논문 2012-1-4

멀티홈드 노드 인터페이스간 MIPv6 핸드오버 기능 시뮬레이션 모델

기장근*

Simulation Model for MIPv6 Handover between Interfaces in Multihomed Node

Jang-Geun Ki*

요 약

네트워크 연구분야에서 널리 사용되고 있는 시뮬레이션 소프트웨어 OPNET은 멀티홈드 노드 자동 생성기능을 지원하고 있지만 생성된 멀티홈드 노드가 gateway 모드로 동작하도록 되어 있기 때문에 Mobile IPv6의 호스트 동작 모드를 지원하지 못한다. 따라서 본 논문에서는 2개 이상의 인터페이스를 갖는 멀티홈드 이동노드가 MIPv6 관점에서 host로서 동작하면서 인터페이스간 핸드오버 기능을 수행할 수 있는 시뮬레이션 모델 및 프로그램 코드를 개발하였다. 개발된 프로그램은 서로 다른 무선 기술을 사용하는 2개의 인터페이스간 핸드오버를 실험할 수 있는 환경 하에서 시뮬레이션을 통해 정상동작 여부를 검증하였다.

Abstract

OPNET software has been widely used in network related research area. OPNET supports creation of the custom device model such as a multihomed client but the generated multihomed node model works in gateway mode and can not be used as a MIPv6 host. In this paper, we has developed the simulation model and program code to support MIPv6 handover between interfaces of a multihomed mobile node. Function of the developed program has been verified by simulations under various environments to test MIPv6 handover between interfaces based on different wireless technologies.

한글키워드 : 멀티홈드 노드, MIPv6, 시뮬레이션 모델

1. 서 론

무선 통신 기술의 급속한 발달로 2개 이상의 무선 인터페이스를 갖는 멀티홈드(multihomed) 이동 단말이 출현하고 이에 따라 단말기 내의 인터페이스간 핸드오버 기술에 관한 관심이 고조되고 있다.[1] 현재 네트워크 분야에서 널리 사용되고 있는 시뮬레이션 소프트웨어 OPNET[2]은 멀티홈드 노드 모델 자동 생성기능을 지원하고 있지만 생성된 멀티홈드 노드가 gateway 모드로 동작하도록 되어 있기 때문에 Mobile IPv6의 호

* 공주대학교 전기전자제어공학부
(email: kjg@kongju.ac.kr)
접수일자: 2012.4.8 수정완료: 2012.5.30

스트 동작 모드를 지원하지 못한다. 따라서 본 논문에서는 2개 이상의 서로 다른 무선 인터페이스를 갖는 멀티홈드 이동노드가 MIPv6 관점에서 host로서 동작하면서 인터페이스간 핸드오버 기능을 수행할 수 있는 시뮬레이션 모델 및 프로그램 코드를 개발하고, 다양한 환경에서 실험함으로써 개발된 모델과 프로그램을 검증하였다.

2. OPNET을 이용한 시뮬레이션 모델링

네트워크 시뮬레이션 소프트웨어 OPNET[2]을 이용해 멀티홈드 노드가 MIPv6 호스트 기능을 수행하도록 하려면 소스코드를 분석하고 수정하여야 한다. 따라서 본 절에서는 OPNET의 기본적인 기능에 대해 간략히 설명한다.

그림 1에 OPNET 모델링 구조를 나타내었다.

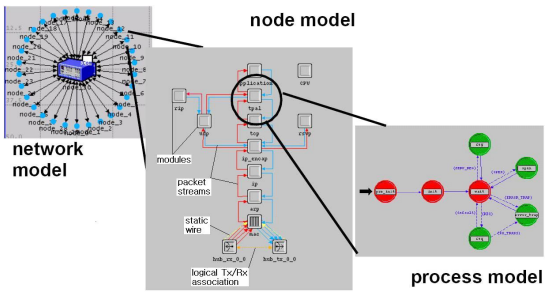


그림 1. OPNET 모델링 구조

OPNET 모델링의 특징은 다음과 같다.

