주 제

모바일 Ad-hoc 환경에서의 라우팅 기술

중앙대학교 김준년, 남호석, 김민근, 조솔

차례

I. 서 론

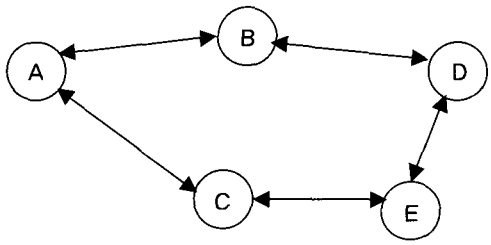
최근 정보통신산업의 급격한 발달과 더불어 센서 네트워크에 대한 관심이 고조되고 있다. 센서 네트워크는 별도의 인프라 구축 없이 ad-hoc 환경에서 언제, 어디서나 네트워킹이 가능한 환경을 제공한다. 또한 가전 및 통신산업 분야에서 임베디드 기술의 발전으로 모바일 디바이스들간의 통신에 대한 요구가 증대되고 있다.

센서 네트워크 기술이 구현될 경우 정보통신 산업뿐만 아니라 환경, 국방, 의료 등 여러 산업분야에서 새로운 패러다임을 형성할 것으로 보인다. 센서 네트워크를 구현하기 위해서는 신호처리, 네트워크 및 프로토콜, 임베디드 시스템 및 분산처리 등 여러 가지 기술을 요구한다. 따라서 본 논문에서는 센서 네트워크를 구현하는데 있어서 핵심적인 역할을 하는 ad-hoc 환경에서의 라우팅 프로토콜 기술에 대해 고찰하고자 한다.

II. Proactive Protocol

모바일 환경에서 라우팅은 발신지와 목적지 사이의 경로를 설정하는 것이 가장 큰 이슈이다. 프로액티브 프로토콜은 컨트롤 메시지의 주기적인 교환을 통해서 라우팅 테이블을 유지한다. 각각의 노드들은 완성된 네트워크 토폴로지 정보를 저장하고 있다. 노드들은 전체적인 네트워크 토폴로지 정보를 알고 있기 때문에, 목적지에 이르는 최선의 경로를 즉시 찾을 수 있다. 하지만 프로액티브 프로토콜은 많은 양의 컨트롤 메시지를 발생 시키기 때문에 이용 가능한 대역폭의 많은 부분을 컨트롤 메시지가 사용한다. 특히 노드의 숫자가 많거나 노드들의 이동성이 증가하면 대역폭의 거의 대부분을 컨트롤 메시지에 사용된다.

2-1. Distributed Bellman-Ford 알고리즘

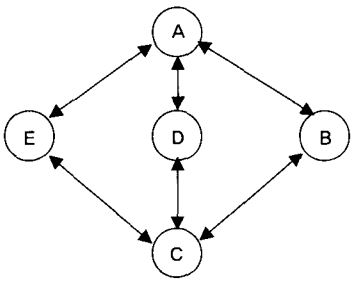


(그림 1) Distributed Bellman - Ford Algorithm

Distributed Bellman-Ford 알고리즘은 가장 짧은 경로를 찾는 방법이다. (그림 1)에서 발신지에서 목적지로 보내는데 필요한 메트릭을 홉의 개수로 생각한다. A가 D로 메시지를 보낸다고 가정할 경우, A에서 C와 E를 거쳐 D로 가는 방법과 A에서 B를 거쳐 D로 가는 방법이 있다. A는 둘 중 가장 짧은 경로인 A에서 B를 거쳐 D로 가는 방법을 선택한다.

Distributed Bellman-Ford 알고리즘은 모든 경로에 대한 정보를 가지고 있지만 경로에 대한 정보가 신속하게 갱신되지 않는다. 그렇기 때문에 루프와 infinity라는 2가지 문제를 발생시킨다.

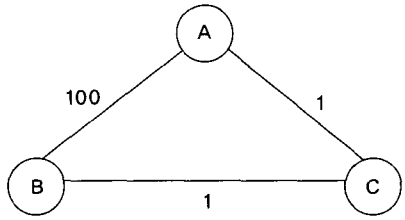
2-1-1 루프의 발생



(그림 2) 루프의 발생

(그림 2)에서 A에서 E로 메시지를 보내고 A-E, C-E의 링크가 단절되었다고 가정하자. A는 E와의 직접 연결이 단절된 것을 알 수 있으므로 D에게 메시지를 보낸다. D는 받은 메시지를 C로 보내고, C는 C-E가 단절되었으므로 E로 가는 다른 방법인 A-E를 이용할 수 있도록 B로 메시지를 보내게 된다. 따라서 A-D-C-B-A로 이어지는 루프가 생성되게 된다.

