논문 2016-1-5

Riverbed 모델러의 응용 프로세스 개발 절차

기장근*

Development Procedure of Application Process in Riverbed Modeler

Jang-Geun Ki*

요 약

최근 다양한 이동단말의 등장과 통신망 기술의 급속한 발전이 이루어지고 있으며, 이러한 통신망 기술 연구 개발기간 단축 및 경제적 효율성을 도모하기 위해 실제 실험망 구축보다는 컴퓨터 시뮬레이션을 통한 기술 개발 및 검증이 보편화되고 있다. 본 논문에서는 통신망 연구에 널리 사용되고 있는 리버베드 모델러에서 새로운 응용 프로세스 모델을 추가하고 기능을 검증하는 방법에 대해 기술하였다.

Abstract

Technologies for mobile terminals and communication networks has been dramatically developed in a decade. In order to reduce the time and budget needed to develop the new technologies about telecommunication networks and terminals, computer simulation has been widely used as a development and verification tool. In this paper, the procedure is described for developing and verifying the new application process model in the Riverbed Modeler which is popularly used in a research on communication networks.

한글키워드 : 모델러, 응용 프로세스 개발, Riverbed

keywords : modeller, application process, Riverbed

1. 서 론

최근 이동단말 기술의 급속한 발전과 보급 확산이 이루어지고 있고, 통신망 또한 새로운 기술에 대한 연구개발이 활발히 이루어지고 있다.

통신망 관련 기술의 개발은 시험망 구축 및 운영에 따른 시간적, 공간적, 비용적 문제 때문에 최근에는 컴퓨터 시뮬레이션을 이용한 연구가 보다 보편화되고 있다.

본 연구에서는 통신망 연구개발에 많이 사용되고 있는 Riverbed사의 모델러 소프트웨어에서 새로운 응용 프로세스 모델을 개발하는 방안에 대해 기술하였다.

기존의 모델러는 자체적으로 Email, Ftp, Http 등과 사용자가 파라미터를 설정 가능한 Custom 응용 등 몇 가지의 응용 프로세스를 지원하고 있지만 기존과 다른 새로운 응용 프로세스를 개발

* 공주대학교 전기전자제어공학부
(email: kjg@kongju.ac.kr)
접수일자: 2016.5.22. 심사완료: 2016.6.5.
게재확정: 2016.6.19.

하고자 할 때 이를 해결하기 위한 방법이 필요하다. 따라서 본 연구에서는 새로운 응용 프로세스를 개발하고 이를 모델러에 반영하여 사용할 수 있도록 하는 방안에 대해 예제를 통해 기술하고, 시뮬레이션을 통해 기능을 검증하였다.

2. 표준 응용 프로세스 모델 개발

(1) gna_mgr.h 파일에 새로운 응용을 위한 구조체를 정의

표준 응용 프로세스 개발 과정을 기술하기 위해 본 논문에서는 새로운 응용 프로세스 자체를 설계하지 않고 대신 기존의 Email 프로세스를 그대로 Chat라는 이름의 프로세스로 복사하여 새로운 응용 프로세스인 것처럼 모델러에 추가한다고 가정한다. 이와 같이 하여도 새로운 응용을 개발하여 모델러에 추가하는 방법 자체에는 변화가 없다.

새로운 응용 프로세스를 모델러에 추가하기 위해 가장 먼저 gna_mgr.h[1] 파일의 내용에 새롭게 정의할 응용타입에 대한 다음과 같은 항목을 추가한다.

① gna_mgr.h 파일안에 정의된 열거형 구조체 GnaT_ApType에서 마지막 엔트리 GnaT_ApType_Count[2] 바로 직전에 추가하기 원하는 새로운 응용타입 이름(GnaT_ApType_Chat)을 삽입함(그림 1(a) 참조)

② 새로 추가할 응용타입(GnaT_ApType_Chat)의 속성값들을 저장하기 위한 구조체 GnaT_Chat_Desc를 정의(그림 1(b) 참조)

1) models₩std₩include₩gna_mgr.h

2) 마지막 엔트리 GnaT_ApType_Count가 열거형 구조체에서 엔트리의 개수를 의미하는 변수로 사용되기 때문에 이 엔트리 앞에다 새로운 응용타입을 추가해야 함

```
/* GnaT_ApType: Specifies various application types,  */
/* such as Video , Voice, Email and etc.              */
/*                                                    */
/* NOTE: The app types are assigned values starting   */
/* from 0, hence the last entry counts the number of  */
/* app types defined. When adding new application type*/
/* do not assign any specific value to the app type.  */
typedef enum
{
   GnaT_ApType_Phase = 0,
   GnaT_ApType_Email,
   GnaT_ApType_Ftp,
   GnaT_ApType_Http,
   GnaT_ApType_Rlogin,
   GnaT_ApType_Print,
   GnaT_ApType_Video,
   GnaT_ApType_Voice,
   GnaT_ApType_Custom,
   GnaT_ApType_Stream,
   GnaT_ApType_p2p,
   GnaT_ApType_Video_Streaming,

   /* Add new application types here, before the last entry*/
   GnaT_ApType_Chat,

   GnaT_ApType_Count
         /* This values counts the number of application*/
         /* types defined. Hence this be the last entry */
         /* of this enum.                               */
} GnaT_ApType;
```

