지속적인 실시간 공동 편집을 위한 프레임워크 개발

정회원 손 규 식*, 이 수 동**, 조 인 휘**°

A Framework for Continuous Real-time Collaborative Editing

Kyu-Seek Sohn*, Soodong Lee**, Inwhee Joe**° *Regular Members*

요 약

스마트폰의 보급과 소셜 네트워크 서비스의 활성화로 인해 문서 편집 분야에서도 여러 사용자가 하나의 문서를 동시에 편집하는 실시간 공동 편집에 대한 연구가 활발하게 진행되고 있다. 실시간 공동 편집에서는 문서의 변경 사항에 대한 동기화를 위해서 참가자 간의 공동 편집 세션을 유지해야 한다. 공동 편집 중에 세션이 종료된다면 연결되어 있던 참가자들은 더 이상 공동 편집을 진행할 수 없게 된다. 본 연구의 목적은 의도하지 않은 상황에서 편집 중인 세션이 종료되지 않는 지속적인 실시간 공동 편집을 위한 프레임워크를 개발하는 것이다. 본 논문에 제안된 공동 편집 프레임워크에서 공동 편집 세션의 참가자는 언제든지 편집을 종료할 수 있고, 불안정한 네트워크로 인해 특정 참가자의 연결이 끊어지는 경우에도 공동 편집 세션은 지속적으로 유지될 수 있다.

Key Words : 실시간 공동 편집, 편집기 Server 변환 방식

ABSTRACT

In the field of document editing, a real-time collaborative editing that multiple users edit a same document simultaneously has been actively studied. The co-editing session should be kept for the synchronization about the changes during the real-time collaborative editing of documents among the participants. If the session ends during collaborative editing, they wouldn't be able to continue co-editing anymore. The purpose of this study is to develop a framework for Continuous Real-time Collaborative Editing that disconnects the session in unintended situations. Participants should be able to terminate editing at any time in opened co-editing session, and certain participant is able to disconnected because of unstable networks. In this case, the editing session should be maintained without termination.

Ⅰ. 서 론

다양한 의사소통이 지원되는 스마트폰과 소셜 네트워크 서비스(SNS)의 보급이 늘어나면서 시간과 장소에 구애받지 않는 정보의 공유와 협업에 대한 중요성이 커지고 있다.

공동 문서 편집은 대표적인 협업이다. 실시간 공동 편집이란 공유 문서를 동시에 다중 사용자가 편집하면서 다른 사용자의 편집 내용이 동기화되어 각 사용자의 문서에 반영되는 것이다. 실시간 공동 편집은 다음과 같은 요구사항을 만족해야 한다.

- 응답성 ~ 어느 한 편집기의 변경 사항이 공동 문서에 반영되는 시간이 짧아야 한다.
- 동시성 ~ 공동 편집 세션의 모든 참가자는 공유 문서를 동시에 편집할 수 있어야 한다.
- 일관성 ~ 모든 편집기의 문서는 동기화되어 동일한 문서로 유지되어야 한다.
- 유연성 ~ 공동 편집 참가자는 언제든지 세션에 참

※ 본 논문은 제1저자가 한양사이버대학교 연구년 기간 중에 연구한 내용을 포함하고 있습니다.
* 한양사이버대학교 정보통신공학과 (kssohn@gmail.com)
** 한양대학교 컴퓨터공학부 이동네트워크 연구실 (soodong0.lee@gmail.com, iwjoe@hanyang.ac.kr), (° : 교신저자)
논문번호 : KICS2011-04-188, 접수일자 : 2011년 4월 15일, 최종논문접수일자 : 2011년 9월 5일

가하거나 종료할 수 있어야 한다.

- 안정성 ~ 참가자 간의 네트워크 연결이 끊어지더라도 공동 편집 세션이 유지되어야 한다.

실시간 공동 편집의 응답성과 동시성을 보장하기 위해서 각 편집기는 공동 편집 문서의 복제본(Replica)을 생성한다. 각 문서 복제본의 일관성을 유지하기 위한 동기화 방법으로 OT(Operational Transformation) 방식과[1,2] CRDT(Commutative Replicated Data Type) 방식이 연구되었다[3]. 이와 같은 동기화는 하나의 Server가 담당하기 때문에 편집기와 Server의 네트워크 연결이 끊어지거나 Server가 종료되는 경우 공동 편집 세션을 유지할 수 없게 된다. 본 논문은 네트워크 환경이나 참가자 변경에 따른 제한을 받지 않고 공동 편집 세션을 지속적으로 유지함으로써 유연성과 안정성이 보장된 실시간 공동 편집을 가능하게 하는 프레임워크를 제안하였다.