2-1-2 Infinity 문제



(그림 3) Infinity 문제

(그림 3)에서 A-C의 링크가 단절되었다고 가정할 경우, B의 관점에서는 아직 A-C의 링크가 단절된 것을 모른다. B에서 C를 거쳐 A으로 가는 메트릭은 2이기 때문에, B는 C로 메시지를 전송하게 된다. 메시지를 받은 C는 A으로 가는 링크가 단절 되었기 때문에, 받은 메시지에 메트릭 1을 더해서 B로 다시 보내게 된다. B는 C에서 받은 경로의 메트릭과 B-A의 메트릭을 비교하게 되는데, B-A의 메트릭은 100이고 받은 메트릭은 3이기 때문에, 다시 C로 메시지를 보내게 된다. 이 메시지는 B-C의 경로비용이 100이 넘을 때까지 계속해서 C로 보내지게 된다.

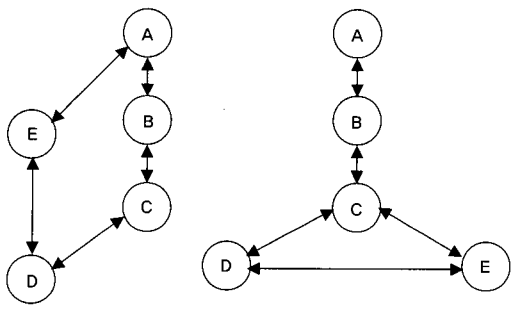
2-2. DSDV (Destination Sequenced Distance Vector) Protocol

DSDV 프로토콜은 기본적으로 Bellman-Ford알고리즘을 사용하지만, 루프와 infinity를 해결 하기 위해서 순차번호라는 개념을 도입하였다. 각 노드들은 주기적으로 컨트롤 메시지를 브로드캐스팅하며, 그

메시지에 순차번호를 포함하여 보낸다. 메시지를 받은 노드들은 자신의 라우팅 테이블에 있는 엔트리와 받은 메시지의 순차번호를 비교하여, 받은 메시지의 순차번호가 크면 새로운 메시지로 간주하여 자신의 라우팅 테이블을 갱신하게 된다.

2-2-1 순차번호에 의한 경로 갱신

라우팅 테이블은 목적지에 이르는 홉의 개수와 이용 가능한 목적지의 리스트, 그리고 목적지로부터 받은 순차번호를 가지고 있다. DSDV 프로토콜은 각각의 노드가 자신의 라우팅 테이블에 변화가 있을 경우 이를 광고한다. 이웃한 노드들은 주기적인 메시지와 광고 메시지를 통해 자신의 엔트리를 빠르게 갱신 할 수 있다.



(그림 4) 순차번호에 의한 경로 갱신의 예

(그림 4(a))에서, 각 노드들이 경로 갱신을 시작하였을 때 D의 관점에서 보면, E=1, C=1, B=∞, A=∞임을 알 수 있다. 모든 노드들은 순차번호를 1로 하고 브로드캐스팅한다. 다음에는 C가 보낸 메시지를 B가 받을 것이고 B의 거리는 2라는 것을 D가 알 수 있을 것이다. 마지막으로, B가 보낸 메시지를 A가 받을 것이고 D가 가지는 경로의 거리는 E=1, C=1, B=2, A=3으로 갱신 될 것이다.

(그림 4(b))에서 E가 새로운 위치로 이동했다고 가정한다. A은 자신의 이웃에 E가 추가 되었음을 알 수 있다. A은 자신의 이웃이 된 E에게 새로운 메시지를 보내게 되며, 순차번호는 2로 설정 된다. 이 메시지를 받은 E는 자신의 이웃인 D에게 다시 전송해 주게 되며, D는 이 메시지를 받은 후 자신의 라우팅 테이블에 저장되어있는 것과 비교하게 된다. 새로운 메시지의 순차번호가 이전에 있던 것 보다 큰 숫자이기 때문에, D는 새롭게 받은 메시지로 자신의 라우팅 테이블을 갱신하게 된다.

2-2-2 DSDV 프로토콜의 장점과 단점

DSDV 프로토콜에서 각 노드들은 이웃하는 노드들과 주기적인 통신을 통해 토폴로지 변화를 감지하고, 테이블을 갱신 할 수 있다. 따라서 경로를 발견하는데 매우 효율적이며, 그 시간도 짧다. 그리고 Bellman-Ford 알고리즘의 문제점인 루프와 infinity 문제도 개선하였다. 하지만 경로를 찾는 과정에서의 많은 컨트롤 메시지를 발생 시키기 때문에, 이를 개선 할 수 있어야 많은 수의 노드를 갖는 네트워크나 이동성이 많은 네트워크에서 사용할 수 있을 것이다.