- 네트워크 모델은 복수개의 노드들과 링크들로 구성됨
- 하나의 노드는 복수개의 모듈(module)들로 구성되고, 이 모듈들은 packet stream (또는 statistic wire)들로 연결됨
- 모듈 종류로는 프로세서(processor)와 큐(queue)가 있음
- 각 모듈은 프로세스(process) 모델로 동작을 나타냄
- 프로세서 모듈을 사용하여 패킷의 소스(source)와 싱크(sink)를 표현할 수 있음
- 큐 모듈의 프로세스모델 예 : acb-fifo, acp-fifo
- 프로세스 모델의 forced state는 녹색으로 표시되고, 다음상태로 곧바로 천이함. 반면에 unforced state는 빨간색으로 표시되고 이벤트 발생시까지 해당 상태에 머물러 있음.

OPNET에서 한 노드 내의 모듈간 통신은 패킷 스트림(packet stream) 또는 ICI((Interface Control Information)가 사용되며, 프로세스간 통신은 공유메모리(module memory 또는 parent-to-child shared memory) 또는 상대프로세스 호출함수의 인수(argument)를 이용한다.

2.1 ICI를 이용한 모듈간 통신 방법

먼저 사용하고자 하는 ICI를 설계하고 ICI 에디터를 이용해 모델을 생성한 후 아래와 같은 처리를 해준다.

(1) 소스 프로세스에서 할 일

- Ici* iciptr = op_ici_create(const char* fmt_name)를 이용해 필요한 ICI 생성
- 생성된 ICI에 대해 op_ici_attr_set()을 이용해 필요한 정보를 설정
- 이 ICI를 이벤트/인터럽트와 연계시키기 위해 op_ici_install(Ici* iciptr)을 이용
- 이벤트/인터럽트를 스케줄링

(2) 목적지 프로세스에서 할 일

- 이벤트/인터럽트를 수신한 경우, 연계된 ICI를 op_intrpt_ici()를 이용해 얻어냄
- op_ici_attr_get()을 이용해 필요한 정보를 추출
- op_ici_destroy(Ici* iciptr)를 이용 ICI를 해제

패킷 송신시 op_pk_send_forced() 함수를 사용하는 경우 어떠한 지연/스케줄링 없이 바로 목적지에서 패킷을 수신하여 처리하게 되고, 따라서 이 경우에는 소스 프로세스의 op_pk_send_forced() 함수 밑에 바로 이어 op_ici_install (OPC_NIL)을 호출하도록 프로그램이 작성된다. 즉, 목적지 프로세스에서 ICI를 해제하지 않고, 소스 프로세스가 ICI를 직접 해제한다.

만일 전송되는 패킷에 ICI를 연계시키고 수신측에서 꺼내기 위해서는 op_pk_ici_set(), op_pk_ici_get() 함수를 이용한다.

제공되는 ICI 함수의 간략한 기능은 다음과 같다.

- op_ici_create(), op_ici_destroy() : ICI 속성값 저장위한 메모리 할당 / 해제
- op_ici_attr_set(), op_ici_attr_get() : ICI 메모리에 속성값 설정 / 읽기
- op_ici_install(Ici* iciptr) : ICI를 이벤트/인터럽트에 연계시킴(이후에 발생하는 이벤트/인터럽트들은 이 ICI 값을 전달하게 됨)
- op_ici_install(OPN_NIL) : 이벤트/인터럽트에 연결된 ICI 해제함(이후에 발생하는 이벤트/인터럽트들은 ICI 사용안함)

2.2 프로세스간 통신(정보교환) 방법

(1) module memory 사용

- 모듈 메모리는 하나의 모듈에 속한 모든 부모/자식 프로세스들이 공유하는 메모리임
- 호출 프로세스가 공유메모리 내용을 변경한 후 op_pro_invoke()를 이용해 제어권을 다른 프로세스에게 넘겨주면, 제어권을 넘겨받은 프로세스가 이 공유메모리에 접근하여 변경정보를 이용. 또 다른 방법은 다른 프로세스들이 자발적으로 이 공유메모리를 접근하여 변경 사항이 있는지 확인하는 방법 가능
- op_pro_modmem_install()은 공유메모리 설치시 사용하고 op_pro_modmem_access() 함수는 공유메모리 액세스시 이용
- (예) gna_clsvr_mgr, ip_dispatch 프로세스 모델 참고

(2) parent-to-child shared memory 사용

- 부모와 특정한 하나의 자식 프로세스 사이에 있는 private shard memory임
- op_pro_create() 함수를 호출하여 자식 프로세스를 생성할 때 두 번째 인자(argument)에 부모-자식간 공유메모리 주소를 사용함으로써 설정하고, 자식 프로세스는 op_pro_parmem_access() 함수를 이용해 접근
- 주로 새로운 자식 프로세스를 생성하면서 초기화에 필요한 정보를 넘겨줄 때 사용