Ⅱ. 관련 연구

실시간 공동 편집 시스템은 컴퓨터 네트워크에 접속되어 각자의 문서 복제본을 동기화하는 다수의 편집기들로 구성되며 그 구조는 대개 Client-Server 모델을 따른다[4-7]. Server를 운용하는 방법에 따라 Server 지정 방식과 전용 Server 방식으로 나눈다.

2.1 편집기 Server 지정 방식

공동 편집 세션을 개시하는 편집기를 Server로 지정하고 이후 세션에 참가하는 편집기는 모두 Client로 지정하는 방식이다. 그림 1과 같이 세션의 편집기들 중에 하나의 편집기가 Server 역할을 담당하고 다른 편집기들은 Client로서 이에 연결하게 된다. 리눅스 환경에서 개발되고 있는 Gobby[8]와 같은 실시간 공동 편집기에서 사용하고 있는 방식이다.

편집 세션을 개시한 편집기가 Server로 지정되기 때문에 서브네트워크(Subnetwork)에 접속된 편집기들만으로도 하나의 완전한 공동 편집 세션을 형성할 수 있다. 그러나 Server 역할을 하는 편집기의 호스트는 대개 개인 컴퓨터이므로 신뢰성이 낮을 뿐만 아니라 공동 편집 세션이 종료될 때까지 세션에서 이탈할 수 없으므로 유연성도 낮다.

그림 1. 편집기 Server 지정 방식 구성도

2.2 전용 Server 생성 방식

공동 편집 세션을 개시할 때 특정 호스트에 전용 Server를 동작하게하고 다른 모든 참가 편집기는 Client가 되는 방식이다. 전용 Server 방식의 시스템 구조는 그림 2와 같다. Etherpad[9]를 비롯한 웹 기반의 실시간 공동 편집기에서 많이 사용되고 있는 방식이다.

전용 Server 호스트는 오직 Server의 역할만을 하도록 성능과 안정성 면에서 우수하지만 Server가 네트워크 상의 지리적 위치가 고정되어 있으므로 Server와 Client 사이의 네트워크 연결 비용을 최적화하기 곤란하고 네트워크 연결의 신뢰성이 편집 시스템의 안정성을 결정하게 되어 최근 급격하게 보급된 이동 통신망이나 무선 인터넷을 기반으로 한 공동 편집 시스템의 안정도가 떨어지게 되는 단점이 있다.

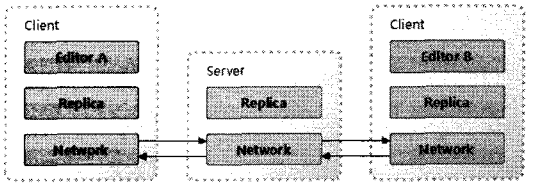


그림 2. 전용 Server 생성 방식 구성도

Ⅲ. 공동 편집 Framework 설계 및 구현

기존의 Server 기반 공통 편집 시스템은 II절에서 살펴 본 바와 같이 유연성과 안정성이 낮은 단점이 있다. III절에서 우리는 편집기 Server 지정 방식을 개선하여 공동 편집에 참여하는 편집기들 사이에서 Client와 Server의 역할이 변환될 수 있게 함으로써 공동 편집 시스템의 안정성과 유연성을 향상시키는 편집기 Server 변환(Transformation) 프레임워크를 제안하고 이를 설계하고 구현하였다.

3.1 편집기 Server 변환 방식

제안된 편집기 Server 변환 방식이 공동 편집 세션을 개시하는 편집기를 Server로 지정하고 그 후에 세션에 참가하는 편집기를 Client로 지정하는 것은 편집기 Server 지정 방식과 유사하다. 제안된 방식과 기존의 편집기 Server 지정 방식과 다른 점은 공동 편집 세션이 진행되는 도중에 편집 Server가 변경될 수 있다는 점이다. 즉 현재의 편집 Server가 여러 가지 이

유로 세션에서 이탈하면 편집 Client 중 하나가 편집 Server로 변환되어 편집 세션을 관할함으로써 편집 세션의 지속성(Seamless Continuity)을 유지한다. 이 편집기 Server 변환 방식은 두 가지 중요한 메커니즘으로 구성된다.

첫째는 편집 Server의 참여자 정보 관리 메커니즘이다. 편집 Server는 공동 편집 세션의 참여자 정보인 Active User List와 Member User List를 가진다. 이들 참여자 정보는 각 참여 노드의 IP주소, 포트 번호를 포함한다. 편집 Client는 편집 세션에 참여할 때나 탈퇴할 때 편집 Server에 자신의 정보를 제공한다. 편집 Server는 주기적으로 모든 편집 Client들에게 Ping Message를 전송함으로써 임의의 편집 Client가 공동 편집 세션에서 돌발적으로 탈퇴하는 것을 감지할 수 있다.

