R-CAT : 노드능력을 고려한 내구적 멀티캐스트 트리 생성기법

김은석 장지용° 박성용
서강대학교 컴퓨터학과
wannajump@dcclab.sogang.ac.kr, {jiyong97°, parksy}@sogang.ac.kr

R-CAT : Resilient Capacity-Aware Multicast Tree Construction Scheme

Eunseok Kim, Jiyong Jang°, Sungyong Park
Dept. of Computer Science, Sogang University

요 약

스트리밍 서비스는 인터넷 트래픽의 많은 부분을 차지할 정도로 인기 있는 서비스가 되었고, 확장성을 위해 P2P기반의 스트리밍 서비스가 제안되었다. P2P기반 스트리밍 환경은 빈번한 피어들의 떠남과 합류가 일어난다. 이러한 멀티캐스트 그룹의 변화에 대처하기 위해서 다중 멀티캐스트 트리가 제안되었다. 이는 중복성을 통해 멀티캐스트 그룹의 변화에 따른 영향을 줄였다. 하지만 노드의 능력 차이를 고려하지 않았기 때문에 트리가 길어지고, 불안정해질 수 있다. 이를 위해 본 논문은 노드의 능력을 고려한 내구적 멀티캐스트 트리 생성 기법(R-CAT)을 제시하여 우수 노드를 트리의 상층부에 위치시킴으로써 트리의 길이를 줄이고 트리 상층부의 안정화 문제를 해결할 수 있다. 또한 제시한 기법의 유효성을 증명하기 위해 기존의 SplitStream을 확장해서 R-CAT을 구현, 비교 검증한다.

1. 서 론

일반적으로 피어-투-피어 기반 스트리밍 시스템은 멀티캐스트 구조를 생성하고 유지하는데 사용된 알고리즘이 분산되어 있느냐, 혹은 그렇지 않느냐에 따라서 분산 알고리즘(distributed algorithm)기반[5][7]과 중앙집중식 알고리즘(centralized algorithm)기반[9]으로 분류될 수 있다. 기존의 분산알고리즘 기반 스트리밍 시스템은 중계에 참여하는 노드들의 능력을 모두 동등하다 가정하고 그 차이를 고려하지 않는다. 노드별 능력 차이를 고려하지 않고 멀티캐스트 트리(multicast tree)를 생성한다면 멀티캐스트 트리의 상부계층에 능력이 떨어지는 노드가 위치할 수 있다. 이는 트리를 불균형적으로 만들고 길이가 길어지게 만드는 원인을 제공한다.

본 논문에서는 스트리밍에 참여하는 피어들의 전달능력(forwarding capacity)의 차이를 고려해서 내구적 능력고려 멀티캐스트 트리(R-CAT)를 구성하는 기법을 제시하고자 한다. R-CAT은 각 피어들이 트리에 합류할 때 피어의 능력치와 노드의 상황을 고려하여 멀티캐스트 트리의 상층부에 능력이 좋은 노드들을 배치함으로써 내구적 트리를 생성할 수 있도록 해준다.

본 논문의 나머지 구성은 다음과 같다. 2장에서는 기존의 문제점을 해결하기 위해 제안된 노드의 능력을 고려한 내구적 멀티캐스트 트리 생성 기법에 대해 알아본다. 3장에서는 기존의 방법과 지연시간 및 받지 못한 패킷(lost packet) 개수 등의 측면에서 비교하여 성능 향상을 검증한다. 마지막으로 4장에서 논문의 결론과 향후[1] 과제에 대해 논의 한다

2. 내구적 능력고려 멀티캐스트 트리 생성 알고리즘

R-CAT은 노드의 능력을 고려한 멀티캐스트 트리를 생성하기 위해 다음의 세 가지 알고리즘으로 구성 된다 : 지역정보 갱신 알고리즘, 등반을 통한 부모 선택 알고리즘, 적합성을 고려한 희생자 선택 알고리즘. 다음은 각 알고리즘에 대한 상세한 설명이다.

2.1 지역정보 갱신 알고리즘

지역정보 갱신 알고리즘은 한 노드가 자신과 연결된 주변 노드들의 현재 상태를 인지하고, 그에 따라 정책 결정을 내리는 밑바탕이 된다.

주변상황 정보를 실시간으로 파악하기 위해서는 지역정보 갱신 알고리즘을 매우 자주 사용해야 한다. 이는 노드의 합류와 떠남이 빈번한 동적인 상황에서 주변상황에 대한 정확한 정보를 제공해주지만, 그에 따른 추가적인 오버헤드가 발생하게 된다. 이 추가적인 오버헤드의 발생을 회소화하기 위해서 주기적인 제어 메시지에 정보를 피기-배킹(piggy-backing)하여 노드의 정보를 교환하도록 한다. 메시지의 교환은 새로운 노드가 합류요청을 하는 상황(node join)과 부모노드와 자식노드들 간의 생사확인 메시지를 교환(heartbeat message)하는 상황에 이루어진다.

