韓國 컴퓨터情報學會 2008 제38차 하계학술발표논문집
第16卷 第1號, 2008. 06.

광범위 Community Computing 환경에서의 Community Computing Network를 이용한 수정된 패킷 결합 알고리즘

송 좌 희*, 최 정 대*, 장 훈*, 김 석 윤*

Modified SPaC Algorithm Using the Community Computing Network in huge area Community Computing Environment

Jwa-Hee Song*, Jung-Dae Choi*, Hoon Chang*, Seok-Yoon Kim*

요 약

본 논문에서는 광범위 Community Computing 환경에서의 Community Computing Network의 에너지 효율과 신뢰성을 높이기 위한 수정된 SPaC(Simple Packet Combining)를 제안한다. 제안하는 수정된 SPaC는 같은 패킷을 두 개 이상의 오류가 있는 패킷을 이용하여 에러를 복구하는 기존의 SPaC를 수정하여 특정 threshold 값을 사용하여 감청 시 CPU의 처리량을 줄이고 패리티 패킷을 이용하여 높은 신뢰성과 보다 향상된 에너지 효율을 가진다.

▸ Keyword : Community Computing Network, Community Computing, SPaC

• 제1저자 : 송좌희
* 숭실대학교 컴퓨터학과
※ 본 연구는 산업자원부 지역산업중점기술개발사업의 지원으로 이루어졌습니다.

1. 서론

Community Computing이란 각각의 통신이 가능한 Entity들이 유기적으로 협조하여 특정한 목적을 이루기 위하여 구축되는 환경으로 최근 유비쿼터스 환경을 구축하기 위한 Community grouping 모델로 사용된다. 이러한 환경 속에서 상호간의 유기적인 동작을 위하여 통신하는데 많은 에너지가 소비되고 이는 각 entity 들의 수명을 단축시켜 형성된 group의 역할을 저해한다. 이에 보다 광범위한 Community Computing Group이 이루어진다면 통신을 위한 에너지 효율 문제는 매우 커지게 된다.

또한 통신 수단으로 이동 통신이나 무선 데이터 통신을 이용할 때 신호 감쇠나 노이즈의 영향으로 비트 에러 및 왜곡된 패킷이 전달될 수 있다. 이러한 비트 에러 및 왜곡된 패킷의 이유로 데이터 통신의 신뢰성에 대한 요구조건도 만족시켜야 하는데 이때 데이터 통신의 신뢰성을 높이기 위해서 수신된 데이터 패킷의 오류가 검출 되었을 때 패킷을 재전송을 요청하는 ARQ(automatic repeat request) 프로토콜을 사용함으로써 데이터 통신의 신뢰성을 높일 수 있다. 하지만 재전송을 요청하는 것은 에너지적인 측면에서 많은 비용이 따른다. 따라서 에너지 제약이 심한 Community Computing Network를 이용할 때 재전송은 Community Computing Network의 수명을 단축시키는 일이다.

재전송에 대한 에너지 소모를 줄이기 위한 방법으로는 정방향 오류정정부호(FEC)가 있다.

FEC는 각각의 패킷을 전송 할 때 에러 복구를 위한 추가 비트를 덧붙여 전송하는 방법이다. 즉, 수신부에서 전송된 패킷이 오류가 있으면 FEC에서 추가 비트를 이용하여 에러를 복구한다. FEC는 전송된 데이터의 오류를 정정할 수는 있지만 정정하기 위한 추가 비트가 매우 크다.[1][2] 그래서 point-to-point 통신에서 주로 사용된다. 하지만 Community Computing Network의 대부분의 통신방법은 multi-hop 방식이다.

대부분의 통신방법은 multi-hop 방식이다.