둘째는 예비 편집 Server 메커니즘이다. 본 연구에서 우리는 공동 편집 세션에 참가한 편집 Client들 중 예상 잔여 작업 시간이 가장 긴 편집 Client를 예비 편집 Server로 지정하는 전략을 사용하였다. 편집 Server와 예비 편집 Server는 공동 편집 문서와 세션에 관련된 일체의 정보를 공유(Mirroring)한다. 편집 Server는 세션에서 이탈하고자 할 때 예비 편집 Server에게 이를 알려서 예비 편집 Server에게 자신의 역할을 원만하게 인계한다. 한편 편집 Server와 예비 편집 Server는 서로 Ping Message를 주기적으로 교환하여 동작 상태를 서로 감시하여 편집 Server가 공동 편집 세션에서 돌발적으로 이탈하면 예비 편집 Server가 이를 감지하여 자신이 편집 Server로 변환하고 자신 안에 저장된 공동 편집 문서 및 세션 정보를 이용하여 공동 편집 세션에 참가한 모든 Client들의 네트워크 연결과 공동 편집 Replica를 재 동기화한다.

그림 3은 세 개의 편집기로 구성된 공동 편집 네트워크의 예를 보여 준 것이다. 편집기 A가 공동 편집 세션을 시작함으로써 편집 Server가 되었고 편집기 B가 세션에 먼저 참가하였고 그 다음에 편집기 C가 세션에 참가했는데 편집 Server는 먼저 세션에 참가한 Client 편집기 B를 예비 편집 Server로 지정하였다고 가정한다.

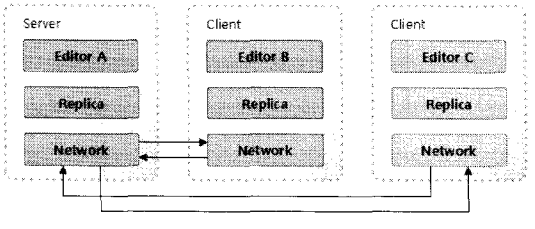


그림 3. 편집기 Server 변환 전의 세션 구성도

편집 Server인 편집기 A가 공동 편집 세션에서 이탈하면 예비 Server인 편집기 B가 편집 Server로 변환된다. 그림 4는 편집기 A가 공동 편집 네트워크에서 끊어졌지만 편집기 B가 편집 Server의 역할을 담당하고 편집기 C가 편집 Client가 되어 여전히 공동 편집 세션을 유지하는 예를 보인 것이다.

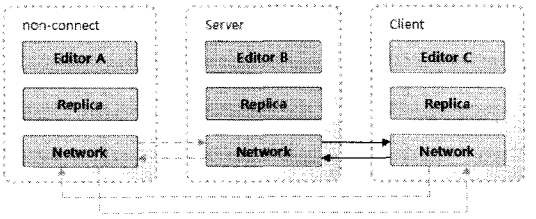


그림 4. 편집기 Server 변환에 의한 세션 유지

3.2 계층 구조

제안된 편집 Server 변환 방식을 이용한 실시간 공동 편집 시스템은 Editor, Replica, 그리고 Network의 3 개의 계층으로 구성되고 변경 사항을 편집기 간에 전달하고 동기화를 위해 User와 Connection 정보를 관리한다. User는 Active User와 Member User로 구별한다. Active User에는 세션에 참가한 사용자에 대한 정보를 저장하고 Member User에는 공동 편집 업무에 참가할 자격이 있는 모든 사용자들에 대한 정보를 저장한다. Member User는 시스템 관리자가 작성하고 관리하는데 본 논문에서는 이들 기능에 대해 자세하게 다루지 않았다. 공동 편집 시스템의 각 계층의 기능과 이들이 다루는 정보의 구조는 그림 5와 같다.