노드는 지역정보 갱신 알고리즘을 통해 노드의 능력치(capacity), 생존성(aliveness), 적합성(goodness), 자신을 루트노드로 하는 부분트리(sub-tree)의 총 자손의 수(the number of sub-children), 이렇게 네 가지 정보를 교환한다. 각각의 정의는 다음과 같다.

c_i: 노드 i의 최대 능력치

h_i: 노드 i의 직계 자손(direct children) 집합

$A_i = 1 - \frac{1}{c_i + k}$: 노드 i의 생존성(k : scaling factor) <식 1>

$S_i = \sum_{j \in h_i} S_j + 1$: 노드 i를 루트노드로 하는 부분트리의 총 노드 수 <식 2>

* 사사표시는 현재 논의 중이며, 관리규정(협약 체결, 국고지원금 교부시 제시 예정)(안)에는
(국문 표기) "이 논문은 2006년도 두뇌한국21사업에 의하여 지원되었음."
(영문 표기) "This work was supported by the Brain Korea 21 Project in 2006." 로 되어있습니다.

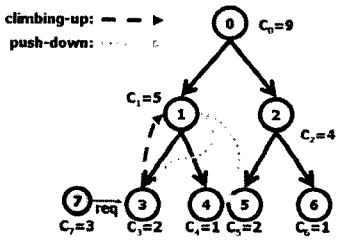
$G_i = A_i \times S_i$: 노드 i의 적합성 <식 3>

노드 i의 생존성 A_i는 능력치가 큰 노드가 안정적이라는 일반적인 연구결과를 이용해서 노드 i가 병목현상 등을 발생시키지 않고 정상적으로 동작할 수 있는 수치를 나타낸다. 여기서 스케일링 인자(scaling factor) k는 노드의 능력치 c_i를 얼마나 생존성에 반영시킬지를 결정하는 상수이다.

노드의 적합성 G_i는 노드 i의 생존성과 부분트리 후손을 모두 고려한다. 노드의 적합성은 생존성이 높거나 부분트리 후손이 많을수록 높아진다. 이 인자는 부모노드의 자식노드 선택 알고리즘에 사용된다.

2.2 등반(Climbing)을 통한 부모 선택 알고리즘

본 논문이 제시한 알고리즘은 트리의 안정성을 높이기 위해서 가장 안정적인 노드를 트리의 최상층에 배치한다. 노드의 능력치에 따라 등반(climbing)과 푸시다운(push-down)이 반복적으로 이루어짐으로써 보다 좋은 능력치를 지닌 노드가 트리의 상부계층에 위치하는 것을 돕는다.



[그림 1] 등반과 푸시-다운

[그림 1]은 등반을 통한 부모 선택 알고리즘의 동작원리의 한 예를 보여준다. 새로운 노드 7이 부모 후보노드 3에게 합류요청 메시지를 보낸다고 가정하자. 이 경우 노드 3은 자신의 능력치와 노드 7의 능력치를 비교하는데, 노드 7의 능력치가 노드 3의 능력치보다 크므로 노드 3은 합류요청 메시지를 자신의 부모인 노드 1에게 전달한다(climbing). 노드 1은 노드 7보다 큰 능력치를 가지고 있기 때문에 노드 7을 수용하려 한다. 여기서 노드 7이 여분의 자식 노드를 수용할 수 있다면 자신의 자식으로 등록을 하고, 그렇지 않다면 합류 요청 메시지를 자신의 자식 노드인 3번과 4번 노드에게 전달하게 된다(push-down). 합류 요청 메시지는 드롭(drop)된 노드가 받아들여질 때까지 푸시-다운된다.

2.2 적합성을 고려한 희생자 선택 알고리즘

부모 노드는 새로운 노드가 합류요청을 했을 때, 합류요청 메시지를 등반시키지 않는다고 결정했다면 자신의 여분의 수용능력을 확인한다. 부모가 여분의 수용능력이 없을 경우, 새로운 자식노드와 기존의 자식 집합의 합집합에서 방출할 희생자(victim)를 선택해야 한다. 희생자 선택 알고리즘은 자식노드의 적합성(goodness)과 지연시간(delay between parent and a child)을 고려한다. 내구성(resilience)은 <식 4>로 표현될 수 있으며, 그룹변화 상황에서 확률적으로 패킷 손실을 줄일 수 있는 능력으로 정의한다. 희생자 선택 알고리즘은 내구성이 가장 최대화 할 수 있는 방향으로 희생자를 선택한다.

