사물인터넷 장치내 경량 자바스크립트 엔진의 힙 메모리 할당 방식 최적화 연구

박은수, 이혜민, 신동군
성균관대학교 소프트웨어대학
pes9488@skku.edu, gudbooy@skku.edu, dongkun@skku.edu

Heap Memory Allocation Technique of Lightweight JavaScript Engine for the IoT Device

Eunsoo Park, Hyemin Lee, Dongkun Shin
Sungkyunkwan University

요 약

경량 사물인터넷 장치에서 사용되는 자바스크립트 엔진은 성능보다 메모리 사용을 최소화 하기 위한 방향으로 발전하고 있다. 자바스크립트 엔진 내에서 동적 메모리를 관리하는 힙 메모리 영역은 할당 방식에 따라 동적 메모리 할당 방식과 정적 메모리 할당 방식으로 나눌 수 있는데 정적 메모리 할당 방식을 사용하면 압축 포인터를 사용하여 오브젝트 크기를 최소화할 수 있으나 메모리 낭비가 발생한다 본 논문에서는 이러한 메모리 낭비를 없애면서 압축 포인터 방식을 사용하는 동적 세그먼트 할당 방식을 제안하여 기존 정적 메모리 할당 방식에 비해 60% 메모리 사용량을 감소 시켰다.

1. 서 론

자바스크립트 프로그래밍 언어는 웹 개발 시 가장 많이 쓰이는 언어로 알려져 있다. 더불어 직관적이고 이식성이 높은 인터프리터 방식의 스크립트 언어의 특성으로 인하여 범용 애플리케이션 개발에도 많이 사용되는 언어로 손꼽힌다. 구글의 V8 엔진과 같은 기존 자바스크립트 엔진에서는 인터프리터 방식 언어의 성능 감소를 최소화하기 위한 연구가 많았다. 그러나 기존 성능 최적화된 자바스크립트 엔진의 경우 장치 내 많은 시스템 자원을 요구하는데 반해, 사물인터넷 장치는 제품 비용 및 전력 사용량을 최소화 하기 위해 제한된 메모리와 저 성능의 프로세서로 이루어져 있어 사물인터넷 장치에서는 메모리 사용량을 최소화 하는 경량 자바스크립트 엔진이 사용된다.

경량 자바스크립트 엔진은 메모리 사용량을 최소화하기 위하여 압축 포인터를 사용하고 오브젝트의 크기를 최소화하여 런타임에 동적으로 할당되는 메모리인 힙 영역에 대한 메모리 관리를 최적화 하였다. 그러나 이러한 최적화 기법들은 여러 애플리케이션이 동작할 수 있는 사물 인터넷 장치의 특성을 반영하지 못했다. 본 논문은 기존 메모리 관리 기법들을 분석하고 저성능 사물인터넷 장치를 보유한 자원에 따라 분류하여 기존 기법들의 문제점을 지적한다. 또한 분석을 토대로 한정적인 자원을 가진 사물인터넷 장치에서 사용 가능한 동적 세그먼트 힙 메모리 할당 방식을 제안하며 경량 자바스크립트 엔진 중 하나인 JerryScript에서의 구현을 통해 메모리 사용량의 감소를 검증한다.

2. 관련 기술

2.1 힙 메모리 관리 기법

자바스크립트 엔진은 런타임 시 동적으로 필요한 메모리를 할당하고 가비지 컬렉션을 통해 사용하지 않는 메모리를 해지하는데 이러한 메모리 공간을 힙 메모리 영역이라 한다. 이 힙 메모리 영역에 대한 메모리 관리 방법은 할당 방식에 따라 동적 메모리 할당 방식, 정적 메모리 할당 방식의 두 가지로 분류할 수 있다.