Editor는 사용자의 문서 편집기로서 입력, 수정, 삭제와 같은 기본적인 편집 기능을 구현하고 다른 사용자와 실시간 대화할 수 있는 수단을 제공하고 사용자

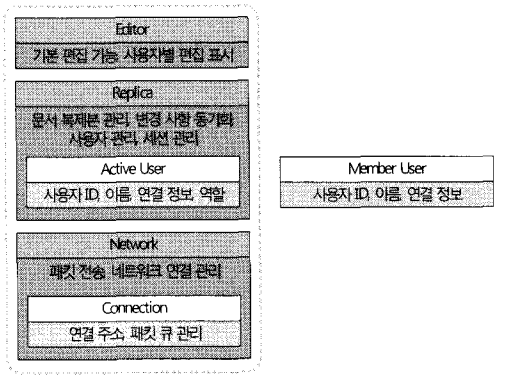


그림 5. 공동 편집 시스템의 계층 구조

에 따른 변경 사항을 음영색 표시와 같은 방식으로 구분하여 보여준다. Editor는 처음 실행될 때 우선 Client Replica를 생성한다. Editor가 Server 편집기의 역할을 맡아 공동 편집 세션을 개시해야 하는 경우에 Client Replica와 Server Replica를 함께 생성한다. Server 편집기는 공동 편집 세션에 새로 참여하는 Client 편집기들 중 하나를 예비 Server 편집기로 지정한다.

Replica는 공유 문서의 공동 편집을 위한 문서 복제본을 관리한다. Replica는 편집기의 역할에 따라 Client Replica와 Server Replica가 있다. Client Replica는 사용자의 문서 변경 사항을 Server 편집기로 전송하고 Server Replica에서 동기화된 결과를 자신의 문서 복제본에 반영한다. Server Replica는 문서 복제본 간의 변경 사항에 대한 일관성을 유지하기 위하여 OT[1,2] 또는 CRDT[3]와 같은 일관성 관리 알고리즘을 구현하고 모든 Client 편집기의 변경 사항을 동기화한다. Replica는 편집기의 역할에 따라 Server Network나 Client Network를 생성한다. Replica는 사용자 ID, 이름, 인증 정보, Connection 정보를 Active User와 Member User에 기록하여 관리한다.

Network는 세션에 참여한 편집기 간에 Message를 전송하기 위한 네트워크를 구현하고 Replica의 사용자 정보로부터 Connection을 생성하여 관리한다.

Connection은 연결된 편집기의 IP Address를 저장하고 Message의 대기 열을 관리하기 위한 queue를 관리한다.

3.3 세션 개시 및 참가

3.1절에서 예시한 공동 편집 네트워크에서 Editor A가 최초에 공동 편집 세션을 개최하고 그 후에 Editor B와 Editor C가 순차적으로 세션에 참여하고 Editor B가 예비 편집기 Server로 지정된다고 가정한다. 이들의 세션 개최와 참가 과정은 아래와 같다.

1. Editor A가 Client Replica와 Client Network를 생성하고 Active User List(AUL)에 자신을 등록한다.
2. Editor A의 Client Replica는 Member User List(MUL)에 등록된 모든 Editor Station에 Hello Server Message(HSM)를 발송한다.
3. 일정 시간 안에 Hello Client Message(HCM)를 수신하지 못하면 네트워크 상에 편집기 Server가 없다고 간주하고 Server Replica와 Server Network를 생성하고 Server Replica의 AUL에 자신을 등록한다.
4. Editor A의 Server Replica는 AUL을 참조하여 Editor A의 Client Replica와 Connection을 형성하고 자신과 Client Replica의 AUL을 갱신한다.
5. Editor B가 Client Replica와 Client Network를 생성하고 AUL에 자신을 등록한다.
6. Editor B의 Client Replica는 MUL에 등록된 모든 Editor Station에 HSM을 발송한다.
7. Editor A의 Server Replica가 Editor B의 HSM을 수신하였을 때 Editor B를 예비 편집기 서버로 지정하기로 결정하고 이를 자신의 AUL에 저장한다. Server Replica는 HCM으로 Editor B의 Client Replica에게 응답한다. 이때, HCM 안에는 Server Replica의 AUL이 포함되어 있다.
8. Editor B가 Editor A로부터 HCM을 수신한다. Editor B의 Client Network는 Connection 정보를 저장하고 Client Replica는 Message 안에 있는 Server Replica의 AUL 내용을 자신의 AUL에 저장한다. 이때 Client Replica는 자신이 예비 편집기 Server로 지정되었음을 인지한다.
9. Editor C가 Client Replica와 Client Network를 생성하고 AUL에 자신을 등록한다.
10. Editor C의 Client Replica는 MUL에 등록된 모든 Editor Station에 HSM을 발송한다.
11. Editor A의 Server Replica가 Editor C의 HSM을 수신하고 Editor C를 자신의 AUL과 Connection에 등록한다. 변경된 AUL을 HCM에 실어서 Editor C와 Editor B에게 발송한다.
12. Editor C와 Editor B는 Editor A로부터 HCM을 수신한다. 이때 Message 안에 있는 Server Replica의 AUL 내용을 자신의 AUL에 저장한다. 이때 Editor C의 Client Network는 A에 대한 연결 정보를 Connection에 저장한다.