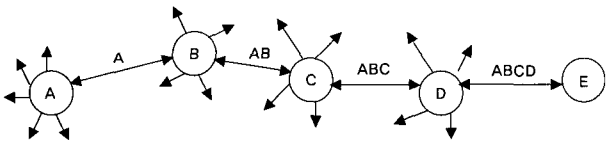
III. Reactive Protocol

3-1. DSR (Dynamic Source Routing) Protocol

DSR은 DSDV와는 달리 요구 기반(on-demand) 방식의 라우팅 프로토콜이다. 다시 말해, 요청이 있을 때에만 컨트롤 메시지를 발생시킨다. 따라서 DSDV 프로토콜에 비해 컨트롤 메시지가 매우 적으며, 에너지 효율적인 프로토콜이다. 하지만 노드들이 움직이면서 통신을 하면 경로가 변하게 되며, 그것에 비례하

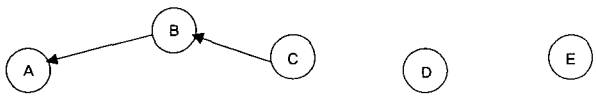
여 패킷 오버헤드가 증가하게 된다. 각 노드들은 목적지에 이르는 다양한 경로를 루트 캐시에 저장하고 있으며, 한 경로가 단절되었을 때 루트 캐시를 참조하여 빠르게 다른 경로를 설정 할 수 있다.

3-1-1 DSR 프로토콜의 경로 발견



(그림 5) DSR 프로토콜의 경로발견

(그림 5)는 DSR 프로토콜에서 노드 A가 목적지 노드 E까지의 경로를 설정하는 과정이다. A는 자신의 루트 캐시를 검색하여 E까지의 경로가 없음을 알고 자신의 이웃에게 E까지의 경로를 요청하는 메시지를 브로드캐스팅한다. 그 메시지를 받은 B는 자신의 루트 캐시를 검색한 후 찾지 못할 경우 패킷의 마지막 부분에 자신의 ID를 붙인 후 브로드캐스팅한다. 이러한 방식으로 목적지인 노드 E까지 전달된다. 노드 E는 이 패킷이 거쳐온 노드들을 모두 알 수 있으며, 그 경로를 통해 응답 메시지를 보낸다.

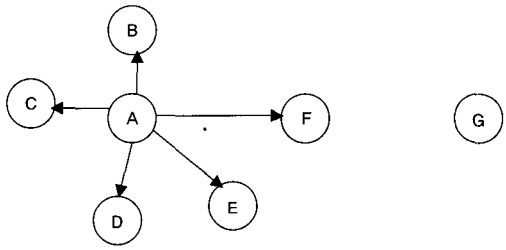


(그림 6) 중간 노드가 목적지에 이르는 경로를 알고 있을 경우 응답 메시지

(그림 6)은 중간 노드 C가 목적지인 E에 이르는 경로를 알고 있을 때, 응답 메시지를 A로 보내주는 것을 보여준다. 경로 요청메시지를 받은 C는 자신의 루트 캐시를 검색하여 E의 경로를 자신이 갖고 있음을 알 수 있다. 따라서 C는 경로 요청 메시지를 D로 전송하지 않고 자신이 직접 응답메시지를 A로 전송하게 된다.

만약 A가 E로 가는 응답 메시지를 받지 못했을 경우에, A는 지금은 E로 가는 경로가 없다고 간주하여 계속적인 요청 메시지를 보내지 않는다. 계속해서 요청 메시지를 보낼 경우에는 많은 오버헤드가 발생하기 때문이다.

3-1-2 Route Reply Storm



(그림 7) 각 노드들이 G로 가는 경로를 알고 있을 경우

(그림 7)에서와 같이 A의 이웃하는 노드들 중 B,C,D,E,F 가 모두 G로 가는 경로를 알고 있을 경우에 각각의 노드들이 모두 A로 응답 메시지를 보내는 것을 Route Reply Storm이라 한다. 이렇게 동시에 응답 메시지가 보내지면 네트워크 혼잡이나 패킷들이 충돌을 일으킬 수 있다. 이것을 방지하기 위해 각 노드는 임의의 시간만큼을 대기하며 흐름을 관찰한다. 이 시간 동안 A가 가장 짧은 경로 응답 메시지를 받아서 목적지까지의 경로를 설정하였다면, 다른 노드들은 응답 메시지를 보낼 필요가 없게 된다.

3-1-3 DSR 프로토콜의 장점과 단점

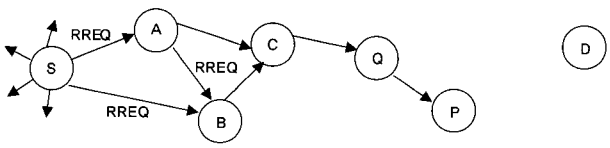
DSR은 간단하고 효율적인 라우팅 프로토콜이며, DSDV와 비교하여 컨트롤 메시지의 오버헤드가 매우 적은 프로토콜이다. 그러나 DSR은 경로를 발견하는데 매우 오랜 시간이 걸리며, 노드의 홉 수가 늘어남에 따라 패킷의 크기가 증가하기 때문에 확장에 제한이 있는 단점이 있다.