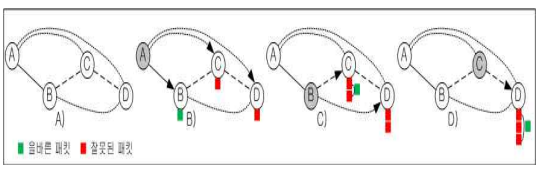

그림 1. multi-hop 에서의 SPaC

Community Computing Network의 multi-hop 방식에서의 전송된 데이터의 오류를 정정할 수 있는 많은 방법이 연구 되었다. 그중 하나가 SPaC(Simple Packet Combining)[5]이다. SPaC는 cooperative communication 방법으로 어떤 노드에서 수신된 데이터를 정정할 필요가 있을 때 데이터를 보낸 노드의 데이터뿐만 아니라 다른 노드에서부터 전송된 데이터를 이용하여 오류를 정정한다. 그림 1처럼 SPaC에서는 노드 A에서 노드 B, 노드 C를 거쳐 노드 D로 데이터를 전송할 때 A는 B로 데이터를 전송한다. 여기서 노드 C와 노드 D는 노드 A로부터의 데이터를 수신할 수 있다. 그러나 노드 C와 노드 D는 노드 A로 부터의 전송된 데이터는 자기의 데이터가 아니므로 버려야 되지만 SPaC는 버퍼에 저장을 한다. 또한 노드 A로부터 전송된 데이터는 노드 C와 노드 D가 수신하기에는 많은 오류를 포함하고 있을 것이다.(그림 1. B) 또한 노드 B는 다음 노드인 노드 C로 데이터를 전송한다. 여기서도 노드 D는 이전단계와 마찬가지로 노드 B에서 받은 데이터를 버퍼에 저장한다. 만약 노드 C에서 수신된 데이터가 오류가 발생하였다면 노드 A로부터 수신된 데이터와 노드 B에서부터 수신된 데이터를 가지고 오류를 복구한다.(그림 1. C) 그리고 노드 C 복구된 데이터를 노드 D로 다시 전송한다. 또한 노드 D에서 노드 C로부터 전송된 데이터가 오류가 발생하였으면 노드 A 그리고 노드 B, 노드 C로부터 전송된 데이터를 가지고 오류를 복구한다.

본 논문에서는 2장에서 기존의 관련연구를 소개하고 3장에서 기존 방식의 문제점과 제안하는 방법을 제시한다. 4장에서는 simulation을 통하여 제안하는 방법을 검증하고 5장의 결론으로 마무리한다.

2. 관련연구

네트워크 환경에서 에러를 처리하는 방법 중 하나가 ARQ이다. ARQ 인 경우 잘못된 패킷이 전송되면 즉시 버리고 재전송을 요청한다. 이러한 재전송 방법은 큰 오버헤드를 가지고 있으며 실시간 통신에 적합하지 않다. 이러한 단점을 줄이고자 고안된 방법이 Hybrid ARQ이다. Hybrid ARQ는 잘못된 패킷이 전송되면 버리지 않고 잘못된 패킷으로부터 패리티 비트를 가지고 복구를 시도한다. 복구가 불가능 할 시 패리티 비트로 구성된 서브패킷만을 재전송 요청하고 패리티 비트 수를 증가시킴으로써 복구 가능하게 한다.[4] SPaC는 Hybrid ARQ와 비슷한 방법을 사용한다. SPaC 역시 잘못된 패킷이 수신되었을 경우 복구를 시도하고 복구가 불가능 하면 Hybrid ARQ와 같은 방법을 취한다. 다만 SPaC는 복구를

시도할 때 Hybrid ARQ와는 방법이 다르다.

SPaC는 원래의 데이터를 linear block codes[3]를 이용한 encoding을 거쳐 원본 패킷과 패리티 패킷으로 나눈 후 전송한다. 이렇게 나뉜 패킷을 multi-hop 방식으로 목적지 까지 전달하는데 경유 노드는 원본 패킷과 패리티 패킷 모두를 받을 수 있다. 경유지 노드 혹은 목적지 노드가 수신된 데이터가 오류가 없다면 그대로 수행하면 된지만 수신된 데이터에 오류가 있다면 복구를 시도한다. 만약 수신된 데이터가 원본 패킷이고 패리티 패킷이면 decoding 작업으로 에러를 복구한다. 하지만 수신된 데이터가 원본패킷과 원본패킷이라면 merge를 통하여 에러를 복구한다. 또한 패리티 패킷과 패리티 패킷 역시 같은 방법으로 복구한다.

그림 2. Merge를 이용한 에러복구

merge를 통한 에러 복구방법은 그림 2와 같이 수신된 두 개 이상의 오류가 있는 패킷을 merge 한다. 그림 2처럼 모든 merge 결과를 가지고 checksum을 이용하여 올바른 패킷을 찾는다. merge를 할 경우 모든 경우가 다 존재함으로 반드시 올바른 패킷이 merge 결과에 존재한다. 그림 3은 경우지 노드나 목적지 노드에서 패킷을 수신하였을 때 수행되는 순서도이다.