(a) 열거형 구조체 GnaT_ApType 수정

```
typedef struct
   {
   /* Dist. for time between each group of sent chats.   */
   char*     send_intertime_dist_name_ptr;
   /* Dist. for time between each group of received chats*/
   char*     recv_intertime_dist_name_ptr;
   /* chat size distribution (sent and received).        */
   OmsT_Dist_Handle   chat_size_dist_handle;
   /* Dist. for number of chats sent in one group of chat*/
   OmsT_Dist_Handle   send_group_size_dist_handle;
   /* Dist. for num of chats received in one group of chat*/
   OmsT_Dist_Handle   rcv_group_size_dist_handle;
   char*     symb_server_ptr;    /* Symbolic server name  */
   Boolean   rsvp_status;        /* Flag to enable RSVP   */
   char*     outbound_flow_ptr;  /* Outbound flow profile */
   char*     inbound_flow_ptr;   /* Inbound flow profile  */
   OmsT_Qm_Tos  tos;             /* Type of Service for QoS*/
   char      custom_app_name [256];/* Name of custom app
         started for each request sent by this application */
   inthit_rate;    /* Probability that custom app will be
                              invoked for this request.*/
   } GnaT_Chat_Desc;
```

(b) 새로 추가하는 응용타입(GnaT_ApType_Chat) 속성값을 저장하기 위한 구조체 정의

```
typedef struct
   {
   /* Type of chat (login or received or sent).*/
   GnaT_Application_Type    chat_type;
   /* Server name for chat session             */
   char*                    server_name;
   /* chat size distribution                   */
   OmsT_Dist_Handle         chat_size_dist_handle;
   /* Number of chats in group                 */
   int                      chat_count;
   /* Information about application           */
   GnaT_Nam_Appl*           app_info_ptr;
   /* Type of Service for QoS                  */
   OmsT_Qm_Tos              tos;
   /* apptracking support                      */
   AppTrackT_Profile_Info*  apptrack_profile_info_ptr;
   } GnaT_Cli_Chat_Params_Info;
```

(c) 세션 시작시 Chat 매니저에 의해 Chat 클라이언트에게 전달되는 파라미터 정의

그림 1. gna_mgr.h 헤더파일 구조체 수정 및 정의
Fig. 1. Header file scheme definition

③ 세션 시작시 Chat 매니저에 의해 Chat 클라이언트에게 전달되는 파라미터 GnaT_Cli_Chat_Params_Info 정의(그림 1(c) 참조)

(2) application_config.pr.m 프로세스 모델 안에 새로운 응용의 모델 속성 정의

① 메뉴 Interfaces > Model Attributes 정의

application_config.pr.m[3] 프로세스 모델을 열고 메뉴 Interfaces > Model Attributes 클릭해 나타난 창에서 Application Definitions 속성을 선택하고 Edit Properties 버튼을 클릭하고, 나타난 창에서 Description 속성 선택하고 Edit Properties 버튼을 클릭하고, 나타난 창의 Attribute Name 컬럼의 맨 아래에 새로운 속성 이름(Chat)을 추가(그림 2 참고)하고 Type은 compound로 설정한 후 Edit Properties 버튼 클릭해 나타난 창에서 새롭게 추가하려는 응용의 모델 속성들을 정의해준다.(그림 3 참고)

모델 속성 정의 완료 후 프로세스 모델 에디터 창에서 메뉴 Compile > Compile Code를 클릭해 컴파일 해준다. 컴파일이 성공적으로 끝나면 네트워크 에디터 창에서 그림 4와 같이 Application Config 노드 객체를 마우스 우클릭해 나타난 속성 창에서 새로 추가한 응용이름을 볼 수 있게 된다.

② FB블록내 gna_application_desc_parser() 함수안에 새 응용을 위한 파싱 코드 추가

사용자가 정의한 응용 속성 값(그림 3 참조)들을 읽어들여 시뮬레이션 데이터베이스에 값을 저장하기위해, application_config.pr.m 프로세스 모델안의 FB(Function Block) 블록에 정의된 gna_application_desc_parser() 함수를 수정하고, HB(Header Block)블록에 새로운 응용에 대한 파서 함수를 선언하고 FB 블록에 이에 대한 파서 함수 코드를 추가해야 한다.

