FAST 프로토콜 템플릿의 자동생성에 관한 연구

박연정*, 지정훈, 우균
부산대학교 컴퓨터공학과
e-mail:{roopy96, jhji, woogyun}@pusan.ac.kr

A Study on Automatic Creation of FAST Protocol Template

Youn-Jung Park*, Jeong-Hoon Ji, Gyun Woo
Dept of Computer Engineering, Pusan National University

요 약

금융 시장에서 전산거래방식이 차지하는 비중이 증가하고 그 방식이 고도화되면서 시장에서 발생하는 데이터양이 급속히 증가하고 있다. FAST 프로토콜은 대량의 정보를 압축하여 저비용 고효율로 데이터를 송수신하기 위해 제안된 텍스트 기반 프로토콜이다. 현재의 FAST 템플릿은 오랜 경험을 가진 전문가에 의해 수동으로 만들어지고 있으며, 최적화된 템플릿을 생성하기가 어렵다. 본 논문에서는 금융거래에 실제로 사용되는 기초데이터들의 패턴을 분석하여 FAST 프로토콜의 압축효율을 최대로 높일 수 있는 수 있는 템플릿 자동생성기법을 제안한다. 필드의 자연적인 속성과 패턴을 그대로 살려 신속하고 정확하게 필드 연산자를 도출, 템플릿을 완성할 수 있었다.

1. 서론

세계적으로 금융거래의 장벽이 허물어지고 대부분의 금융 거래가 과거의 공개호가방식(Open Out-Cry)에서 전산거래 방식으로 전환되면서 금융 시장에서 IT가 차지하는 비중이 날로 증가하고 있다. 특히, 고도의 투자전략을 구현하는 알고리즘 트레이딩(Algorithmic Trading)[1,2]의 확산으로 시스템에서 생성되는 시장데이터의 량은 폭발적으로 증가하였고, 이러한 대량의 데이터를 신속하게 송수신하는 데는 상당한 대역폭과 처리비용이 요구되었다. 이에 대한 해결책의 일환으로 2005년 비영리단체인 FPL(FIX Protocol Limited)에서는, FIX와 연동하면서 스트리밍 서비스에 최적화되어 설계된 고속 데이터 압축(encoding/decoding) 기술인 FAST(FIX Adapted for STreaming) 프로토콜을 제안하였다[3,4].

FAST 프로토콜이 FIX와 구별되는 것은 구조적 데이터 형식인 템플릿을 이용하여 데이터의 중복을 제거함으로써 고수준의 압축이 가능하다는 것이다. FAST 프로토콜을 이용한 데이터 압축에서는 템플릿에 포함되는 필드 연산자(Field Operator)의 적절한 선정에 따라 데이터의 압축률이 결정된다. 현재의 금융거래시스템에서의 FAST 프로토콜을 사용은 오랜 경험을 가진 전문가(expert)에 의하여 수동으로 템플릿이 만들어 지고 사용되고 있다.

본 논문은 금융거래에서 발생하는 기초데이터들의 패턴을 분석하여 FAST 프로토콜의 압축효율을 최대로 높일 수 있는 템플릿 자동생성기법에 대하여 제안한다. 실제 한국증권선물거래소 상품선물 시장에서 발생하는 기초데이터들을 분석하여 정형화된 패턴을 도출하고, 최적의 필드 연산자들을 찾아내어 FAST 프로토콜을 위한 템플릿을 자동으로 생성한다. 자동 생성된 템플릿은 기존의 금융시스템 전문가들이 수동으로 생성한 템플릿과의 비교를 통해 압축효율을 검증하였다.

본 논문의 구조는 다음과 같다. 2장에서는 관련연구로 FAST 프로토콜과 템플릿에 대하여 알아보고, 3장에서는 본 논문에서 제시하는 FAST 템플릿 자동생성기법에 대하여 알아본다. 그리고 4장에서는 실험을 통하여 본 논문에서 제시한 기법에 의해 자동 생성된 FAST 템플릿의 효율성을 알아본다. 마지막으로 5장에서는 본 논문의 결론과 향후 연구에 대한 방향을 제시한다.

2. 관련 연구

2.1 FAST 프로토콜 개발 배경 및 특징

1990년대 초 해외 선진 거래소들과 FPL(FIX Protocol Limited)[3]이 데이터 송수신 프로토콜의 표준화 노력으로 만들어낸 FIX 프로토콜은 세계적으로 널리 보급되어 표준으로 사용되어 왔다. 그러나 최근 들어 대용량의 데이터 송수신 에 문제가 발생하면서, 대용량의 데이터 송수신을 위한 새로운 프로토콜의 개발이 필요하게 되었다.