부모 노드는 자식노드의 적합성과 부모노드와 자식노드간의 지연시간을 고려해서, 적합성은 최대화시키고 지연시간은 최소화시키는 자식노드 집합 h_i^*(optimal children set)를 선택한다. 안정적인 노드와 부분트리의 자식노드의 개수가 많은 것을 먼저 고려하는 것은 트리의 상층부에 안정적인 노드를 배치하고, 부분트리 후손이 많은 자식을 희생자에서 배제시킴으로써 트리 전체의 패킷 손실을 줄이기 위함이다. 여기서 적합성과 지연시간의 중요도를 적절히 조절하는 트레이드-오프 파라미터 (trade-off parameter) λ를 도입하여 최종적으로 다음의 식을 얻는다.

H_i : 노드 i가 선택 가능한 모든 자식 집합의 집합

$\forall h_i \in H_i,\ |h_i| \le C_i$

λ : 트레이드-오프 파라미터

$$R_i = \max_{h_i \in H_i} \left\{ \sum_{j \in h_i} (G_j - \lambda d_{ij}) \right\} \quad \text{<식 4>}$$

$$h_i^* = \arg\max \left\{ \sum_{j \in h_i, h_i \in H_i} (G_j - \lambda d_{ij}) \right\} \quad \text{<식 5>}$$

희생자 선택 알고리즘을 적용하기 이전의 자식 집합을 h_i이라 하고, 새로운 자식 노드를 j라 하면 희생자 노드는 다음의 식을 통해 구할 수 있다.

$$victim = \left\{ h_i \cup j \right\} - h_i^* \quad \text{<식 6>}$$

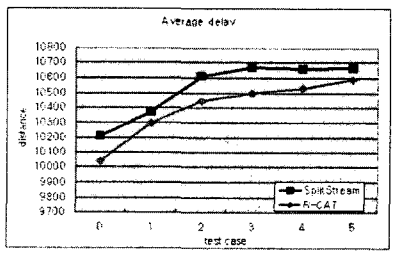
3. 성능 평가

본 논문에서는 앞서 소개한 대표적인 피어-투-피어 기반의 스트리밍 시스템인 SplitStream[7]과 R-CAT의 성능을 비교 측정한다. 성능평가는 [표 1]과 같은 노드 능력치[1]를 이용하여 실험을 했다. [그림 2]와 [그림 3]은 각각 평균 지연시간과 총 패킷 손실을 보여준다.

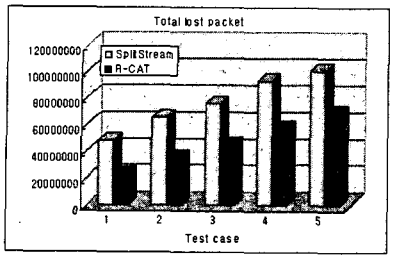
[표 1] 스트리밍 시스템 클라이언트의 노드 분포

Node Outdegree (Capacity)	Percentage (%)
0	56
1	21
2	9.5
3-19	5.6
20	7.9

[그림2]는 R-CAT이 SplitStream보다 짧은 지연시간을 유지하는 것을 보여준다. 이는 노드의 등반에 따른 지속적인 노드들 간의 성능 비교에 따라 얻어진 결과이다.



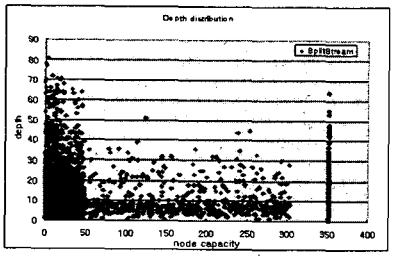
[그림 2] 평균 지연시간



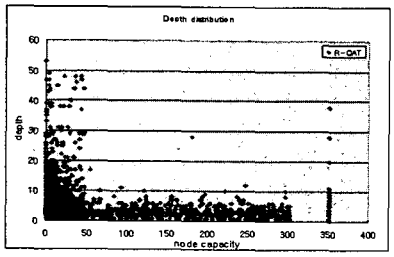
[그림 3] 총 패킷 손실

[그림3]에서는 R-CAT이 SplitStream에 비해 적은 패킷 손실률을 보여주고 있는 것을 알 수 있는데, 이는 R-CAT의 등반 알고리즘과 적합성을 고려한 희생자 선택 알고리즘의 적용으로 얻어진 결과이다. 적합성을 고려한 희생자 선택 알고리즘에서 선택된 희생자는 <식 3>의 적합성이 의미하는 바와 같이 가장 덜 안정적이거나 가장 부분트리 자식의 개수가 적은 노드이다. 이것은 패킷 손실을 낮추는 결과로 연결된다.