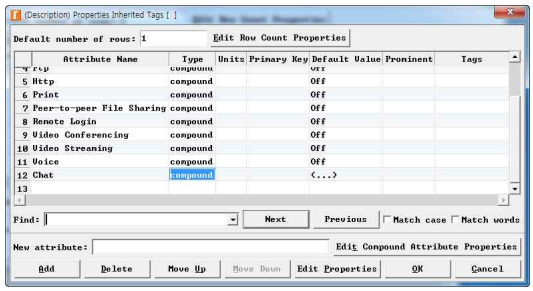


그림 2. Chat 속성이름 추가
Fig. 2. Chat attribute name

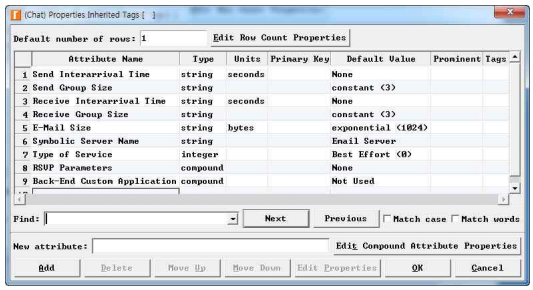


그림 3. Chat 속성 정의
Fig. 3. Chat attribute definition

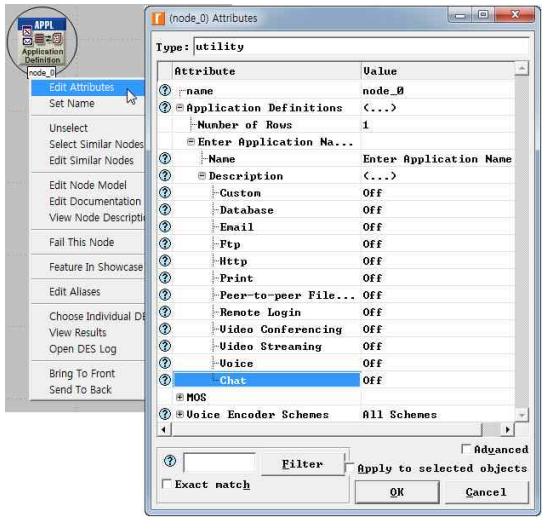


그림 4. Application Config 노드 객체에 새로 추가된 응용이름
Fig. 4. Application Config node name

3) models₩std₩applications₩application_config.pr.m

FB블록에 정의되어 있는 gna_application_desc_parser() 함수는 각각의 응용에 관련된 해당 파싱(parsing) 함수를 호출하도록 코딩되어야 함으로, 새로이 추가될 Chat 응용에 대한 파싱 함수를 호출하는 코드(그림 5 참조)를 gna_application_desc_parser() 함수 안에 추가한다.

실제적인 파싱함수 gna_chat_desc_parser()는 그림 6(a)와 같이 HB블록에 함수선언하고, 그림 6(b)와 같이 FB 블록에 함수내용을 기술한다.

(3) gna_api.h 파일안에 새로운 응용에 대한 이름과 정보를 추가

gna_api.h 파일안에 #define 문장을 이용하여 새로운 응용 이름을 규정하고, 기존에 정의되어 있는 GnaT_Application_Name, GnaT_Application_Type, GnaT_App 구조체에 새로운 응용에 대한 정보를 그림 7과 같이 추가한다.

```
/****** Parse chat application *********/
temp_application_ptr = gna_chat_desc_parser
   (application_row_objid, application_already_parsed);
if (temp_application_ptr != OPC_NIL)
   {
   /* Store the application in the parsing structure. */
   application_ptr->application_desc_ptr =
                                    temp_application_ptr;
   /* Store the type of the application          */
   application_ptr->application_type = GnaT_ApType_Chat;

   /* Set a flag telling that an application has already been
      parsed. Other applications will be ignored. If another
      application is defined a simulation will be written. */
   application_already_parsed = OPC_TRUE;
   }
```

그림 5. 새로운 Chat 응용 파서함수 호출을 위해 gna_application_desc_parser() 함수안에 추가된 코드
Fig. 5. Additive code

```
static void*    gna_chat_desc_parser (Objid, Boolean);
```