3-2. AODV (Ad-hoc On-Demand Distance Vector) Protocol

AODV 프로토콜은 요구기반 방식의 리액티브 프로토콜이며 라우팅 테이블을 사용한다. AODV는 앞 절에서 설명한 DSR과는 달리 패킷 사이즈가 일정하며 DSDV와 달리 로컬의 토폴로지 변화에서 대해서 전체적인 컨트롤 메시지의 브로드캐스트가 필요하지 않다. AODV는 발신지와 목적지 사이의 하나의 경로를 유지하며 일정 시간이 경과하면 엔트리에서 삭제된다. AODV는 경로 탐색을 위해서 RREQ와 RREP라는 메시지를 사용하며 DSDV에서 사용했던 것처럼 순차번호를 사용해서 경로의 최신성을 유지한다.

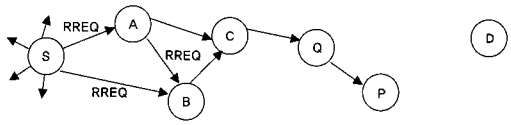
3-2-1 경로탐색

AODV에서 경로탐색은 RREQ메시지를 전송을 통해서 이뤄진다. 발신지 노드가 목적지 노드에 메시지 전송을 필요로 할 때 발신지 노드는 라우팅 테이블을 검색한 후, 경로가 존재하지 않을 때 RREQ 메시지를 발생한다. RREQ 메시지는 발신지와 목적지의 IP 주소, 발신지와 목적지의 순차번호 및 브로드캐스트 ID를 포함한다. 여기서 브로드캐스트 ID와 IP 주소는 RREQ 메시지를 구별하는 식별자로 사용한다. (그림 8)에서와 같이 노드 B는 노드 S부터 RREQ 메시지를 수신했을 경우, 노드 A로부터 수신된 RREQ 메시지는 동일한 메시지이므로 폐기하고 브로드캐스트하지 않는다.



(그림 8) 중복된 RREQ를 수신한 경우

임의의 노드는 RREQ 메시지를 수신했을 경우, 노드는 발신지와 목적지 IP주소와 순차번호, 홉 카운트와 RREQ 메시지를 받은 이웃 노드의 IP 주소를 리버스 루트 엔트리에 저장하는데 이 과정을 리버스 경로 설정이라고 한다. (그림 9)는 각각의 노드들에 대한 리버스 루트 엔트가 설정되는 과정을 보여준다.

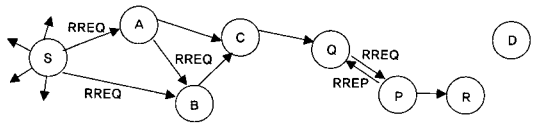


(그림 9) 역방향 루트 엔트리 설정

Node	Source Address	Destination Address	Source Sequence #	Hop Count	Neighbor Address
A	S	D	1	1	S
B	S	D	1	1	S
C	S	D	1	2	A
D	S	D	1	3	C
P	S	D	1	4	Q

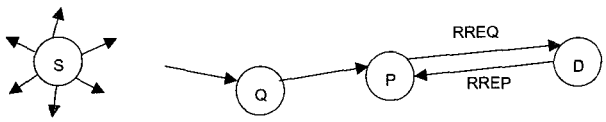
역방향 경로 설정을 하는 이유는 해당 RREQ 메시지에 대한 응답으로 RREP 메시지를 수신했을 경우 RREQ 메시지를 발생한 이웃 노드에게 RREP 메시지를 전송하기 위해서이다.

(그림 10)과 같이 RREQ 메시지가 목적지 주소를 알고 있는 임의의 노드 P에 도착했을 때 노드 P는 RREP 메시지로 응답한다. 여기서 노드 P는 파기되지 않은 목적지 D에 대한 엔트리를 가지고 있어야 하며, 노드 P가 가지고 있는 목적지 D에 대한 순차번호가 RREQ 메시지에 있는 순차번호와 비교해서 작아서는 안 된다. 이는 발신지와 목적지 사이에 루프가 형성되는 것을 방지한다.



(그림 10) 중간 노드의 RREP 응답

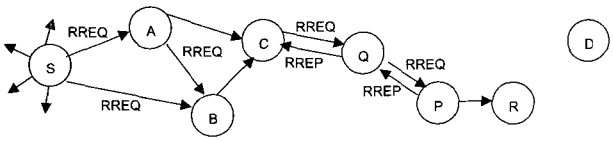
(그림 11)에서와 같이 노드 P가 발신지 노드 S로부터의 RREQ 메시지에 대한 응답을 할 수 없을 경우 노드 P는 RREQ 메시지의 홉 카운터를 증가시키고 이웃 노드에 브로드캐스팅한다. 최종적으로 목적지 노드 D가 RREQ메시지를 수신했을 겨우 순차번호가 가장 크므로 RREP 메시지를 발생 할 수 있다.