표 1은 세션의 개최와 참가 과정을 통해 각 Editor Station에 Replica와 Network 관련 정보가 형성되는

표 1. 세션 개시 및 참가

Editor A		Editor B		Editor C	
User(AUL)	Connection	User(AUL)	Connection	User(AUL)	Connection
-	-	-	-	-	-
A, A" 추가					
	A-A" 설정		A" 설정		
	B 추가				
B' 추가					
		A, A", B' 추가			
					A" 설정
	C 추가				
C 추가					
		C 추가		A, A", B', C 추가	
A-A", B', C	A-A", B, C	A-A", B', C	A"	A-A", B', C	A"

(철자 **는 Server를 뜻하고 철자 *는 예비 Server를 뜻함)

과정을 표현한 것이다. 이 표의 AUL과 Connection 열에는 Server와 Client의 내용을 구별하지 않고 함께 표시하였다.

그림 6은 각 Editor Station의 Connection 정보에 따라 구성된 공동 편집 세션의 구조를 표현한 것이다. 이 그림에서 세 공동 편집 Editor는 1:N의 전형적인 Client-Server 구조를 형성했지만 각각의 AUL 정보를 이용하면 N:N(또는 P2P) 구조로도 쉽게 바꿀 수 있다.

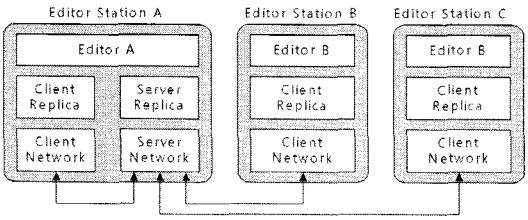


그림 6. 공동 편집 세션의 구조

3.4 서버 중퇴 및 세션 유지

이 절에서는 Server 편집기인 Editor A의 네트워크 연결이 끊어져서 Server 편집기가 공동 편집 세션에서 중퇴했을 때 편집기 Server 변환을 통해 세션이 유지되는 과정을 절차에 따라 구체적으로 설명하였다.

1. Editor A가 세션에서 중퇴한다. 이때 Editor A는 자신의 AUL과 Connection을 삭제한다.
2. 예비 Server인 Editor B의 Client Replica가 일정 시간 동안 Server Replica로부터 Ping Message를 받지 못하면 Server Replica가 세션에서 중퇴했다고 판단하고 Editor A에 대한 연결을 해제하고 자신의 AUL에서 Editor A를 삭제하고 Server Replica를 생성한다.
3. Editor B의 Server Replica는 Client Replica의 AUL을 자신의 AUL로 복사하고 Server Network를 생성한다.
4. Server Network는 Server Client의 AUL에 있는 모든 Client 편집기인 Client C에 대한 연결 정보를 Connection에 등록한다. (만일 AUL에 N 개의 Client 편집기가 있다면 이들 모두에 대한 연결 정보를 Connection에 등록한다.)
5. 새로운 Server Replica는 AUL을 HCM에 실어서 Editor C에게 송신한다.
6. Editor C는 B로부터 HCM을 받으면 A에 대한 연결을 해제하고 B로 연결을 설정하고 자신의 Connection과 AUL을 수정한다.

이렇게 하여 그림 4와 같이 Editor B가 Server로 참여하고 Editor C가 Client로 참여하는 공동 편집 세션이 형성된다. 아래의 표 2는 Editor A의 세션의 중퇴와 Editor B의 Server 변환 과정이 진행됨에 따라 각 Editor Station에 Replica와 Network 관련 정보가 형성되는 과정을 표현한 것이다. 이 표의 AUL과 Connection 열에는 Server와 Client의 내용을 구별하지 않고 함께 표시하였다.

표 2. 세션 종료 및 유지

Editor A		Editor B		Editor C	
User (AUL)	Connection	User (AUL)	Connection	User (AUL)	Connection
A,A'',B',C	A,A'',B, C	A,A'',B',C	A''	A,A'',B',C	A''
A,A'',B',C 삭제					
	B,C 삭제				
	A,A'' 삭제		A'' 삭제		
		A,A'',B' 삭제	B,B'' 설정		
		B,B'' 추가	C 추가		
					A'' 삭제
				A,A'',B' 삭제	B'' 설정
				B,B'' 추가	
-	-	B,B'',C	B,B'',C	B,B'',C	B''

(* 절차 *와 절차 **는 각각 예비 Server와 Server를 표시함)

Ⅳ. 성능 평가

Ⅳ절에서는 제안된 Server 변환 방식의 성능을 유연성과 부하영향도(Impact of load)의 측면에서 분석할 수 있는 프레임워크를 제시하였다. Server 변환 방식의 성능 분석 프레임워크를 개발하기 위하여 다음과 같은 공동 편집 작업 환경을 가정하였다.