그림 6(a). HB블록에 선언된 파싱함수 원형
Fig. 6(a). HB block function

```
static void*
gna_chat_desc_parser (Objid application_row_objid
                      Boolean application_already_parsed)
   {
   Objid              temp_row_objid, temp_comp_objid;
   Objid              comp_cust_app_objid, temp_crow_objid;
   GnaT_Chat_Desc*    chat_desc_ptr;
   char               dist_args [256];
   char               send_interarrival_time [255];
   char               receive_interarrival_time [255];
   char               chat_size [255];
   char               temp_name [255];

   FIN (gna_chat_desc_parser (application_row_objid
                              application_already_parsed))
   /** Parse Chat application. If the chat application   **/
   /** is not defined returns NIL pointer, otherwise    **/
   /** returns the pointer to the parsing structure.    **/

   /* Read attributes     */
   op_ima_obj_attr_get (application_row_objid, "Chat"
                                        &temp_comp_objid);
   temp_row_objid = op_topo_child (temp_comp_objid
                                   OPC_OBJTYPE_GENERIC, 0);
   op_ima_obj_attr_get (temp_row_objid
         "Send Interarrival Time", send_interarrival_time);
   op_ima_obj_attr_get (temp_row_objid
   "Receive Interarrival Time", receive_interarrival_time);
   op_ima_obj_attr_get (temp_row_objid
                                        "Chat Size", chat_size);
   /* Check whether the application is off. If it is, the
      application will not be parsed and OPC_NIL will be
      returned */
   if (((strcmp (send_interarrival_time, "None") !=0) ||
      (strcmp (receive_interarrival_time, "None")!=0)) &&
      (strcmp (chat_size, "None")))
      {
      /* Check whether another application is already defined
         in the list. Only one application should be defined
         at the time. Returns OPC_NIL and write a simulation
         log if more than one application is defined. */
      if (application_already_parsed)
         {
         op_sim_end ("More than one application is defined",
                                                "", "", "");
         }

      /* Allocate memory for parsing this attribute.  */
      chat_desc_ptr = (GnaT_Chat_Desc*) op_prg_mem_alloc
                        (sizeof (GnaT_Chat_Desc));

      /* Filling structure with attributes.  */
      if (strcmp (send_interarrival_time, "None") !=0)
         {
         chat_desc_ptr->send_intertime_dist_name_ptr =
           (char*) op_prg_mem_alloc (sizeof (char) * (strlen
                        (send_interarrival_time) + 1));
         strcpy(chat_desc_ptr->send_intertime_dist_name_ptr
                              send_interarrival_time);
         }
      else
         {  /* None has been specified as the send
               interarrival time. Set a very large value for
               the send interarrival time */
         chat_desc_ptr->send_intertime_dist_name_ptr =
               (char*) op_prg_mem_alloc (sizeof (char) *
               (strlen ("constant (100000000000000)") + 1));
         strcpy(chat_desc_ptr->send_intertime_dist_name_ptr
                           "constant (100000000000000)");
         }

      ..................

      /* Get outbound flow.  */
      op_ima_obj_attr_get (temp_row_objid, "Outbound Flow"
                                              temp_name);
      chat_desc_ptr->outbound_flow_ptr =
         (char *) op_prg_mem_alloc (strlen (temp_name) + 1));
      strcpy (chat_desc_ptr->outbound_flow_ptr, temp_name);
      FRET (chat_desc_ptr);
      }
   else
      {  /* The application is off. Return OPC_NIL.  */
      FRET (OPC_NIL);
      }
   }
```

그림 6(b). FB블록에 기술된 gna_chat_desc_parser() 파싱함수 내용
Fig. 6(b). FB block parsing function

```
#define GNAC_APP_CHAT       "Chat"
.......

/* Enumerated constants for different GNA application types.*/
typedef enum GnaT_Application_Name
    {
    GnaC_App_Custom_Application,


    GnaC_App_Chat
    } GnaT_Application_Name;

/* Enumerated constants for different GNA application types.*/
typedef enum GnaT_Application_Type
    {
    GnaC_App_Type_Custom_Application,


    GnaC_App_Type_Chat_Send
    GnaC_App_Type_Chat_Recv
    } GnaT_Application_Type;

/* GnaT_App: enumerate all apps, base on the port numbers. */
typedef enum
    {
    Ftp = 20


    Chat = 999
    } GnaT_App;
```

그림 7. gna_api.h 파일에 추가된 새로운 응용에 대한 이름과 정보
Fig. 7. gna_api.h file name and information

(4) 응용 프로세스 모델 개발

다음 단계는 새로운 응용에 대한 프로세스 모델을 개발하는 것이다. 본 논문에서는 새로운 응용을 모델러에 추가하는 과정에 초점이 있으므로 새로운 응용 자체를 개발하는 내용은 포함하고 있지 않으며, 대신 기존의 Email 매니저와 클라이언트 프로세스 모델을 Chat 매니저와 클라이언트라는 이름으로 복사하고 이를 수정하여 사용하는 것으로 가정한다. 이 과정에서 생성된 프로세스 모델 이름은 gna_chat_mgr.pr.m과 gna_chat_cli.pr.m 이라고 칭한다.