(3) argument memory 사용

- 다른 프로세스를 호출할 때 정보가 들어있는 메모리 주소를 인수(argument)로 넘김
- 호출 프로세스는 op_pro_invoke (pro_handle, argmem_ptr) 함수를 이용해 argument memory 주소를 호출된 프로세스에게 넘겨줌.
- 매번 호출시마다 메모리 주소를 새로 넘겨주어야 하며, 메모리가 유지되지 않음에 유의
- 호출된 프로세스는 op_pro_argmem_access() 함수를 이용해 넘겨받은 정보를 접근

2.3 Interrupt Steering

일반적으로 모듈에서 인터럽트(self와 process 인터럽트는 제외)가 발생하면 기본적으로 root process에게 전달되며, root process는 인터럽트

를 조사하여 적절한 child process를 호출하게 된다(통상 root process를 dispatcher 프로세스라고 부름).

특정 종류의 인터럽트에 대해 root가 아닌 다른 지정된 process가 인터럽트를 직접 받도록 하려면 op_intrpt_port_register() (input stream or input statistics 관련 인터럽트) 또는 op_intrpt_type_register() 함수를 이용해 특정 프로세스를 등록해 놓으면 된다(port인터럽트가 type인터럽트보다 우선순위 높음). 예로서, ip 모듈에서 이 방법을 사용하고 있으며, ip_rte_central_cpu 프로세스 모델의 초기화 과정중 ip_rte_central_cpu_inits() 함수내에서 op_intrpt_port_register(...)을 이용해 ip_rte_central_cpu 프로세스 자신이 상위 계층(ip_encap 모듈)으로 부터의 스트림 인터럽트를 받도록 등록한다.

2.4 IP 모듈 프로세스 모델

OPNET의 일반적인 노드 모델에서 IP 계층 기능을 수행하는 모듈의 루트 프로세스 모델인 ip_dispatch.pr.m의 init_too 상태의 Exit executive 안에 있는 함수 ip_dispatch_cleanup_and_create_child_processes()에서, 노드에 MIPv4 관련 속성값이 정의되어 있는 경우 mobile_ip_mgr 프로세스가, MIPv6 관련 속성값이 정의되어 있는 경우 mipv6_mgr 및 ipv6_ra_host/ipv6_ra_gtwy[1] 프로세스가 생성된다. 또한 mipv6_mgr는 모바일노드(host)의 경우 mipv6_mn 프로세스를 생성한다.

참고로 IP 계층 관련 프로세스들간의 정보 교환을 위해 module memory(하나의 모듈에 속한 모든 부모/자식 프로세스들이 공유하는 메모리)를 사용하는데 이 메모리의 데이터 타입은 IpT_Rte_Module_Data 이다. MIPv6 관련 정보들은 그림 2와 같이 이 데이터 구조 변수의 mipv6_info_ptr 필드가 가리키는 곳에 들어 있다.

```
********************** IPv6 & MIP 관련 데이터 구조
typedef struct IpT_Rte_Module_Data
{
          int             first_ipv6_loopback_intf_index;
          Objid               ipv6_params_objid;
                 // 노드가 gateway일경우에만 사용됨
                 // IPv6 attribute object id 가르킴

          Prohandle           ipv6_prohandle;
          Prohandle           ipv6_ra_prohandle;

          struct IpT_Mip_Info*       mip_info_ptr;
                 /* Mobile IP specific information        */
          struct IpT_Mipv6_Info*   mipv6_info_ptr;
                 /* Mobile IPv6 related information.      */

} IpT_Rte_Module_Data;
```

```
<ip_rte_support.h 파일에 정의되어 있음>
/* Data stored for MIPv6 support. */
typedef struct IpT_Mipv6_Info
   {
   Mipv6T_Node_Type    node_type;
         /* MIPv6 node type                  */
   InetT_Address       home_agent_addr;
         /* Home Agent Addres                */
   Boolean             out_of_home;
         /* Flag that indicates when the   */
         /* node is out of its home network   */
   Mipv6T_Bind_Cache_Hash_Table* binding_cache_hashtbl_ptr;
         /* Binding Cache Table (binary hash table) */
   Mipv6T_Bind_Update_List_Hash_Table* binding_update_hashtbl_ptr;
         /* Binding Update List (binary hash table) */
   Prohandle        mipv6_prohandle;
         /* MIPv6 manager process            */
   Prohandle        mipv6_mn_prohandle;
         /* MIPv6 mobile node process      */
   InetT_Address*   care_of_addr_ptr;
         /* Current CoA for the mobile node.*/
   InetT_Address    home_addr;
         /* Home Addres of the mobile node  */
   List*            bind_update_list_lptr;
         /* List of key for the binding update list. */
   List*            bind_cache_table_lptr;
         /* List of keys for the binding cache table */

   /* MIPv6 control traffic stathandlers   */
   Stathandle       local_ctrl_traffic_sent_pkts_shndl;

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

   } IpT_Mipv6_Info;
```