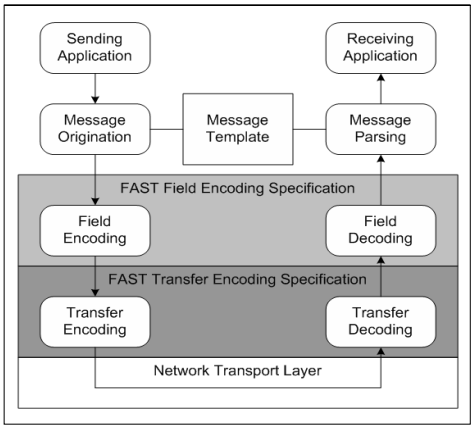
FPL은 2005년에 해외 주요 기관들(ARCA, CME, ISE, LSE, Microsoft and SGX)과 함께 FIX 프로토콜을 확장한 FAST 프로토콜을 발표했다, FAST 프로토콜은 급속도로 증가되고 있는 금융거래 시장의 데이터 처리 문제를 해결하는데 매우 효율적이며 실용적인 것으로 평가받고 있다. 이는 2005년 FPL의 주도로 진행된 PoC(Proof Of Concept)에서 입증된 바 있다.

FAST는 데이터의 길이를 획기적으로 단축시켜 데이터 송수신의 효율을 높이고, 송신자와 수신자간의 지연을 줄인다. 그리고 데이터 처리과정이 매우 단순하여 프로세싱 오버헤드가 낮고, Broadcast(Multicast, UDP) 및 point-to-point(TCP)방식으로 모두 구현 가능하다는 장점

이 있다. 또한, 여러 기관이 공동으로 개발에 참여함으로서 프로토콜의 개발 및 유지보수의 비용을 감소시키는 효과를 가져왔다.

2.2 FAST 프로토콜의 구조 및 동작과정

FAST 프로토콜은 크게 템플릿, 필드 인코딩(field encoding) 그리고 전송 인코딩(transfer encoding)의 세 부분으로 나뉜다. 템플릿은 인코딩 및 디코딩되는 데이터의 내용과 특징을 기술한 것으로 FIX의 명시적 태그의 일종으로 개발되었으며 고수준의 데이터 압축을 가능하게 한다. 필드 인코딩은 템플릿에 정의된 데이터의 속성을 이용하여 불필요한 데이터를 제거하는 과정이며, 전송 인코딩은 이진 인코딩(binary encoding)을 통하여 데이터를 축약하는 과정이다. 그림 2는 FAST 프로토콜의 구조 및 동작과정을 나타낸다.



(그림 1) FAST 프로토콜의 구조 및 동작과정

2.3 FAST 템플릿

FAST의 템플릿은 인코딩 및 디코딩 데이터의 구조를 정의한 것으로 FAST가 해석 가능한 표준 문법(syntax)을 이용하여 정의된다. 템플릿으로 표현되는 데이터의 속성은 필드명, 필드 연산자, 그리고 데이터 타입이다. FAST 템플릿은 주로 압축표기(compact notation) 방식과 XML 방식으로 정의되는데 압축표기방식은 tag number와 기존에 정의된 연산자 및 데이터 타입 심볼을 이용하여 표현하므로, XML방식에 비하여 단순한 반면 표현이 제약적이다. XML 방식은 XML에 대한 기본지식이 요구되기는 하나 보다 완전하게 템플릿을 표현할 수 있다는 장점이 있다[5].

템플릿에 표현되는 데이터의 속성 중 가장 중요한 부분은 필드 연산자이다. 메시지의 각 필드에 대하여 필드 연산자를 지정하고, 값이 중복되는 필드는 값을 제거하거나 혹은 축소하여 전송하게 되는데, 수신자는 템플릿의 내용을 알고 있으므로 해당 필드가 없거나 완전하지 않아도 정확한 값으로 디코딩하여 처리할 수 있게 된다. 이처럼, 각각의 필드에 대하여 발생하는 값의 패턴에 따라 최적화된 필드 연산자를 선택하는 경우, 삭제 또는 축소되는 필드가 많아지게 되므로 압축률을 높일 수 있다.