동적 메모리 할당 방식을 사용하는 Duktape, Espruino와 같은 자바스크립트 엔진은 오브젝트 생성 시마다 malloc과 같은 시스템 메모리 할당자를 사용하여 메모리를 할당한다. 모든 메모리 접근이 시스템 메모리 할당자로부터 할당 받은 메모리의 주소를 직접 사용하여 이루어지므로 오브젝트 접근 시 성능적 손해가 발생하지 않고 오브젝트에 비례하여 메모리 사용량이 증가한다. 그러나 시스템 메모리 할당자에 오브젝트 단위의 메모리를 동적으로 할당할 경우, 시스템 메모리 할당자에서 관리하는 메모리 영역에 대한 메타데이터 (메모리 영역 주소 및 크기)로 인한 메모리 낭비가 발생한다.

JerryScript가 사용하는 정적 메모리 할당 방식에서는 자바스크립트 엔진 인스턴스가 초기화 될 때 사용할 메모리를 미리 예약하여 자바스크립트 엔진이 직접 동적 메모리 할당을 관리한다. 이 때 연속적인 메모리 공간을 예약하므로 이 영역에 대한 오프셋을 메모리 주소로 사용할 수 있다. 이를 압축 포인터 방식[1]이라 하는데 기존 32-bit 시스템 메모리 할당자가 제공하는 4 바이트 메모리 주소를 2 바이트로 표현할 수 있다. 따라서 오브젝트가 다른 오브젝트를 참조할 때 오브젝트 내에 저장해야 할 메모리 공간이 절약되어 동적 할당 방식에 비해 오브젝트 크기를 최소화 하여 메모리 사용이 적다. 32-bit 머신 기준으로 Duktape, Espruino, JerryScript 세 가지 자바스크립트 엔진에서 각 오브젝트는 각각 26~32 바이트 (설정 값에 따라 변경), 16 바이트,

이 논문은 2016년 정부(미래창조과학부)의 재원으로 (재)스마트IT융합시스템 연구단(글로벌프론티어사업)의 지원을 받아 수행된 연구임 ((재)스마트IT융합시스템 연구단-2011-0031863)

그리고 8 바이트를 사용하는데, 정적 메모리 할당을 사용하는 JerryScript의 오브젝트 크기가 가장 작은 것을 확인할 수 있다.

2.2 사물인터넷 장치 분류

경량 자바스크립트 엔진이 동작하는 저성능 사물인터넷 장치의 경우, 온도 센서, 누수 센서, 도착 센서, 스마트 콘센트와 같이 단순 센싱 및 동작 작업만을 하는 초경량 장치와 여러 센서와 작동기를 기반으로 다양한 로직을 직접 계산하는 스마트 장치[2]로 나눌 수 있다.

초경량 장치의 경우 매우 제한된 컴퓨팅 파워와 전력, 메모리 자원을 가지므로 단순하고 메모리 사용량이 적은 애플리케이션이 주로 동작한다. 또한, 고정된 펌웨어를 통해 초기에 설정된 애플리케이션만이 장치가 해체되기 전까지 정적으로 동작한다. 그러므로 애플리케이션 내에서 사용되는 메모리 양이 고정적이고 예측 가능하다. 따라서 초경량 장치 내에서 동작하는 자바스크립트 엔진의 경우에는 다양한 종류의 초경량 장치의 메모리 제한에 따라 최대 메모리 사용량을 설정할 수 있어야 하며 한 번 설정된 메모리 사용량을 동적으로 변경할 필요가 없다.

스마트 장치는 여전히 저성능 장치이지만 초경량 장치에 비해 비교적 자유로운 메모리 제한을 갖는다. 여러 센서와 작동기가 설치되어 있고 애플리케이션의 설치 및 실행이 자유롭기 때문에 다양한 애플리케이션이 동시에 동작할 수 있다. 이 때, 각 애플리케이션의 메모리 사용량은 다른 애플리케이션의 동작에 영향을 미친다. 따라서 스마트 장치에서 동작하는 자바스크립트 엔진은 각 애플리케이션의 요구 메모리에 따라 동적으로 메모리 사용량을 조절할 수 있어야 한다.