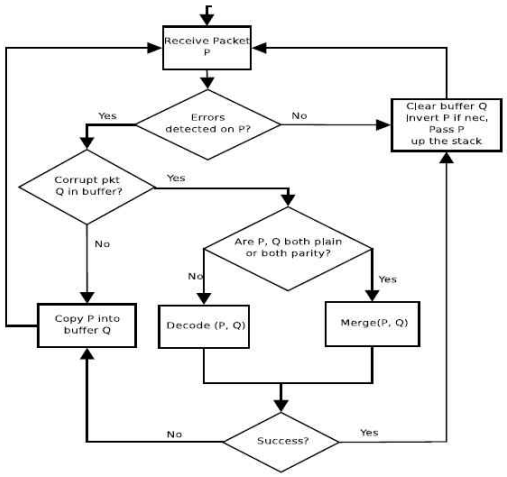

그림 3. 데이터를 수신한 노드의 에러 검출 및 복구 순서도[5]

3. 제안하는 모델

2장에서 SPaC는 데이터 통신의 신뢰성을 높이기 위해 merge 방법을 사용하였다. 하지만 merge 방법에는 두 가지의 문제점이 있다. 첫째로 merge를 이용하여 나온 데이터를 checksum으로 확인 후 그 복구된 데이터를 100% 신뢰할 수 없다는 것이다. 둘째로 잘못된 데이터를 merge를 이용하여 복구 하였을 때 너무 많은 비트가 에러가 발생 되었을 경우 즉 merge 결과가 너무 많을 경우 재전송보다 못한 효율이 발생될 수 있다.

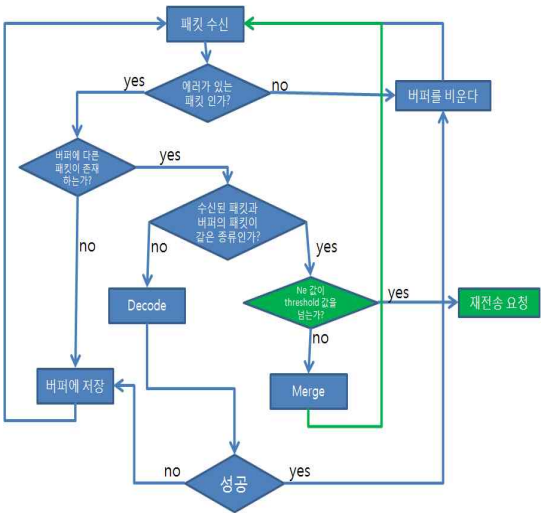

그림 4. 제안하는 모델의 순서도

그림 4와 같이 제안하는 모델은 SPaC 모델과 다른 점은 다음과 같다. 첫째로 오류가 있는 수신된 패킷과 버퍼의 패킷을 merge 하여 계산 된 패킷을 올바른 패킷으로 인식 하지 않는다. 다시 패킷 수신으로 상태로 돌아가서 패킷을 수신한다. 그 이후 그 패킷에 해당하는 패리티 패킷을 수신하였을 때 decoding 거친 패킷을 신뢰한다. 이러한 이유는 merge 결과로 나온 패킷을 신뢰 할 수 있는 패킷으로 가정 후 패리티 패킷을 수신함으로써 decoding을 거쳐 신뢰할 수 있는 패킷으로 검증하는 것 이다. 또 다른 다른 점은 무조건 merge 를 하는 것이 아니라 오류가 있는 패킷 A 와 또 다른 오류가 있는 패킷 B를 exclusive or를 수행한다.(A xor B) exclusive or를 수행한 결과 값의 1의 값을 Ne로 정의 하면 2Ne - 2는 merge 결과 값의 개수가 된다. 즉 Ne 값이 일정 이상 크면 재전송 비용보다 merge 비용이 커질 수 있으므로 적절한 Ne 값 즉 threshold 값을 찾아 Ne 값에 따른 재전송 및 merge

를 선택적으로 사용해야 한다.

이와 같이 제안하는 방법에서는 기존의 SPaC 방법을 바탕으로 재 전송률을 최대한 줄여 전송에 따른 에너지 효율을 높이는 가운데 전송 된 패킷의 신뢰성을 높이고 threshold 값을 이용하여 패킷의 복구에 사용되는 overhead를 효율적으로 줄이고자 하였다.

4. 실험결과

실험 방법은 C 프로그램을 이용하였다. 사용된 툴은 gcc(version 4)를 이용하였고 입력 데이터는 표 1을 사용하였다. 표 1은 수행시간을 나타내는데 decode의 수행시간은 2155클럭(L=29bytes)이 필요하다. 단순 merge 를 이용한 데이터 복구 방법은 172클럭(Ne=2), 1563클럭(Ne=2), 9305클럭(Ne=2)이 필요하다. 패킷 크기가 29bytes 를 재전송하기 위해서는 86200클럭이 필요하다.[19] 이것을 바탕으로 실험을 위해 29bytes 크기의 패킷을 임의로 생성하여 임의의 잘못된 패킷을 만들었다. 임의의 잘못된 패킷의 에러 비트의 개수는 2의 배수로 설정 하였다.