한편 등반 알고리즘과 적합성을 고려한 희생자 선택 알고리즘이 트리의 안정성을 높였다는 것을 [그림 4]와 [그림 5]를 통해서 확인할 수 있다. SplitStream은 노드의 능력을 고려하지 않았기 때문에 우수 노드들이 트리의 하단에 많이 위치하였다. 반면 [그림 5]에서 R-CAT은 SplitStream에 비해 우수 노드들이 트리의 상층부에 존재하고 전체적으로 트리의 길이가 짧아지는 모습을 보인다. 또한 우리는 실험 결과를 통해서 R-CAT이 그룹변화 상황에서도 지속적으로 트리의 안정성을 높인다는 것을 알 수 있었다. 시뮬레이션 기간 동안 SplitStream은 점점 트리의 깊이가 길어지는 현상을 보인 반면, R-CAT은 거의 일정수준의 트리 깊이를 유지하였다.



[그림 4] SplitStream의 노드 분포



[그림 5] R-CAT의 노드 분포

4. 결론 및 향후 과제

기존의 피어-투-피어 기반의 스트리밍 시스템은 노드의 능력 차이에 대한 고려가 없이 연구가 진행되어 왔다. 따라서 지연시간만을 고려하거나 임의의 노드를 자식, 혹은 부모 노드로 선택했다. 또한 희생자를 선택함에 있어서 그 노드의 능력, 부분트리 자식의 개수 등을 고려하지 않았다. 이는 멀티캐스트 그룹의 노드들이 합류, 떠남, 실패 등을 반복할 때, 트리의 안정성을 떨어뜨리는 결과를 낳았다.

따라서 본 논문에서는 노드의 능력 차이를 고려해서 그룹의 변화에도 안정성을 유지할 수 있는 내구적 멀티캐스트 트리 생성 알고리즘을 제시하였다. 노드의 정보 수집을 위한 오버헤드를 피기-배킹을 통해 최소화하는 알고리즘을 제시하였다. 또한 노드의 능력을 고려해 합류 시 등반을 통해 우수한 노드를 트리의 상층부에 위치시키고, 희생자 선택 시 노드의 적합성을 고려해서 내구적인 트리를 생성하는 알고리즘(Resilient Capacity-Aware multicast Tree, R-CAT)을 제시하였다.

테스트를 통해 R-CAT의 평균 지연시간이 향상되었음을 확인하였고 패킷 손실률도 SplitStream에 비해 2배 가까이 좋아진 것을 확인할 수 있었다. 그러나 피기-배킹만을 의존해 지역정보를 수집함으로 인해 전역 최적화가 아닌 지역적 최적화를 이룬 것은 보완이 필요한 부분이다. 전역 최적화를 이루기 위해서는 효율적인 멤버십 프로토콜을 개발을 통해 지역정보를 보다 정확하고 더 넓은 범위에서 수집할 수 있어야 한다. 따라서 추후 효율적이고 확장성이 높은 멤버십 프로토콜에 대한 연구가 필요하다.

4. 참고문헌

[1] Sripanidkulchai, K., Ganjam, A., Maggs, B., and Zhang, H. 2004. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. SIGCOMM Comput. Commun. Rev. 34, 4 (Aug. 2004), 107-120.

[2] Ganjam, A.; Zhang, H. Internet Multicast Video Delivery. Proceedings of the IEEE Volume 93, Issue 1, Jan 2005 Page(s):159 - 170

[3] *Tran, D.A.; Hua, K.A.; Do, T.* ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE Volume 2, 30 March-3 April 2003 Page(s):1283 - 1292 vol.2

[4] Banerjee, S., Bhattacharjee, B., and Kommareddy, C. 2002. Scalable application layer multicast. In Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (Pittsburgh, Pennsylvania, USA, August 19 - 23, 2002). SIGCOMM '02. ACM Press, New York, NY, 205-217.

[5] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", SOSP'03, Lake Bolton, New York, October, 2003.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001

[7] H. Deshpande, M. Bawa, and H. Garcia-Molina. "Streaming Live Media over a Peer-to-Peer Network", Technical Report, Stanford University, August 2001. http://dbpubs.stanford.edu:8090/pub/2001-31

[8] Sripanidkulchai, K., Maggs, B., and Zhang, H. 2004. An analysis of live streaming workloads on the internet. In Proceedings of the 4th ACM SIGCOMM Conference on internet Measurement (Taormina, Sicily, Italy, October 25 - 27, 2004). IMC '04. ACM Press, New York, NY, 41-54

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, Distributing Streaming Media Content Using Cooperative Networking ACM NOSSDAV, Miami Beach, FL, USA May 2002