2.3 기존 연구의 문제점

경량 자바스크립트 엔진인 JerryScript에서는 메모리를 절약하기 위해 압축 포인터를 지원하는 정적 메모리 할당 방식을 사용하는데, 이 방식은 여러 애플리케이션이 동작하는 스마트 장치 환경에 적합하지 않다. 각 애플리케이션이 실행되면 프로세스는 자바스크립트 엔진 인스턴스를 생성한다. 이 때, 정적 메모리 할당을 사용하면 애플리케이션의 메모리 사용량과 상관 없이 메모리 공간을 예약하여 할당 받는다. 이 예약된 공간 중 런타임 시에 사용하지 않는 메모리 공간은 오히려 메모리 낭비를 초래한다.

ECMA 표준 외의 기능들을 지원하기 위해 사물인터넷 플랫폼과 자바스크립트 엔진을 같이 사용할 경우[3], 자바스크립트 엔진에서 낭비되는 메모리로 인하여 사물인터넷 플랫폼이 메모리를 확보할 수 없는 상황이 발생한다. 이 문제는 비단 스마트 장치뿐만 아니라 애플리케이션 설치 환경이 정적인 초경량 장치에서도 동일하게 발생하여 자바스크립트 엔진의 메모리 낭비로 인해 사물인터넷 플랫폼이 동적 메모리를 할당 받을 수 없어 런타임 시에 치명적인 오류를 발생시킬 수 있다.

그림 1은 정적 힙 메모리 할당 방식의 JerryScript를 사용하는 IoT.js 플랫폼을 사용하여 IoT.js에서 제공하는 HTTP, TCP, File System 모듈에 대한 테스트 코드를 수행한 결과이다. 그림에서 보여지듯이 자바스크립트 엔진인 JerryScript의 메모리 사용량이 적음에도 불구하고 256KB의 메모리를 할당하므로 자바스크립트 엔진이 할당 받은 메모리 공간의 최대 90%를 낭비하는 것을 확인할 수 있다.

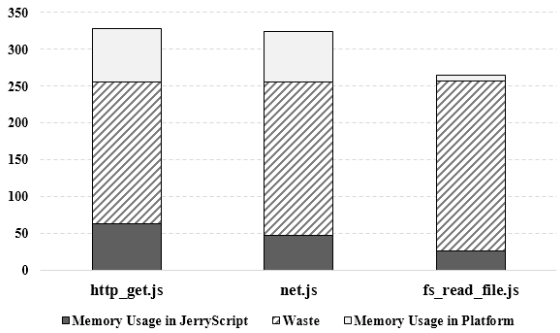

그림 1 IoT.js 메모리 사용량

3. 동적 세그먼트 할당 방식

앞서 설명한대로 동적 힙 메모리 할당 방식과 정적 힙 메모리 할당 방식에는 예약하는 메모리 크기와 오브젝트 크기에 서로 트레이드 오프가 존재한다. 본 설계에서는 동적으로 메모리를 할당 받으면서 오브젝트에 대한 압축 포인터를 사용할 수 있는 그림 2와 같은 동적 세그먼트 할당 방식을 제안한다. 동적 세그먼트 할당 방식에서 힙 메모리 할당은 오브젝트 할당과 세그먼트 할당으로 구성된다. 오브젝트 할당은 자바스크립트 힙 메모리 할당자에 의해 수행되며 요청한 크기의 오브젝트를 할당할 수 있는 세그먼트로부터 오브젝트를 할당한다. 만약 오브젝트를 할당할 수 있는 세그먼트가 존재하지 않으면, 시스템 메모리 할당자에 의해 세그먼트를 할당 받아 세그먼트 관리 테이블에 세그먼트 정보를 기록하고 오브젝트 할당을 재시도 한다.