그림 2. 모듈 메모리 IpT_Rte_Module_Data 구조

MIPv6의 경우 mipv6_mgr 프로세스가 처음 실행되면 mipv6_attributes_read() 함수내에서 노드가 host냐 router냐에 따라[2] 해당 MIPv6 값을 읽어들인다.

MN(Mobile Node)의 이동사실 감지 방법은 ipv6_ra_host.pr.m 프로세스 모델이 RA(Router Advertisement) 메시지 수신시 그림 3과 같은 함수 순서로 mipv6_mn 프로세스에게 이동사실을

1) router advertisement 처리 프로세스이며, 노드가 host / gateway(router) 여부에 따라 해당 프로세스 생성됨

2) if (!ip_rte_node_is_gateway (module_data_ptr))

통보하게 되며, mipv6_mn 프로세스는 바인딩 절차를 수행하게 된다.

<ipv6_ra_host.pr.m 프로세스 모델>
ipv6_ra_host_ra_message_handle (void)
→ ipv6_ra_host_router_list_entry_add (ra_info_ptr->router_list,
router_list_entry_ptr);
→ ipv6_ra_host_mobile_ipv6_inform (new_router_list_entry_ptr);[3)]
→ mipv6_notification_event = op_intrpt_schedule_process (iprmd_ptr->
mipv6_info_ptr->mipv6_mn_prohandle, interrupt_time,
IPC_RA_MOBILITY_DETECTION_INTRPT_CODE);

<mipv6_mn.pr.m 프로세스 모델>
mipv6_mn_invocation_process (void)
→ l3_ho_f = mipv6_layer3_handoff_process ();

그림 3. MN의 이동사실 감지 관련 함수 호출

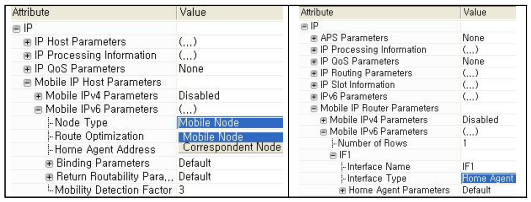
3. 멀티홈드 노드 인터페이스간 MIPv6 핸드오버 시뮬레이션 모델

OPNET[4)]에서 멀티홈드 노드를 생성[5)]하면 기본적으로 host가 아닌 gateway 기능(라우터 기능)이 수행되도록 설정된다. 즉, IP계층 기능을 수행하는 ip_dispatch 프로세스의 모델 속성인 gateway의 값이 enabled로 설정되고, 또한 그림 4(b)에 나타낸 것과 같이 노드의 IP 속성 아래에 Mobile IP Router Parameters 속성을 갖게 되고, Mobile IPv6 Parameters의 경우 Interface Type의 값으로 오직 Home Agent 값만 가질 수 있도록 되어 있다. 반면에 그림 4(a)에 나타낸 것과 같이 일반적인 host 노드의 경우 노드의 IP 속성 아래에 Mobile IP Host Parameters 속성을 가지며 Mobile IPv6의 경우 Node Type 으로 Mobile Node 또는 Correspondent Node를 선택할 수 있도록 되어 있다.