(그림 11) 목적지 노드의 RREP 응답

3-2-2 순방향 경로 설정

RREP 메시지는 발신지와 목적지의 IP 주소, 목적지 순차번호 홉-카운트 및 수명시간을 포함하며 목적지가 RREP 메시지를 발생하는 경우 현재 목적지의 순차번호와 경로에 대한 수명시간과 홉-카운터는 0으로 설정해서 보낸다. 중간 노드가 RREP 메시지를 발생하는 경우 목적지로부터 마지막으로 받은 순차번호와 목적지로부터의 홉 카운트를 설정해서 보낸다. (그림 12)와 같이 중간 노드들이 RREP 메시지를 수신했을 경우 각각의 노드들은 목적지 노드에 대한 정보를 라우팅 테이블에 저장한 후, RREQ 메시지를 수신한 노드로 RREP 메시지를 전송한다. 이 과정을 순방향 경로 설정이라고 한다.

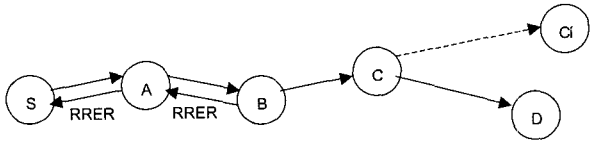


(그림 12) RREQ 응답

중간 노드 P는 동일한 RREQ 메시지에 대하여 다수의 RREP 메시지를 수신 할 수 있는데, RREP 메시지의 순차번호가 더 큰 경우와 목적지로의 홉-카운트가 작은 경우에 대해서는 RREP 메시지를 포워딩하고 그 외는 폐기한다. 이는 발신지로 전송되는 RREP 메시지의 수를 줄인다.

3-2-3 경로유지

발신지와 목적지 사이의 유니캐스트 경로가 확보되면 경로는 라우팅 테이블로 유지한다. 발신지 노드가 이동해서 목적지에 대한 경로를 확보할 수 없으면 발신지 노드는 RREQ 메시지를 다시 발생 시키며 중간 노드들의 이동에 의해서 목적지에 도달할 수 없으면 발신지 노드로 REER 메시지가 전송된다. (그림 13)에서와 같이 노드 C의 이동으로 인해서 목적지로의 링크가 단절된 경우 노드 B는 노드 A에 RRER 메시지를 전송한다. 그러면 발신지 노드 S는 RREQ 메시지를 다시 발생한다.



(그림 13) REER 메시지의 발생

AODV에서 이웃 노드들의 정보는 주기적으로 hello 메시지의 교환을 통해서 이뤄지며, hello 메시지는 1-홉만 전송된다.

3-2-4 AODV 프로토콜의 장점과 단점

AODV는 손실된 패킷에 대한 재전송을 지원하지 않으며 따라서 패킷 전송에 신뢰성을 확보할 수 없다. 그렇지만 상대적으로 작은 숫자의 노드에 대해서는 패킷 전송률은 100%에 가깝다. 또한 이동성이 증가할수록 패킷 전송률 또한 떨어진다. AODV에서 패킷

오버헤드는 RREQ, RREP 및 RERR 메시지로 인한 것이며 DSDV와 비교했을 때, 패킷 오버헤드는 상대적으로 훨씬 작다. 이동성이 증가했을 때 잦은 링크의 단절과 경로 탐색으로 인한 패킷 오버헤드 또한 증가한다. AODV에서 경로 탐색 지연시간은 DSR과 DSDV와 비교했을 때 작다.

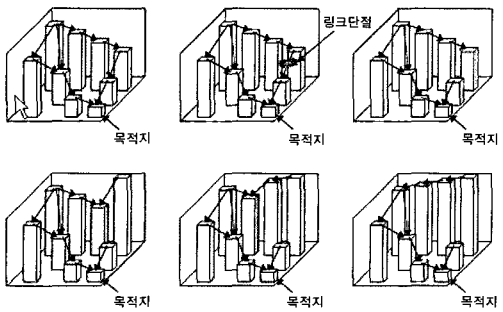
3-3. TORA (Temporally Ordered Routing Algorithm)

TORA에서는 GB(Gafni-Bertsekas) 알고리즘의 QRY-RPY 메커니즘과 LMR(Lightweight Mobile Routing) 알고리즘의 부분 링크 역전(partial link reversal) 메커니즘을 보완하여 개발되었다.

TORA의 주요한 특징은 링크가 끊어졌을 시의 대응 방법에서 나타난다. TORA에서는 분산 알고리즘인 single-pass 알고리즘을 이용하여, 유효하지 않은 경로를 제거하고 새로운 경로를 찾는다. Single-pass 경로 탐색을 통해, 노드의 수가 많은 네트워크에서 컨트롤 메시지의 오버헤드를 많이 발생시키지 않으면서 경로를 재설정 할 수 있다.

3-3-1 노드의 높이

링크의 방향은 (그림 14)에서와 같이 DAG (Directed Acyclic Graph)에서 링크를 정의하는 두 노드 사이의 상대적인 높이에 따라 결정되며, 링크 실패가 발생할 시 각 노드들은 각각의 높이를 바꾸어줌으로써 링크의 방향을 최종 목적지로 향하도록 설정한다.