(5) gna_clsvr_mgr.pr.m 프로세스 모델 수정하여 개발된 응용 프로세스를 모델러 코드와 합침(Integration)

개발된 응용 프로세스를 모델러 코드와 합치기위해(Integration) gna_clsvr_mgr.pr.m[4] 프로세스 모델을 다음과 같이 수정한다.

① 메뉴 Interfaces > Model Attributes 수정

먼저 gna_clsvr_mgr.pr.m 프로세스 모델 에디터 창에서 메뉴 Interfaces > Model Attributes 클릭해 나타난 창에서 Transport Protocol 속성을 선택하고 Edit Properties 버튼 클릭해 나타난 창의 맨 아래쪽에 새로 추가하는 응용의 전송 프로토콜을 그림 8과 같이 정의해 준다.

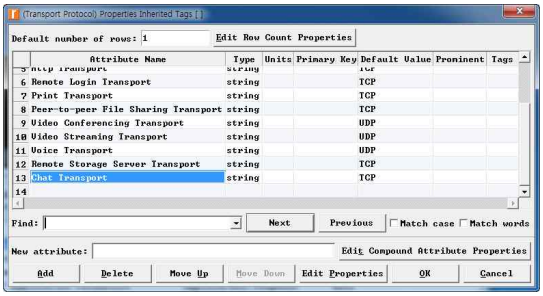


그림 8. gna_clsvr_mgr.pr.m 프로세스 모델의 모델 속성에 새로운 응용의 전송 프로토콜 정의
Fig. 8. gna_clsvr_mgr.pr.m protocol

② HB블럭내 ASC_NUM_APPS 값 증가시킴

다음에 gna_clsvr_mgr 프로세스의 HB 블록에 정의되어 있는 ASC_NUM_APPS 값(서비스 타입의 개수를 나타냄)을 gna_api.h 파일의 GnaT_Application_Type 구조체(그림 7 참조)에 추가해준 엔트리(GnaC_App_Type_Chat_Send와 GnaC_App_Type_Chat_Recv) 개수(2) 만큼 증가시켜준다.(그림 9 참조)

```
/* Maximum number of service types available      */
#define NASC_NUM_APPS        16    // 14->16
```

그림 9. gna_clsvr_mgr 프로세스의 HB 블록에 정의된 ASC_NUM_APPS 값 수정
Fig. 9. HB block value of process

4) C:₩Riverbed₩18.0₩models₩std₩applications₩gna_clsvr_mgr.pr.m

③ SV블록에 상태변수 추가

다음에는 새로운 응용을 위한 최대 통계치 인덱스(Maximum free statistic index)를 위한 상태변수를 gna_clsvr_mgr 프로세스의 SV 블록에 그림 10과 같이 추가해준다.

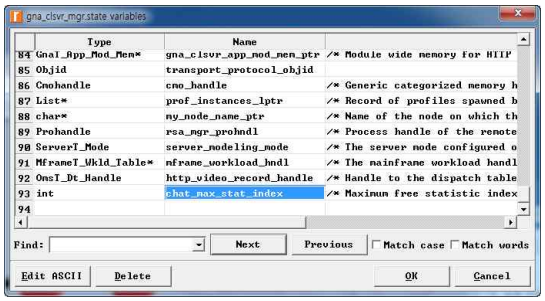


그림 10. gna_clsvr_mgr.pr.m 프로세스의 SV 블록에 상태변수 추가
Fig. 10. SV block state variable

④ FB 블록내 관련 함수들의 내용을 수정

ⓐ gna_clsvr_mgr_service_time_compute() 수정

서버 응답 시간을 계산하는 if문에 새로운 응용의 서버 응답 시간을 계산할 수 있도록 그림 11과 같은 코드를 추가한다.

```
if (app type == GnaC App Type Ftp Get            ||
    app_type == GnaC_App_Type_Email_Recv         ||

    app type == GnaC App Type Chat Recv          ||
    //app_type == GnaC_App_Type_Chat_Send        ||

    (app_type == GnaC App Type Database &&
              type of request == GnaC Query)     ||
    app_type == GnaC_App_Type_Http )
```

그림 11. gna_clsvr_mgr.pr.m 프로세스의 FB 블록내 gna_clsvr_mgr_service_time_compute(...) 함수 수정
Fig. 11. Function revision