편집 작업이 완료될 때까지 공동 편집 세션 시간 T_{CES}에는 세션을 유지하기에 충분한 편집기들(하나의 Server 편집기와 하나 이상의 Client 편집기)이 존재하고 각각의 공동 편집 작업자는 언제라도 공동 편집 세션에 참여하거나 탈퇴할 수 있다. 현재의 Server 편집기는 새로운 Client 편집기가 세션에 참여할 때마다 이를 예비 Server 편집기로 지정한다. 현재 Server 편집기와 동일한 호스트 컴퓨터에서 동작하는 Client 편집기가 세션에서 탈퇴하게 되었다면 그 Server 편집기도 함께 중퇴시키고 편집기 Server 변환 과정이 일어난다.

공동 편집 세션에서 Server 변환이 많이 일어날수록 Server 변환 방식의 유효성 또는 필요성이 크다고 할 수 있다. 즉, Server 변환 방식의 유효성을 Server 변환 발생 횟수의 기대치로 표현될 수 있다. 이를 구하기 위해 우선 Server 변환 사건에 대한 랜덤 변수의 분포를 구한다.[10] Server 변환이 임의의 시간 t 안에 발생할 확률은 오직 현재 Server 편집기의 상태를 조

건부로 하여 결정되므로 인접 Server 변환 발생 간격 시간들 $\{\widetilde{t_n}\}, n=1,2,...$은 무기억 특성(Memoryless property)을 가진 랜덤 프로세스로서 지수 분포를 가진다. 여기서 $\widetilde{t_n}$은 n번째 Server 변환과 n+1번째 Server 변환 사이의 시간이다. 즉 n번째 Server의 동작 시간이라 할 수 있다. 임의의 시점에서 보았을 때 시간 t 이내에 Server 변환이 발생할 사건의 확률 분포 $P_{ST}(t)$는 아래와 같이 정의된다.

$$P_{ST}(t) \equiv P\{\widetilde{t_{ST}} \le t\} = 1 - e^{-\overline{\lambda}t}$$

이때, $P\{E\}$는 사건 E의 발생 확률을 뜻하고 $\overline{\lambda}$는 Server 변환의 평균 발생률이다. 그림 7은 $\overline{\lambda}=0.3$ [hour-1]일 때 10 시간의 세션 시간에 대한 $P_{ST}(t)$의 그래프이다. $t=0$에서 공동 편집 세션이 시작된 후 시간이 지날수록 그 시간 안에 Server 변환 발생 확률이 높아짐을 볼 수 있다.

$\widetilde{t_{ST}}$의 pdf와 기대치는 각각 아래와 같다:

$$f_{ST}(t) = \overline{\lambda} e^{-\overline{\lambda}t}$$

$$E\{\widetilde{t_{ST}}\} = \frac{1}{\overline{\lambda}}$$

공동 편집 세션 시간 T_{CES}는 공동 편집 세션이 시작하여 편집 작업이 완료되기까지 걸린 시간이다. 공동 편집 세션 중에 발생하는 Server 변환 횟수의 기대치 $\overline{n_{ST}}$는 아래와 같이 구할 수 있다:

$$\overline{n_{ST}} = \frac{T_{CES}}{E\{\widetilde{t_{ST}}\}} = \overline{\lambda} T_{CES}$$

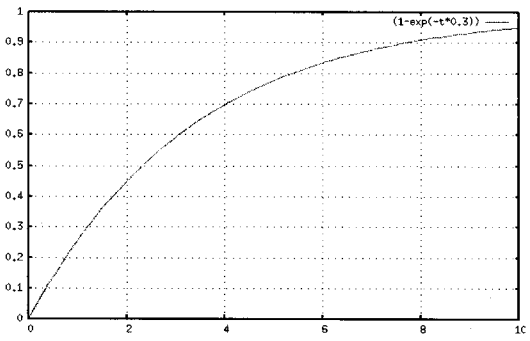


그림 7. Server 변환의 이론적 확률 분포($\overline{\lambda}=0.3$)

$\overline{\lambda}$는 Client 편집기의 평균 잔여 작업 시간 $\overline{t}_{CEW}$의 역수이다. 즉 $\overline{\lambda}=1/\overline{t}_{CEW}$이다. 그러므로 제안된 Server 변환 방식의 유효성은 각각의 공동 작업자의 작업 시간이 짧거나 세션의 지속시간이 길수록 높아진다.