3. FAST 프로토콜 템플릿의 자동생성 연구

3.1 패턴 분석 기준

기초데이터를 토대로 패턴을 분석하는 경우 가장 중요한 부분은 패턴분석 기준을 결정하는 것이다. 이는 FAST의 필드연산자를 고려하여 결정하는 것이 합리적이다. 표1은 기초데이터 패턴 분석 기준을 나타낸 것이다.

<표 1> 기초데이터 패턴 분석 기준

분석 기준	고려사항
데이터 타입	string
	integer
직전 값과의 차이	string
	integer
연속적으로 동일한 값이 발생하는 빈도수와 값	모든 값이 동일한 경우
	특정 값이 연속되는 경우
	값이 바뀌면서 연속되는 경우
	값의 변화가 규칙적인 경우
	값의 변화가 불규칙적인 경우
연속적이지 않으나 특정 값이 많이 발생하는 빈도수와 값	모든 값이 동일한 경우
	연속적으로 발생하는 경우
	연속적으로 발생하지 않는 경우

3.2 필드 연산자 분석

일반적으로 필드연산자의 압축률은 Constant > Increment > Default=Copy > Delta 순으로 나타난다. 이와 함께 아래와 같은 패턴을 도출할 수 있다.

<표 2> 패턴과 필드 연산자

번호	패턴	필드 연산자
1	Na == Nt	constant
2	Na(Vd) == Nt	increment
3	(Nm > Nt) && (sum(len(Vnm)) < sum(len(Vd)))	default
4	(Nm > Nt) && (sum(len(Vd)) < sum(len(Vnt)))	delta
5	(Nt > Nm) && (sum(len(Vnt)) < sum(len(Vd)))	copy
6	(Nt > Nm) && (sum(len(Vd)) < sum(len(Vnm)))	delta

* 전체데이터 개수 : Na
* 동일 값이 연속적으로 발생하는 횟수 : Nt
* 특정 값이 나타나는 최대 횟수 : Nm
* 최대 빈도수를 갖는 값(Value) : Vm
* 최대 빈도수를 갖지 않는 값(Value) : Vnm
* 연속되어 나타나는 값 : Vt,
* 연속되지 않는 값 : Vnt
* 현재 값 - 직전 값 : Vd
* 주어진 값의 합 : sum(V)
* 주어진 값의 길이 : len(V)

3.3 템플릿 자동 생성

본 논문에서는 각 필드의 데이터 타입과 3.2 장의 공식을 기초데이터에 적용하여 도출한 필드연산자를 조합해 XML 방식의 FAST 템플릿을 자동으로 생성한다[7]. 자동으로 생성되는 템플릿의 형태는 아래와 같다. 필드연산자가 Constant 또는 Default 인 경우에는 템플릿 상에 필드값이 표시되며 각 필드가 초기 값을 가지는 경우에도 값이 존재할 수 있다.

<template name="템플릿명">

<string name="필드명"><필드연산자 value=".."/></string>
</template>

4. 실험

4.1 기초데이터를 이용한 템플릿 자동생성 실험

본 실험에서는 실험데이터로 한국증권선물거래소 상품선물 실시간 체결데이터를 사용했다. 실험데이터의 전체 거래 건수는 2510건이다. 본 실험에서는 최적화된 필드 연산자 도출을 위하여 실험데이터로부터 필요한 정보들을 추출하였다. 이와 같이 추출된 값들을 필드 연산자 패턴 분석 기법을 적용하여 최적화된 필드 연산자를 도출하였다. 표 3는 실험데이터로부터 분석한 결과이다.

<표 3> 기초데이터를 이용한 패턴 분석 실험

	Type		전송일자		전송시간	
	Value	직전값 대비	Value	직전값 대비	Value	직전값 대비
	B01		20070102		100000	
	B01		20070102		100000	
	B01		20070102		100002	2
	B01		20070102		100003	3
	...					
최대빈도값(Vm)	B01		20070102		100152	
최대빈도횟수(Nm)	2510		2510		14	
연속발생횟수(Nt)	2510		2510		732	
적용 패턴	1		1		6	
필드 연산자	Constant		Constant		Delta	
	일련번호		상품코드		종목코드	
	Value	직전값 대비	Value	직전값 대비	Value	직전값 대비
	0000001	1	06		KTB703	
	0000002	1	01	1	USD701	USD701
	0000003	1	01		USD702	2
	0000004	1	01		USD701	1
	...					
최대빈도값(Vm)	-		06		KTB703	
최대빈도횟수(Nm)	-		1511		1511	
연속발생횟수(Nt)	0		1957		1899	
적용 패턴	2		6		5	
필드 연산자	Increment		Delta		Copy	

4.2 성능평가

본 논문에서는 자동 생성된 FAST 템플릿의 성능을 평가하기 위하여 자동 생성된 템플릿과 전문가에 의해 수동으로 생성된 템플릿의 압축효율을 비교하였다. 표 4는 자동생성과 수동생성으로 생성된 템플릿의 압축효율을 비교한 결과이다.