	CPU Cycles		Equivalent transmitted bits	
Function	L=29 bytes	128	29	128
Encode/Invert	436	1673	1.1	4.3
Decode	2155	9234	5.6	24.1
Diff	1228	4396	3.2	11.4
Merge ($n_e = 2$)	172	172	0.5	0.5
Merge ($n_e = 4$)	1563	1563	4.1	4.1
Merge ($n_e = 6$)	9305	9305	24.2	24.2

표 1. SPaC에서의 Worst-case CPU 오버헤드[5]

우선 Ne 에 따른 수행시간에 따른 실험 결과는 그림 5와 같다. 실험 결과에서 알 수 있듯이 Ne 값이 8일 때까지 재전송보다 좋은 성능을 보여주고 있다. 하지만 Ne 값이 9이상부터는 재전송 비용보다 좋은 성능을 보여주지 못한다.

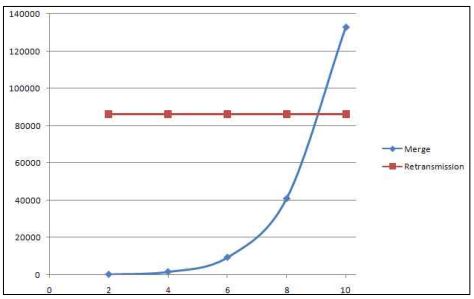

그림 5. Ne 값의 변화에 따른 수행시간
x축 Ne 변화량, y축 수행시간(cpu 클럭)

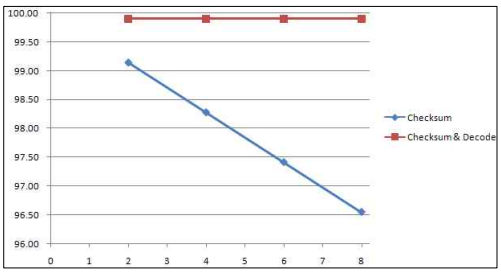

그림 6. Ne 값 변화에 따른 데이터 신뢰도
x축 Ne 값, y축 데이터 신뢰율(%)

그림 6. SPaC에서 사용된 checksum 방법을 사용한 데이터 신뢰도와 checksum 및 decode 방법을 쓴 데이터 신뢰도이다. 결과에서와 같이 decode를 같이 사용한 방법이 단순한 checksum 만 사용한 방법보다 데이터 신뢰성이 높았다. 그리고 Checksum & Decode 방식은 decode 신뢰성과 일치한다.

5. 결론

본 논문에서는 광범위 Community Computing 환경에서 Community group 의 수명을 늘이기 위하여 기존 SPaC의 에너지 효율성을 유지하면서 데이터 신뢰성을 높이기 위한 방법을 제안하였다. 먼저, 센서 노드에서의 에너지 효율의 중요성을 소개하였으며 기존 방법인 SPaC의 장점 및 단점을 소개하였다. SPaC의 단점인 특정 순간 에너지 효율이 떨어지는 것을 방지하기 위하여 threshold 값을 정하였다. 즉 SPaC의 에너지 효율성을 위하여 무조건적인 merge 방법이 아닌 잘못된 패킷의 잘못된 비트 개수에 따른 선택적 merge 방법을 이용하였다. 그리고 SPaC의 데이터 신뢰도를 높이기 위하여 merge만 이용하여 데이터를 복구하는 것이 아니라 decode까지 이용하여 데이터 신뢰도를 높였다.

본 제안된 방법을 실험한 결과 SPaC의 Ne 값의 threshold값이 8인 것을 찾아냈으며 단순한 merge를 이용하여 checksum 방법보다 decode를 같이 이용한 방법의 신뢰율이 1~3.5% 향상 된 것을 보였고 찾아낸 threshold값을 이용하여 cpu의 수행시간을 줄여 전체적인 에너지 효율을 이루었음을 알 수 있다.

참고문헌

[1] S. Lin and D. J. Costello. Error Control Coding,

Second Edition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[2] R. E. Blahut. Theory and Practice of Error Control Coding. Addison-Wesley, 1983

[2] S. Lin and D. J. Costello. Error Control Coding, Second Edition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[3] P. Sindhu. Retransmission error control with memory. IEEE Transactions on Communications, 1977.

[4] Henri Dubois-Ferriere, Deborah Estrin and Martin Vetteri. Packet Combining in Sensor Networks. SenSys'05, 2005

[5] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, 2004.