(그림 14) 링크 단절에 따른 노드 높이의 변화

주어진 시간에서의 각 목적지에 대한 노드의 높이는 다음과 같이 정의된다.

$$H_i = (\tau_i,\ oid_i,\ r_i,\ \delta_i,\ i)$$

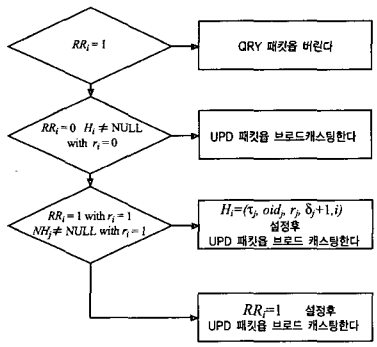
위의 식에서 $(\tau_i,\ oid_i,\ r_i)$를 참조 레벨이라고 하며, $(\delta_i,\ i)$는 참조 레벨에 따른 오프셋이라 한다. 여기서, τ_i는 링크 실패 시의 논리적 시간을 의미하고, oid_i, δ_i, i)는 참조 레벨을 정의한 노드의 ID, r_i는 반사를 나타내는 비트, δ_i는 전파 파라미터 그리고 i는 노드의 ID를 의미한다.

각 노드 i는 각 목적지에 따라 서로 다른 H_i를 유지한다. 초기 상태에서는 각 노드의 높이는 null인 상태(즉, $H_i=(-,-,-,-,i)$)로 설정되며, 목적지의 높이는 언제나 0(즉, $H_i=(0,0,0,0,dest)$)으로 설정된다. 각 노드 i는 그 노드 주위의 이웃들(즉, $j \varepsilon N_i$)에 대하여, 높이 배열인 $HN_{i,j}$를 유지한다. 노드 i와 이웃 노드 j 사이의 링크의 방향은 H_i와 $HN_{i,j}$에 의해 결정된다. 만약 i의 높이보다 j의 높이가 더 높을 경우, i와 j사이의 링크는 업스트림(UP)으로 설정될 것이고, 반대의 경우는 다운스트림(DN)으로 설정될 것이다. 만약 $HN_{i,j}$이 null로 설정되어 있다면, 그 링크는 설정되지 않은 상태(UN)라는 것을 의미한다.

3-3-2 경로 생성

LMR 알고리즘에서와 마찬가지로, TORA 역시 경로 생성에 QRY 패킷과 UPD 패킷을 이용한다. QRY 패킷은 DEST 필드로 구성되어 있고, UPD 패킷은 DEST 필드와 H_i로 구성되어 있다. 각 노드 i는 RR_i (route required flag, 초기에는 0)와 UPD 패킷이 마지막으로 브로드캐스팅 된 시간을 저장하고 있

다. 임의의 한 노드에서 링크의 방향이 설정되어 있지 않고 RR_i=0 이라면, DEST에 대한 경로를 요구한다는 뜻이며, QRY 패킷을 브로드캐스팅 한 후, RR_i를 1로 설정한다. 각 노드가 QRY 패킷을 받았을 때 노드 i가 반응하는 방법은 (그림 15)와 같이 4가지가 있다.



(그림 15) QRY 패킷을 수신한 노드 i의 반응 방법

Case 1

RR_i=1이면, i가 이미 DEST에 대한 QRY를 받았다는 의미 의미이므로 QRY 패킷을 처분한다.

Case 2

RR_i=0, r_i=0이고 H_i가 null이 아니라면, 노드 i는 UPD 패킷을 보내야 한다.

Case 3

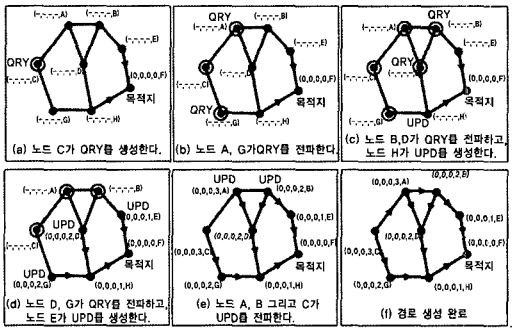
노드 i에서 RR_i=0, r_i=1 이고(H_i는 null이어도 되고, null이 아니어도 된다), 이웃 노드 j에서 높이가 null이 아니고, r_j=0 이면, 노드 i는 H_i를 다음과 같이 설정한다:

Case 4

위의 조건들이 모두 만족하지 않는 경우, 즉 RR_i=0 이고, 이웃 노드들이 전부 유효하지 않은 높이를 가지고 있을 때, 노드 i는 QRY 패킷을 브로드캐스팅하고, RR_i를 1로 설정한다.

$$H_i=(\tau_i, oid_j, r_j, \delta_j+1, i)$$

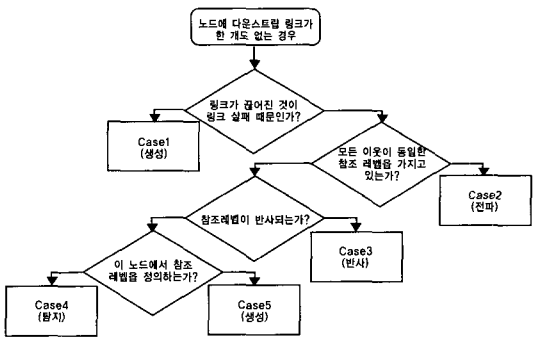
경로 생성 시, 앞서 설명한 4가지 경우에 따라 QRY 혹은 UPD 패킷를 브로드캐스팅하게 되며, (그림 16)에 경로 생성에 대한 예가 제시되고 있다.