<표 4> 기초데이터에 템플릿을 적용한 압축 실험 결과
(길이 : byte, 압축률 : %)

	원본		자동		수동	
	길이	압축률	길이	압축률	길이	압축률
최장	105	0	34	67.6	46	56.2
최단	105	0	2	98.1	2	98.1
평균	105	0	10	90.5	13	87.6

본 실험에서는 데이터의 전송헤더와 필드 인코딩 시에 추가되는 헤더(Presence Map)의 길이는 제외하고 수치를 산정하였다. 테스트에 사용된 기초데이터는 메시지 한 건의 길이가 105바이트인 아스키(ASCII) 데이터이다. FAST 프로토콜을 적용하지 않은 105바이트 데이터를 원본으로 하고, 동일한 원본데이터에 대해 자동 및 수동 방식으로 생성된 템플릿을 각각 적용하여 필드의 대한 중복을 제거하고, 데이터 타입이 정수형(integer) 필드인 경우, 이진 인코딩(binary encoding)을 적용한 후 그 길이와 압축률을 계산하였다. 자동생성 방식에서 생성된 최장 길이의 데이터는 34바이트였으며 수동방식에서는 46바이트였다. 두 가지 방식 모두 최단길이의 메시지는, 직전메시지와 거의 모든 값이 동일한 경우가 발생하여 2byte가 생성되었다. 자동생성 방식은 평균적으로 10byte의 데이터가 발생하여 90.5%의 압축률을 보여 수동생성 방식의 87.6%의 압축률보다 좋은 것으로 나타났다.

기초데이터는 기본적으로 아스키(ASCII) 형태의 데이터이므로 자동 및 수동생성 방식의 템플릿 모두에서 높은 압축률을 나타내었으며 최고 98% 이상의 압축률을 보임을 알 수 있었다. 자동방식의 경우 최장 길이의 데이터가 수동방식에 비하여 상당히 단축되었고, 결과적으로도 자동으로 생성된 템플릿의 경우가 수동의 경우보다 더 높은 압축률을 나타내었다.

5. 결론 및 향후연구

기존의 FAST 템플릿 생성 방법인, 전문가에 의한 FAST 템플릿 생성 방법은 금융거래시장의 데이터가 가지는 특수한 속성과 패턴 때문에 최적화된 템플릿 생성이 쉽지가 않았다. 본 논문에서는 금융거래에 사용되는 기초데이터를 분석하여 대용량 데이터의 전송을 위한 최적화된 템플릿 자동생성기법을 제시하였다. 기초데이터를 분석해 데이터의 정형화된 패턴을 찾아내고, 이를 근거로 FAST 템플릿의 각 필드에 적절한 연산자를 도출해 최적의 압축효율을 낼 수 있는 템플릿을 생성하였다.

향후 연구과제로는 패턴 분석 기법을 확장하여 예외적인 메시지에 대한 잘못된 필드 연산자 생성 오류를 줄여야 한다. 그리고 정형화된 패턴분석 기법과 그래픽한 사용자 인터페이스를 제공하는 FAST 템플릿 분석 및 자동생성 시스템을 구현할 것이다.

참고문헌

[1] Wikipedia, Algorithmic trading, http://en.wikipedia.org/wiki/Algorithmic_Trading

[2] 강대성, 김종호, 박주영, 박경욱, "RLS기반 Natural Actor-Critic 알고리즘을 이용한 트레이딩 전략", 한국퍼지및지능시스템학회 05 추계학술대회 학술발표 논문집, 2005, pp.238-241

[3] FIX PROTOCOL Homepage,
http://www.fixprotocol.org/
[4] FIX PROTOCOL Ltd, "FAST^{sm} Specification version 1.x.1", 2006,
http://www.fixprotocol.org/documents/3066/FAST%20Specification%201%20x%201.pdf
[5] 김성욱, 정호영, 김천식, 손기락, "XML문서저장 시스템을 위한 DOM 인터페이스의 설계 및 구현", 한국정보과학회 2000년도 봄 학술발표논문집 제27권 제1호(B), 2000.4, pp.75-77