ⓑ gna_clsvr_mgr_conf_read() 수정

각 응용에 따른 정보를 읽어 들이는 switch 문을 그림 12와 같이 수정한다.

```
GnaT_Chat_Desc *  chat_appl_ptr = OPC_NIL;
...
   case GnaT_ApType_Chat:
        {
        serv_index = Chat;
        /* Use the compound attribute object id to get     */
        /* the transport protocol for this application     */
        op_ima_obj_attr_get (prtcl_cattr_id, "Chat Transport"
                                                protocol);

        /* Check if the status of the service in this node */
        op_ima_obj_attr_get (comp_attr_objid, "Service Status"
                                                &status);

        /* If the service is disabled in this node, do not */
        /* read any other configuration parameters for this*/
        /* service                                         */
        if (status == OPC_FALSE)
           continue;

        /* Get application description                     */
        chat_appl_ptr = (GnaT_Chat_Desc *)
                  application_desc_ptr->application_desc_ptr;

        /* Store RSVP Parameters                           */
        rsvp_params_ptr = client_nam_rsvp_params_read
                                  (chat_appl_ptr->rsvp_status,
                        chat_appl_ptr->outbound_flow_ptr,
                         chat_appl_ptr->inbound_flow_ptr,
                               GnaC_App_Type_Chat_Send);

        /* Configure stats for this application            */
        gna_clsvr_mgr_stat_initialize ("Chat"
                  application_name_ptr, p_speed, overhead);

        /* Increase stat index                             */
        stat_indexes_table [GnaT_ApType_Chat]++;
        }
        break;
.......
```

그림 12. gna_clsvr_mgr_conf_read() 함수 수정
Fig. 12. Function revision

ⓒ gna_clsvr_mgr_port_to_apptype() 수정

포트 번호에 따른 응용 타입을 얻어내기 위한 switch문을 그림 13과 같이 수정한다.

```
switch (port_type)
       {

       case Chat:
          {
          result = GnaC_App_Type_Chat_Recv;
          break;
          }
       .......
```

그림 13. gna_clsvr_mgr_port_to_apptype() 함수 수정
Fig. 13. Function revision

ⓓ gna_client_profile_parser() 수정

새로운 응용에 대한 정보를 포함하는 프로파일을 파싱하기 위한 switch문을 그림 14와 같이 수정한다.

```
...
   case GnaT_ApType_Chat:
      {
      /* Get the transport protocol   */
      op_ima_obj_attr_get (transport_protocol_objid
                           "Chat Transport", protocol_name_ptr);
      nam_appl_ptr->transport_protocol = (char *)
     prg_cmo_alloc (cmo_handle, strlen (protocol_name_ptr) *
                                          sizeof (char) + 1);
      strcpy (nam_appl_ptr->transport_protocol
                                          protocol_name_ptr);

      /* Get the name of the symbolic server */
      chat_appl_ptr = (GnaT_Chat_Desc *)
                        appl_desc_ptr->application_desc_ptr;
      server_name_ptr = chat_appl_ptr->symb_server_ptr;

      /* Set statistic index for couple <profile/application>*/
      nam_appl_ptr->stat_index = chat_max_stat_index;

      /* Keep the same statistic index if the node is a lan*/
      if (same_lan_profile == OPC_FALSE)
         {
         chat_max_stat_index ++;
         }

      /* Parse RSVP parameters */
      nam_appl_ptr->rsvp_parameters_ptr =
    client_nam_rsvp_params_read (chat_appl_ptr->rsvp_status,
          chat_appl_ptr->outbound_flow_ptr
   chat_appl_ptr->inbound_flow_ptr, GnaC_App_Type_Chat_Send);
      break;
      }
.......
```

그림 14. gna_client_profile_parser() 함수 수정
Fig. 14. Function revision

(6) gna_profile_mgr.pr.m 프로세스 모델의 spawn 상태 enter execs 수정

응용에 맞는 매니저 프로세스를 생성(spawning)하기 위해 그림 15와 같이 switch 문장에 코드를 추가한다.

```
case GnaT_ApType_Chat:
   {
   /* Create many processes if traffic scaling factor is used*/
   /* Insert process handles in list. */

   /* Allocate memory for process handles   */
   application_mgr_prohandle_ptr = (Prohandle*)
          gna_sup_pmo_alloc ("PMO/prof_mgr/app_mgr_prohdl",
            sizeof (Prohandle) * int_traffic_growth_factor *
                                    (1 + extra_instances), 10);

   for (ith_app = 0; ith_app < int_traffic_growth_factor *
                          (1 + extra_instances); ith_app++)
      {
      if (trace_arch_active)
         {
         op_prg_odb_print_minor ("gna_profile_mgr spawning
    process with \"gna_chat_mgr\" process model ", OPC_NIL);
         }

      application_mgr_prohandle_ptr [ith_app] = op_pro_create
                ("gna_chat_mgr", apptrack_profile_info_ptr);
      }

   break;
   }
.......
```

그림 15. gna_profile_mgr.pr.m 프로세스 모델의 spawn 상태 enter execs 수정
Fig. 15. Enter execs of process model function