3) ipv6_ra_host_mobile_ipv6_inform (Ipv6T_Ra_Rtr_List_Entry* router_list_entry_ptr)

4) OPNET v.16.1 기준

5) 프로젝트 에디터 창 메뉴 Topology > Create Custom Device Model... 클릭후 나타난 창에서 디바이스 타입으로 Multihomed Host > Client 선택하고 파라미터 칸에서 원하는 인터페이스들의 숫자를 입력



(a) host (b) gateway(router)

그림 4. host와 gageway(router)의 Mobile IPv6 관련 속성 차이

따라서 본 논문에서 개발하고자 하는 멀티홈드 노드의 경우 MIPv6를 지원하는 모바일 노드(호스트)로 동작하여야 함으로 OPNET 소스코드의 수정이 필요[6)]하다.

본 논문에서 수정한 OPNET 소스코드 모델파일의 내역은 아래와 같다.

<ip_dispatch.pr.m>

① 함수 ip_dispatch_ipv6_ra_process_create()에서 멀티홈드 노드의 경우 ipv6_ra_gtwy 프로세스 대신 ipv6_ra_host 프로세스가 생성되도록 수정

② ip_dispatch_icmp_pk_higher_layer_forward(..) 함수에서 멀티홈드 노드가 Router Solicitation 메시지나 Router Advertisement 메시지 수신시 원래의 gateway 노드와 같이 동작하는 것이 아니라 host처럼 동작하도록 코드 수정. 즉 멀티홈드 노드는 Router Solicitation 메시지 수신시 바로 폐기하고, Router Advertisement 메시지 수신시 ipv6_ra_host 프로세스에게 넘겨주도록 프로그램 수정

6) 단순히 ip_dispatch 프로세스의 모델 속성인 gateway의 값을 disabled로 설정해서 host 기능을 수행하도록 설정하게 되면, OPNET 소스코드가 노드 전체에 인터페이스가 오직 1개만 있는 것으로 가정하여 프로그램이 실행되도록 되어 있어 멀티홈드 노드로서의 기능 수행이 불가능함

<ip_rte_support.ex.c>

① 함수 ip_src_address_determine(...)에서 루프백 주소를 사용하는 멀티홈드 노드의 경우 소스주소로 루프백 주소를 리턴하도록 수정

② 함수 ip_rte_datagram_dest_get(...)에서 멀티홈드 노드의 경우 gateway로 설정되어 있지만 MIPv6의 관점에서는 host로 동작하도록 수정 (2개 함수 KJG_ip_rte_multihomed_higher_layer_datagram_dest_get(...)와 KJG_ip_rte_multihomed_lower_layer_datagram_dest_get(...) 추가)

<ipv6_ra.h>

① 구조체 정의 typedef struct Ipv6T_Ra_Rtr_List_Entry{...} 안에 Router Advertisement 메시지 수신 인터페이스 번호 저장을 위한 변수 short KJG_intf_recvd_ index; 추가

<ipv6_ra_host.pr.m>

① host 노드의 경우 함수 ipv6_ra_host_do_init()에서 Router Solicitation/ Advertisement 관련 정보를 얻기 위한 인터페이스로 0번 인터페이스를 무조건 사용하게 되어 있는데, 멀티홈드 노드의 경우 LB0 인터페이스를 사용하도록 수정하고 관련 속성값 읽어들이는 루틴 수정

② host의 IP계층이 ARP모듈로부터 RA(Router Advertisement) 메시지 수신시 수행되는 함수 ipv6_ra_host_ra_message_handle()에서 수신패킷과 같이 전달된 ICI 정보로부터 RA 메시지가 수신된 인터페이스 번호 정보를 추출하여 라우터 리스트에 RA메시지를 송신한 라우터의 주소와 함께 기록해 놓도록 코드 수정

③ 수신된 RA 메시지를 송신했던 라우터의 주소를 라우터 리스트에 추가할 때 수행되는 함수 ipv6_ra_host_router_list_entry_add(...)에서 인터페이스가 하나뿐이라고 가정하고 만들어진 소스코드를 수정하여 RA메시지가 수신된 인터페이스 번호에 따라 소팅된 순서로 리스트에 정보(라우터 주소 및 수신 인터페이스 번호)를 추가하도록 수정. 이와 관련하여 ipv6_ra_host_router_list_entry_compare_proc(...) 함수도 라우터 리스트가 RA 메시지 수신 인터페이스 번호에 따라 소팅(sorting)될 수 있도록 수정

④ host 노드가 라우터로부터 일정시간동안 RA 메시지를 수신하지 못하면 해당 라우터 정보를 라우터 리스트에서 삭제하고 디폴트 게이트웨이 라우터를 변경하기 위해 수행하는 함수 ipv6_ra_host_rtr_timeout_handle()에서 멀티홈드 노드의 경우 디폴트 게이트웨이 라우터의 변경과 함께 이 라우터로 연결되는 디폴트 인터페이스 정보도 변경되도록 코드 수정