이제 제안된 편집기 Server 변환 방식의 트래픽 가중도(加重度) 성능을 분석한다. 공동 편집 세션에 참가한 Client 편집기의 수를 N이라 하고 각 Client 편집기의 동기화 요구 횟수의 평균을 R이라 하고 동기화할 때마다 각 Client로 전송되는 Replica 이미지의 평균 크기를 M 바이트라 하면 공동 편집 세션에서 N Client들의 Replica 동기화 요구로 발생한 트래픽량 M_{CES}를 아래와 같이 계산한다.

$$M_{CES} = N^2 RM$$

하나의 Server 변환 과정에서는 한 번의 Replica 동기화가 일어나서 Server Replica가 N Client 편집기들로 전송된다. 공동 편집 세션에서 일어나는 Server 변환 과정에 의해 발생하는 동기화 트래픽량 M_{ST}는 아래와 같이 정의된다.

$$M_{ST} = \overline{n_{ST}} NM$$

Server 변환 방식이 유발하는 트래픽의 가중도 γ는 아래와 같이 Client 요구 동기화 트래픽량에 대한 Server 변환 트래픽량의 비율로 정의한다.

$$\gamma \equiv \frac{M_{ST}}{M_{CES}} = \frac{\overline{n}_{ST}}{NR} = \frac{\overline{\lambda} T_{CES}}{NR}$$

이때 $\overline{\lambda}$와 T_{CES}는 편집 대상 데이터의 속성, 작업자들의 작업 분담 방식, 작업자들의 세션 참여 스케줄링 방식 등에 따라 결정되므로 작업자의 수 N과 관계가 없다고 가정하면 γ는 N이나 R이 클수록 더 작은 값이 된다. 즉 작업자의 수나 Client의 동기화 요구가 많을수록 제안된 Server 변환 방식의 트래픽 부하 가중도는 낮아진다.

마지막으로 4장에서 개발한 성능 평가 프레임워크의 기초가 Server 변환 분포의 타당성을 시뮬레이션으로 확인하였다. 시뮬레이션의 조건은 N=100, $T_{CES}=10$인 세션 시간 안에서 균일 분포(Uniform distribution)로 Client 편집기의 세션 참가 사건을 발생시키고 각각의

Client 편집기마다 $\lambda = 0.3$인 지수 분포 랜덤 발생기를 이용하여 $\overline{t_{CEW}}$의 값을 생성하여 Client 편집기의 작업 시간을 지정하였다. 이를 Server 변환 알고리즘에 적용했을 때 최초의 Server 변환이 일어난 시간을 측정하는 것을 일 회의 측정 라운드로 삼는다. 이런 시뮬레이션 측정 라운드를 1000번 반복 수행하여 1 시간 간격으로 빈도를 측정한 것이다. 앞의 이론식에서는 지수 분포 랜덤 값을 $\tilde{t}_{ST}$에 적용했지만 시뮬레이션 측정 과정에서는 $\bar{t}_{CEW}$에 적용함으로써 실제 상황에 가깝게 하였다. 그림 8은 시뮬레이션을 통해 측정한 Server 변환 빈도의 그래프이다. 그림 7과 그림 8을 비교하면 그림 8도 역시 지수 분포를 가짐을 확인할 수 있고 이로써 Server 변환 방식의 성능 평가 프레임워크가 타당함을 알 수 있다.

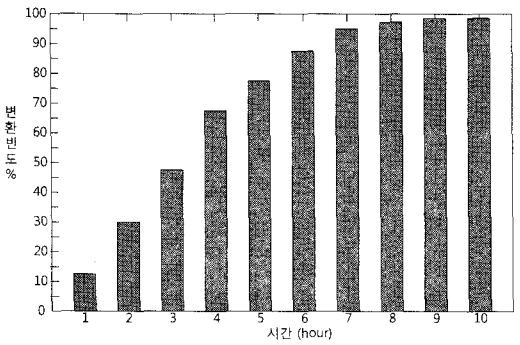


그림 8. Server 변환 빈도 측정 결과 (시뮬레이션)

V. 결론 및 향후 연구

본 논문은 실시간 공동 편집 세션을 구성하는데 있어서 Server 편집기가 종료되거나 네트워크 연결이 끊어지는 경우가 발생하더라도 Client 편집기를 Server 편집기로 변환하여 세션을 지속적으로 유지할 수 있게 하는 편집기 Server 변환 방법을 제안하였다. 아울러 Server 변환 방식의 성능을 분석하는데 필요한 이론적 프레임워크를 제시함으로써 Server 변환 발생 횟수의 기대치와 Server 변환에 의한 트래픽 가중도를 분석하여 제안된 방식의 성능을 평가할 수 있도록 하였다. 이 성능 분석 및 평가 프레임워크에 따라 시뮬레이션을 통해 Server 변환으로 세션이 유지되는 확률과 변환 발생의 기대치를 계산함으로써 제안된 편집기 Server 변환 방식이 Server가 수시로 종료되는 환경에서도 공동 편집 시스템의 세션을 지속적으로 유지할 수 있게 함으로써 유연성과 안전성을 동시에 획기적으로 개선하는 효과가 있음을 확인하였다.