(7) gna_support.ex.c 파일내 gna_application_name_find() 함수 수정

새로운 응용에 대한 이름 번역을 위해 그림 16과 같이 수정한다.

```
......
      case GnaC_App_Type_Chat_Send:
         {
         GNAC_APP_NAME_SET(appln_name, "Chat Send");
         break;
         }
      case GnaC_App_Type_Chat_Recv:
         {
         GNAC_APP_NAME_SET(appln_name, "Chat Receive");
         break;
         }
.......
```

그림 16. gna_support.ex.c 파일내 gna_application_name_find() 함수 수정
Fig. 16. Function revision

(8) 차일드 프로세스 선언 추가 및 컴파일

그림 17에 새로운 응용 프로세서와 관련된 프로세스들간의 부모 자식간 관계를 나타내었으며, 아래쪽이 위쪽 프로세스의 자식 프로세스이다.

```
gna_clsvr_mgr.pr.m
        ↳ gna_profile_mgr.pr.m
                ↳ gna_chat_mgr.pr.m
                        ↳ gna_chat_cli.pr.m
```

그림 17. 새로운 응용프로세스의 부모-자식 관계
Fig. 17. Parent-child relation

① gna_profile_mgr.pr.m 프로세스 모델을 읽어들여 메뉴에서 File > Declare Child Process Models... 클릭하고 나타난 창에서 새로이 추가된 응응의 매니저 gna_chat_mgr 프로세스 모델을 포함시켜준 다음 Compile > Compile Code 메뉴를 이용해 컴파일 해준다.

② 다시 gna_chat_mgr.pr.m 프로세스 모델을 읽어들여 메뉴에서 File > Declare Child Process Models... 클릭하고 나타난 창에서 새로이 추가된

응응의 클라이언트 gna_chat_cli 프로세스 모델을 포함시켜준 다음 Compile > Compile Code 메뉴를 이용해 컴파일 해준다.

(9) 새로운 응용을 위한 노드모델 생성

새롭게 추가된 응용을 위한 노드모델을 간단히 생성하기 위해 기존의 노드모델을 노드 에디터로 불러들인 후 메뉴 File > Save As...를 이용해 다른 이름으로 저장한 후, 메뉴 Interfaces > Node Statistics를 클릭해 나타난 창에서 관련 통계치들을 추가하되, 클라이언트 노드모델에는 클라이언트 관련 통계치를, 서버 노드모델에는 서버 관련 통계치를 추가한다. 예를 들어 그림 18에는 ethernet_wkstn_adv.nd.m을 복사해 새로 만든 클라이언트 노드모델에 추가된 통계치를 나타내었고, 그림 19에는 ethernet_server_adv.nd.m을 복사해 만든 서버 노드에 추가된 통계치들을 나타내었다.

3. 시뮬레이션을 통한 동작 검증

앞장에 기술한 절차에 따라 새로 개발된 응용 프로세스의 정상적인 동작 검증을 위해 그림 20과 같은 네트워크 모델을 구성하고 시뮬레이션을 수행하였다.

그림 20의 위쪽에 위치한 chat_client와 chat_server는 새롭게 추가된 Chat 응용 기능의 클라이언트와 서버 기능을 수행하는 노드이고, 아래쪽에 위치한 email_client와 email_server는 기존의 Email 응용 기능을 수행하는 클라이언트와 서버 노드이다. 이와 같은 망 구조에서 두 응용의 모든 조건을 동일하게 설정하고 시뮬레이션하여 새로이 추가된 Chat 응용기능이 정상적으로 동작함을 확인하였다.

그림 21에는 다운로드 응답시간과 업로드 응답시간을 나타내었으며, 그림으로부터 Chat 응용과 Email 응용의 결과 값이 거의 같음을 알 수 있고 따라서 Chat 응용이 정상적으로 동작함을 알 수 있다. 또한 본 논문에 그림으로 나타내지는 않았지만 그밖에도 송수신되는 트래픽의 양 등에 대한 다양한 시뮬레이션 결과도 동일함을 확인하였다.

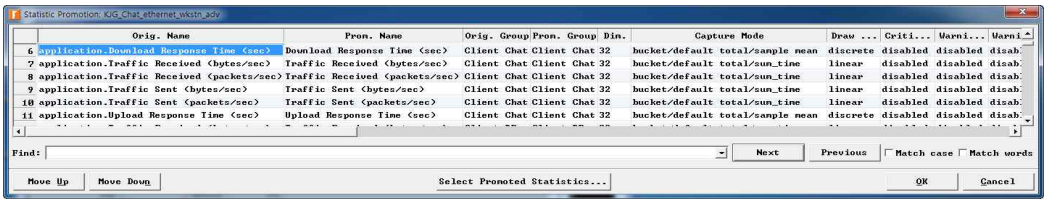


그림 18. 클라이언트 노드모델에 추가된 Node Statistics
Fig. 18. Node Statistics added to client node model