<mipv6_mgr.pr.m >

① MIPv6관련 속성 값을 읽어 들이는 함수 mipv6_attributes_read()에서 멀티홈드 노드의 경우 host와 같이 노드 속성 IP > Mobile IP Host Parameters > Mobile IPv6 Parameters 에서 관련 값들을 읽어 들이도록 코드 수정

<ipv6_nd.pr.m> (ARP계층에서 동작)

① ARP계층이 MAC계층으로부터 IPv6 ICMP 패킷을 수신하였을 때 수행되는 함수 ipv6_nd_mac_packet_handle(...)에서 멀티홈드 노드가 Router Solicitation 메시지나 Router Advertisement 메시지 수신시 host와 같은 방법으로 처리하도록 수정

② 함수 ipv6_nd_ns_msg_handle(...)은 neighbor

solicitation 메시지를 ARP 모듈이 수신했을 때 수행되는 함수로 패킷의 target 주소가 ARP모듈이 속한 인터페이스의 주소중 하나일 때 neighbor advertisement 메시지로 응답하도록 되어있음. 그러나 멀티홈드 노드의 경우에는 target 주소로 LB0(또는 CoA)가 사용될 수도 있으므로 이 경우에도 neighbor advertisement 메시지로 응답하도록 코드 수정

4. 시뮬레이션 결과 및 분석

본 논문에서 개발된 MIPv6 호스트 모드 지원 멀티홈드 노드의 시뮬레이션 환경을 그림 5에 나타내었다. 그림에서 좌측에 있는 멀티홈드 노드(multihomed_1_1)은 WiMAX[3] 망의 베이스 스테이션 BS1, BS2와 WLAN[4] 망의 액세스 포인트 AP2, AP1 사이를 이동하며, 우측에 있는 멀티홈드 노드(multihomed_1_2)는 WLAN 망의 액세스 포인트 AP6, AP7과 WiMAX 망의 베이스 스테이션 BS7, BS6 사이를 MIPv6를 이용해 이동하면서 서로에게 UDP 데이터를 송수신한다.

그림 6에는 시뮬레이션 결과로 수신된 데이터 트래픽, WiMAX망과 WLAN 망의 throughput을 각각 나타내었다. 그림 6(a)는 multihomed_1_1 노드와 multihomed_1_2가 그림 5에 나타낸 시뮬레이션 환경과 같이 서로 다른 무선기술을 사용하는 망에 위치하도록 구성하고 시뮬레이션한 결과이고, 그림 6(b)는 multihomed_1_1 노드와 multihomed_1_2 노드가 서로 같은 무선기술을 사용하는 망에 동시에 위치하도록 구성하고 시뮬레이션한 결과이다. 그림으로부터 멀티홈드 노드들은 현재 자신에게 가용한 무선기술(WiMAX 또는 WLAN)에 따른 인터페이스간 핸드오버를 수행하여 데이터 트래픽을 잘 전송하고 있음을 볼 수 있다.

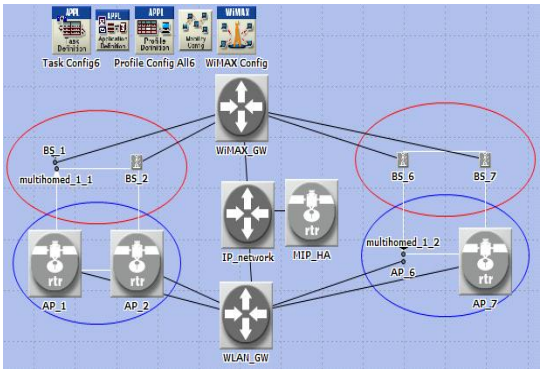
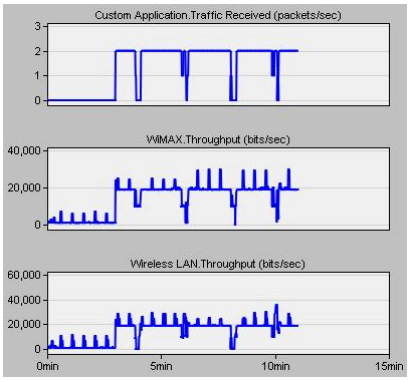
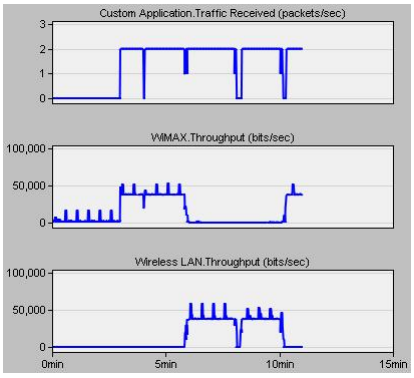