(그림 16) 경로 생성의 예

3-3-3 경로 유지

경로 유지는 높이가 null이 아닌 노드들에 의해서만 수행된다. 다운스트림 링크를 하나도 지니고 있지 않은 노드에서는 반드시 그 노드의 높이가 수정되어야 한다. 링크 실패가 발생하거나 UPD 패킷을 받아 이웃 노드들의 높이가 변화 될 때, 특정 노드에서 다운스트림 링크를 가지지 않게 되는 경우가 발생할 수 있다. 앞서 언급한대로, 다운스트림 링크를 지니고 있지 않은 노드에서는 그 노드의 높이가 수정되어야 하는데, (그림 17)과 같이 5가지 기준을 근거로 높이가 수정된다.



(그림 17) 다운스트림 링크를 지니지 않은 임의의 한 노드에서의 반응 방법

Case 1: 생성

노드 i에 다운스트림 링크가 없는 원인이 링크 실패이고, 노드 i주변에 업스트림 이웃들이 존재할 경우, 노드 i는 새로운 참조 레벨을 다음과 같이 정의한다: $(\tau_i, oid_i, r_i) = (t, i, 0)$, $(\delta_i, i) = (0, i)$

Case 2: 전파

UPD 패킷의 수신으로 인해 링크 역전(link reversal)이 일어날 때 발생하는 경우이다. 만약 모든 $j \varepsilon N_i$에 대하여 (τ_i, oid_i, r_i)가 동일하지 않으면, 노드 i는 자신의 높이를 다음과 같이 정의한다: $(\tau_i, oid_i, r_i) = \max\{(\tau_j, oid_j, r_j) \mid j \varepsilon N_i\}$, $(\delta_i, i) = (\min\{\delta_j\} - 1, i)$ 여기서 δ_j는 이웃들 중 $\max\{(\tau_j, oid_j, r_j)$인 것으로 선택된다.

Case 3: 반사

UPD 패킷의 수신으로 인해 링크 역전(link reversal)이 일어날 때 발생하는 경우이다. 모든 $j \varepsilon N_i$에 대하여 (τ_i, oid_i, r_i)가 동일하고 $r_j = 0$ 이면, 노드 i는 자신의 높이를 다음과 같이 정의한다: $(\tau_i, oid_i, r_i) = (\tau_j, oid_j, 1)$, $(\delta_i, i) = (0, i)$

모든 이웃들이 같은 높이를 가지고 있고 $r_j = 0$ 이라는 것은 그 이웃들의 높이가 아직 반사(reflection)되지 않았다는 것을 의미한다. 이제 노드 i는 r_i를 1로 설정하여 자신의 높이를 반사하고, 미래에 동일한 경우가 발생하면 네트워크 단절(network partition)을 감지할 수 있게 된다.

Case 4: 탐지

UPD 패킷의 수신으로 인해 링크 역전(link reversal)이 일어날 때 발생하는 경우이다. 모든 $j \varepsilon N_i$에 대하여 (τ_i, oid_i, r_i)가 동일하고 $r_j = 1$ 이며 $oid_j = i$이면, 노드 i는 자신의 높이를 다음과 같이 정의한다: $(\tau_i, oid_i, r_i) = (t, i, 0)$, $(\delta_i, i) = (0, i)$

Case 5: 생성

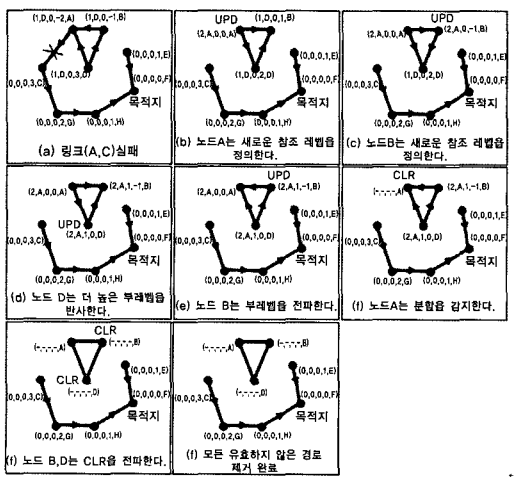
UPD 패킷의 수신으로 인해 링크 역전(link reversal)이 일어날 때 발생하는 경우이다. 모든 $j \varepsilon N_i$에 대하여 (τ_i, oid_i, r_i)가 동일하고 $r_j = 1$이며 $oid_j \neq i$이면, 노드 i는 자신의 높이를 다음과 같이 정의한다: $(\tau_i, oid_i, r_i) = (t, i, 0)$, $(\delta_i, i) = (0, i)$

3-3-4 경로 제거

경로 유지의 case 4에서 노드 i는 자신의 높이와 각 이웃에 대한 높이 엔트리를 null로 설정한다. 노드 i는 자신의 링크 상태 배열(link status array) 내부의 모든 엔트리를 업데이트하고 CLR 패킷을 브로드캐스팅 하는데, CLR 패킷은 DEST와 노드 i의 반사된 참조 레벨인 $(\tau_i, oid_i, 1)$로 이루어져 있다. 노드 i가 이웃으로부터 CLR 패킷을 받았을 경우, 다음과 같이 반응한다.