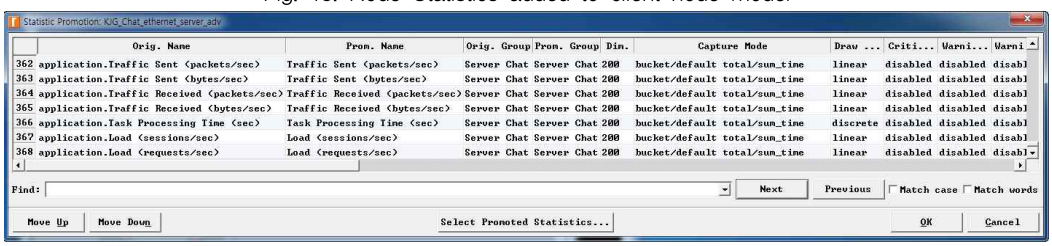


그림 19. 서버 노드모델에 추가된 Node Statistics
Fig. 19. Node Statistics added to server node model

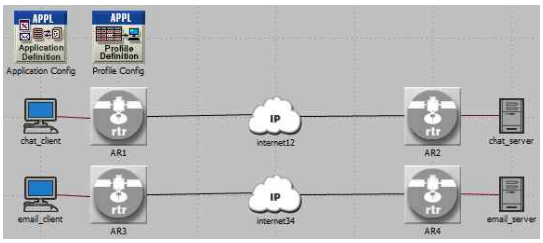
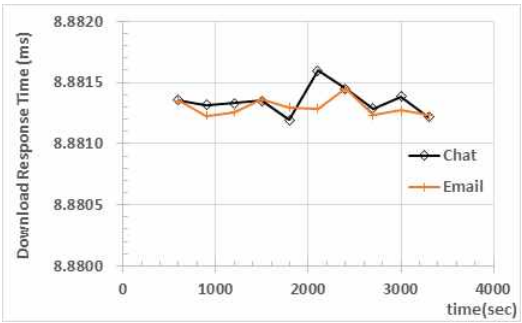


그림 20. 시뮬레이션 망 구조
Fig. 20. Simulation network structure



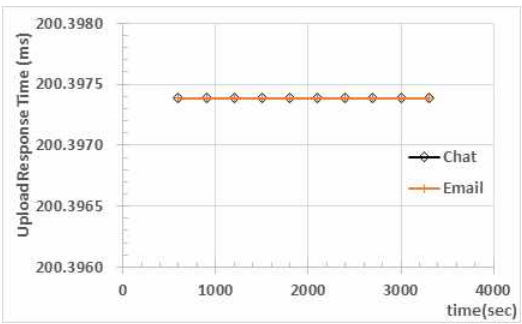


그림 21. 시뮬레이션 결과 비교
Fig. 21. Comparison of results

4. 결 론

급격한 통신기술의 발전과 진화된 이동단말에 대한 사용자 요구 급증으로 인한 여러 가지 요구 조건을 만족시키기 위해 효율적인 통신망 기술 개발 방안이 필요하다. 특히 최근에는 시간적, 공간적, 비용적 측면에서 효율적인 컴퓨터 시뮬레이션을 이용한 통신망 연구가 널리 보편화되고 있다.

본 논문에서는 통신망 연구에 많이 사용되고 있는 Riverbed 모델러 소프트웨어에서 지원하지 않는 새로운 응용 프로세스 모델을 개발하고 모델러 기능에 추가하는 방법에 대해 기술하였으며, 예제 응용 프로세스를 모델러에 추가해 기존 응용 프로세스와 시뮬레이션 결과를 비교함으로써 기능이 정상적으로 동작함을 확인하였다.

참 고 문 헌

[1] V. Hnatyshin, M. Simmons, “Practical Methodology for Development and Deployment of Standard Applications Process Models using OPNET Modeler”, Proceeding of OPNETWORKS 2007, Session 1558 Case Studies: Advanced Simulation Technology, Washington DC, Aug. 2007.

[2] Riverbed(OPNET) Molder, http://www.riverbed.com/kr/products/steelcentral/opnet.html, 2016.

저 자 소 개

기장근(Jang-Geun Ki)

1986.2 고려대학교 전자공학과 졸업
1988.2 고려대학교 전자공학과 석사
1992.2 고려대학교 전자공학과 박사
2002.6-2003.6 Univ. of Arizona 방문교수
2010.6-2011.8 Univ. of Arizona 방문교수
1992.3-현재 : 공주대학교 공과대학 전기전자제어공학부 교수
<주관심분야>통신프로토콜,이동통신시스템