향후 우리는 본 논문에서 제안한 편집기 Server 변환 프레임워크를 애드 혹(Ad-hoc) 공동 편집 시스템에 적용할 수 있도록 확장하는 연구를 수행하고자 한다.

참 고 문 헌

[1] A. Imine, "Decentralized Concurrency Control for Real-time Collaborative Editors," 2008.

[2] Du Li, And Rui Li. "An Admissibility-Based Operational Transformation Framework for Collaborative Editing Systems," Computer Supported Cooperative Work, Vol.19, No.1, pp. 1-43, Springer, 2009.

[3] Nuno Preguica, Joan Manuel Margues, Marc Shapiro Mihai Letia, "A Commutative Replicated Data Type For Cooperative Editing," in the 29th IEEE International Conference on Distributed Computing Systems, Montreal, Quebec, Canada, IEEE, June 2009, pp.395-403.

[4] Krishna P. N. Puttaswamy, Catherine C. Marshall, Venugopalan Ramasubramanian, Patrick Stuedi, Douglas B. Terry, Ted Wobber. "Docx2Go: Collaborative Editing of Fidelity Reduced Documents on Mobile Devices," in Proceeding of MobiSys'10, pp.345-356, ACM Press, 2010.

[5] Venugopalan Ramasubramanian, Thomas L. Rodeheffer, Douglas B. Terry, Meg Walraed-Sullivan, Ted Wobber, Catherine C. Marshall, AminVahdat. "Cimbiosys: A platform for content-based partial replication," in NSDI'09, Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ACM Press, 2009.

[6] Qinyi Wu, C. Pu, "Modeling and implementing collaborative editing systems with transactional techniques," 2010 6th International Conference on Collaborative Computing, pp.1-10, IEEE, 9-12 Oct. 2010.

[7] Citro, S. A Framework for Real Time Collaborative Editing in a Mobile Replicated Architecture, 2007

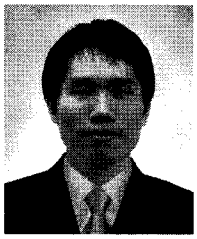
[8] A Collaborative Text Editor, Gobby. http://gobby.0x539.de/trac, 2010.

[9] Realtime Collaborative Text Editing, Etherpad. http://etherpad.org, 2010.
[10] Leonard Kleinrock, "Queueing Systems: Vol.1," pp.10-78, John Wiley & Sons, Inc., 1975.

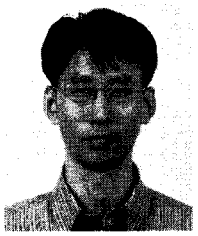
손 규 식 (Kyu-Seek Sohn) 정회원

1982년 2월 한양대학교 전자공학과 학사
1984년 2월 한양대학교 전자통신공학과 석사
2003년 8월 한국과학기술원 전자전산학과 박사
2002년 4월 LG전선(주) 광통신연구소 전임연구원
2004년 3월~현재 한양사이버대학교 정보통신공학과 조교수
<관심분야> Network Reliability, Mobile Internet, Ad-hoc Networks, Information Security

이 수 동 (Soodong Lee) 정회원



2003년 2월 한양대학교 공학대학 전자컴퓨터공학부 학사
2011년 2월 한양대학교 공학대학원 컴퓨터공학전공 석사
2003년 6월~현재 (주)한글과컴퓨터 선임연구원
<관심분야> Text Formatting, Software Architecture, Data Communication, Cloud Computing, MVC Model



조 인 휘 (Inwhee Joe) 정회원
1983년 2월 한양대학교 전자공학과 학사
1994년 12월 미국 University of Arizona, Electrical and Computer Engineering, M.S.
1998년 9월 미국 Georgia Tech, Electrical and Computer Engineering, Ph.D.
1992년 12월 (주) 데이콤 종합연구소 선임연구원
2000년 6월 미국 Oak Ridge 국립연구소 연구원
2002년 8월 미국 Bellcore Lab (Telcordia) 연구원
2002년 9월~현재 한양대학교 컴퓨터공학부 교수
<관심분야> Mobile Internet, Cellular System and PCS, Sensor Networks, Mobility Management