그림 5. 멀티홈드 노드 MIPv6 시뮬레이션 환경



(a) 두 멀티홈드 노드가 서로 다른 무선기술을 사용하는 망에 위치하는 경우



(b) 두 멀티홈드 노드가 같은 무선기술을 사용하는 망에 위치하는 경우

그림 6. 멀티홈드 노드 시뮬레이션 결과

그림 7에는 시뮬레이션 결과중 MIPv6 관련 HA(Home Agent) 바인딩 지연시간과 터널을 통

해 전송된 트래픽 양을 나타내었다. 데이터 트래픽은 핸드오버 발생시 초기에는 HA와의 터널을 통해 전송되지만 MIPv6의 optimization 기능을 통해 곧 멀티홈드 노드끼리 직접 데이터 트래픽을 전송하게 된다.

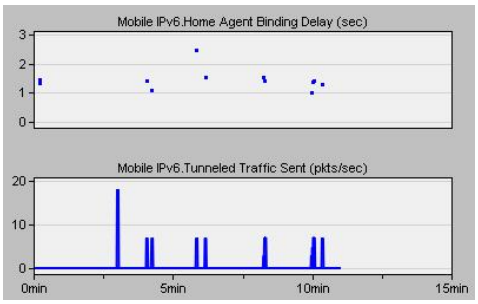


그림 7. Mobile IPv6 관련 시뮬레이션 결과

5. 결 론

무선 통신 기술의 발달로 2개 이상의 무선 멀티 인터페이스를 갖는 멀티홈드(multihomed) 이동 단말이 출현하고 이에 따라 멀티홈드 노드의 인터페이스간 부하분담 또는 핸드오버에 관한 연구수행을 위해 시뮬레이션 모델의 필요성이 대두되고 있다. 네트워크 연구분야에서 널리 사용되고 있는 시뮬레이션 소프트웨어 OPNET은 멀티홈드 노드 자동 생성기능을 지원하고 있지만 생성된 멀티홈드 노드가 gateway 모드로 동작하도록 되어 있기 때문에 Mobile IPv6의 호스트 동작 모드를 지원하지 못한다. 따라서 본 논문에서는 2개 이상의 인터페이스를 갖는 멀티홈드 노드가 MIPv6 관점에서 host로서 동작하면서 인터페이스간 핸드오버 기능을 수행할 수 있는 시뮬레이션 모델 및 프로그램 코드를 개발하였다. 개발된 프로그램은 서로 다른 무선 기술을 사용하는 2개의 인터페이스간 핸드오버를 실험할 수 있는 환경 하에서 시뮬레이션을 통해 정상동작 여부를 검증하였다.

참 고 문 헌

[1] Xiao-lei Zhang, Ye Wang, Jang-Geun Ki and Kyu-Tea Lee, "Simlation model of a multihomed node with WiMAX and WLAN," The Institute of Webcasting, Internet and Telecommunication, Vol.10-3-15, pp.111-119, June 2010.
[2] OPNET Simulator, http://www.opnet.com, 2012.
[3] WiMAX, http://www.wimaxforum.org/
[4] WLAN, http://www.ieee802.org/11/
[5] Karl Andersson, ANM Zaheduzzaman Sarker, Christer Ahlund, "Multihomed Mobile IPv6: OPNET Simulation of Network Selection and Handover Timing in Heterogeneous Network Environments", HybriNet@Skellefte project Sweden, 2008.
[6] Deguang Le, Xiaoming Fu, Dieter Hogrefe, "Evaluation of Mobile IPv6 Based on an OPNET Model", the 8th International Conference for Young Computer Scientists (ICYCS'05), Beijing, September 2005.

저 자 소 개

기장근 (奇長根)

1986.2 고려대학교 전자공학과졸업
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