +
                  #9 + IntToStr(MaxLiveTime));
end;
// This is event handle for" send' buuton
procedure TfmMain.btSendClick(Sender: TObject);
begin
  ClientSocket.Socket.SendText('Client ARP Querry'+ #9 +
        ClientATMAddress + #9 + edQueryIPAddress.Text +
        #9 + IntToStr(MaxLiveTime));
end;
// This is event handle for messages from Server
procedure TfmMain.ClientSocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
// if this messages is ARP_REQUEST reply then process it
  Value := Socket.ReceiveText;
  if ReturnStr(#9, Value, 1) = 'Server Answer Client ARP Querry :'
     then
  begin
    edQueryATMResult.Text := ReturnStr(#9, Value, 2);
  end;
  if Pos('Refresh Please', Value) = 1 then
  begin
// if this message is refresh reply
    Socket.SendText('Refresh Reply'+ #9 + ClientATMAddress +
```

```
            #9 + ClientSocket.Socket.LocalAddress +
            #9 + IntToStr(MaxLiveTime));
  end;
end;
// This is event handle for" connect' button
procedure TfmMain.btConnectClick(Sender: TObject);
begin
  if btConnect.Caption = 'Connect' then
  begin
    ClientSocket.Address := edServerIPAddress.Text;
    ClientSocket.Port := StrToInt(edServerIPPort.Text);
    ClientSocket.Open;
    btConnect.Caption := 'Disconnect';
  end else
  begin
    ClientSocket.Close;
    btConnect.Caption := 'Connect';
  end;
end;
```

REFERENCES

1] Mark Laubach. *"RFC 1577"*, Hewlett-Packard Laboratories, Palo Alto, 1994.

2] David E.McDysan, Darrent L.Spohn. *"ATM Theory and Application"*. McGraw-Hill, 1994.

3] Rainer Händel. Manfred N.Huber, Stefan Schröder. *"ATM networks Concept, Protocol, Application"*. Addison-Wesley Publishing. Munich, 1994.

4] Walter J. Goralski. *"Introduction to ATM Networking"*. McGraw-Hill, 1994.

5] Anthony Alles. *"ATM Internetworking"*, Cisco system, 1995.

AP CHÍ KHOA HỌC ĐHQGHN, KHXH, t.XV, N°3, 2000

MỘT CÀI ĐẶT THUẬT TOÁN PHÂN GIẢI ĐỊA CHỈ CỦA GIAO THỨC KẾT NỐI IP-ATM

Hoàng Vĩnh Diện

College of Natual Sciences - VNU

Nguyễn Thúc Hải

Hanoi University of Technology

Bài báo trình bầy về thuật toán phân giải địa chỉ của giao thức kết nối liên mạng IP-ATM (Classical IP Over ATM) trong RFC 1577. Sau đó đưa ra một cài đặt cụ thể của thuật toán này trên môi trường mạng TCP/IP.