(1) 만약 CLR 패킷 내부의 참조 레벨이 노드의 참조 레벨과 일치한다면, 노드는 자신의 높이와 각 이웃들에 대한 높이 엔트리를 null로 설정한다. 또한 노드는 자신의 링크 상태 배열(link status array) 내부의 모든 엔트리를 업데이트하고 CLR 패킷을 브로드캐스팅 한다.

(2) 만약 CLR 패킷 내부의 참조 레벨이 노드의 참조 레벨과 일치하지 않는다면, 노드는 각 이웃



(그림 18) 경로 제거의 예

들에 대한 높이 엔트리를 null로 설정하고 자신의 링크 상태 배열(link status array) 내부의 모든 엔트리를 업데이트한다. (그림 18)에 경로 제거에 대한 예가 제시되고 있다.

IV. 결 론

지금까지 본 논문에서는 ad-hoc 환경에서 현재 활발히 연구가 진행 중인 라우팅 프로토콜기술을 고찰하였다. 그 중 대표적인 DSDV, AODV, DSR 및 TORA의 라우팅 기술의 원리와 특징을 고찰하였다. 이러한 원천 기술을 확보하고 장기적으로 유비쿼터스 네트워킹 환경으로 진입하기 위해서 기존 유선 네트워크에서 사용되는 여러 프로토콜 호환성이 보장되고 무선 네트워크에서도 원활한 통신이 가능한 새로운 통신 기술에 대한 연구와 표준화가 필요하며 정책적 지원도 절실하다. 21세기 성장동력 산업으로서 IT분야에서 센서 네트워크에 대한 기술 확보로 다가오는 유비쿼터스 시대를 준비할 수 있을 것이다.

[참 고 문 헌]

[1] Josh Broch, David B. Johnson, and David A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-00.txt, March 1998. Work in progress.

[2] David B. Johnson. Routing in ad hoc networks of mobile hosts. In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, pages 158-163, December 1994.

[3] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.

[4] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In Proceedings of INFORCOM'97, pages 1405-1413, April 1997.

[5] Vincent D. Park and M. Scott Corson. Temporally-Ordered Routing Algorithm (TORA) version 1 : Functional specification. Internet-Draft, draft-ietf-manet-tora-spec-00.txt, November 1997. Work in progress.

[6] Charles Perkins. Ad Hoc On Demand Distance Vector(AODV) routing. Internet-Draft, draft-ietf-manet-aodv-00.txt, November 1997. Work in progress.

[7] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing(DSDV) for mobile computers. IN Proceedings of the SIGCOMM' 94 Conference on Communications Architectures, Protocols and Applications, page 234-244, August 1994. A revised version of the paper is available from Http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps.

[8] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. Mobile Computing Systems

and Applications, 1999. Proceedings, WMCSA '99. Second IEEE Workshop on 25-26 Feb. 1999 Page(s):90 - 100.

[9] Eli M. Gafni and Dimitri P. Bertsekas. Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology, IEEE Transactions On Communications, VOL. COM-29, NO.1, January 1981.

김준년

1998년 ~ 현재 중앙대학교 전자전기공학부 교수
2000년 ~ 2006년 ISO/IEC JTC 1/SC 6 Chairman
2001년 ~ 2004년 중앙대학교 연구지원처장/기획실장
2002년 개방형컴퓨터통신연구회(OSIA)회장
2004년 ~ 2005년 한국통신학회 학술이사

남호석

1993년 중앙대학교 전자공학과 학사
1995년 중앙대학교 전자공학과 석사
1995년 ~ 2000년 국방과학연구소 4체계연구부
2001년 ~ 2001년 한국전자통신연구원 무선방송기술연구소
2001년 ~ 2002년 LG전자 DTV 연구소
2003년 ~ 2006년 부천대학 모바일통신과
2005년 ~ 현재 중앙대학교 전자전기공학부 박사과정
관심분야 : 무선통신, 센서세트워크, USN

김민근

2005년 중앙대학교 전자전기공학부 학사
2005년 ~ 현재 중앙대학교 전자전기공학부 석사과정
관심분야 : 센서네트워크

조솔

2006년 중앙대학교 전자전기공학부 학사
2006년 ~ 현재 중앙대학교 전자전기공학부 석사과정
관심분야 : 센서네트워크