오브젝트가 할당되면 자바스크립트 힙 메모리 할당자는 오브젝트의 첫 시스템 메모리 주소를 압축하여 압축 주소로 반환한다. 기존 압축 주소는 예약된 메모리 영역의 오프셋으로 표현했던 반면에 본 기법에서는 각 세그먼트가 시스템 메모리 할당자로부터 무작위의 주소를 할당 받으므로 새로운 압축 주소 표현 방식이 필요하다. 이를 위해 세그먼트 관리 테이블은 세그먼트의 오프셋을 기록하고 있으며 세그먼트 내의 오프셋과 더불어 압축 주소를 표현할 수 있다. 16 비트의 압축 주소를 사용하면 {세그먼트 오프셋:4,

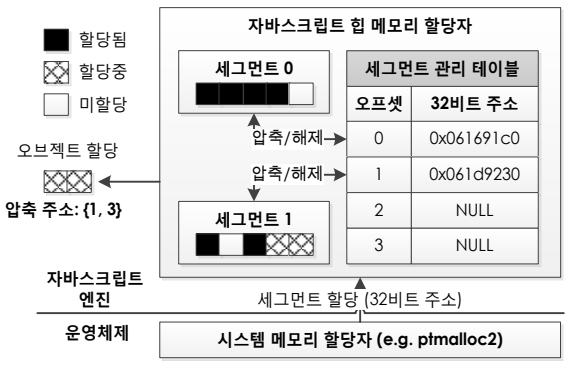

그림 2 동적 세그먼트 할당 방식

세그먼트 내 오브젝트 오프셋:12} 페어로 256KB의 자바스크립트 힙 메모리 영역을 16KB 세그먼트로 관리할 수 있고 최소 오브젝트 할당 단위는 4 바이트가 된다. 본 기법을 통해 압축 주소를 사용하면서 메모리 낭비를 최소화할 수 있다. 반면, 기존과 달리 32비트 오브젝트 주소를 압축할 때, 세그먼트 오프셋을 알아내기 위하여 세그먼트 관리 테이블을 탐색해야 하는데 이 때 탐색 오버헤드가 발생한다. 본 논문에서는 구현하지 않았으나 세그먼트 주소 캐시 자료구조를 추가하여 최근에 압축/해제가 발생한 세그먼트를 캐싱하거나 세그먼트 관리 테이블의 구조를 트리 형태로 변경함으로써 최적화할 수 있다.

4. 실 험

4.1 실험 환경

본 기법은 경량 자바스크립트 엔진인 JerryScript의 힙 영역을 관리하는 부분을 수정하여 구현되었다. 또한 사물인터넷 애플리케이션을 동작시키기 위하여 JerryScript를 사용하는 IoT.js를 일부 수정하였고 JerryScript의 기존 정적 힙 메모리 예약 크기는 256KB 설정 값을 사용하였다. JerryScript 내에서 수정한 부분은 힙 메모리 할당 및 해지, 힙 메모리 주소 압축 및 압축 해제, 그리고 메모리 프로파일링을 위해 힙 메모리 영역에서 메모리를 할당 하거나 해지하는 요청을 로깅하는 프로파일링 코드 부분이다.

실험에 사용한 애플리케이션은 표 1과 같이 Amazon 사의 Echo 제품을 모방하여 구현하였다. 실험에는 Raspberry-pi2 2개가 사용되었다. 첫 번째 장치에는 Echo와 4개의 Mic 애플리케이션을 동작하였고 두 번째 장치에서 2개의 Light 애플리케이션을 동작하였다.

표 1 애플리케이션 정보

애플리케이션 (수량)	애플리케이션 정보
Echo (1) **200 LOC**	같은 장치내에 4개의 마이크의 정보를 100ms 간격으로 수집하여 방 안의 사람의 위치를 파악하고 해당 부분에 있는 전구를 동작 시킨다. 100번 반복하여 수행한다.
Mic (4) **30 LOC**	각 마이크는 TCP 서버를 생성하여 요청을 받고 마이크 센싱 값을 응답으로 전송한다.
Light (2) **30 LOC**	Echo와 다른 장치에서 TCP 서버를 생성하여 요청을 받고 전구를 제어한다.

4.2 메모리 사용량 분석

그림 3은 가장 복잡하게 동적 메모리를 사용하는 Echo 애플리케이션의 런타임 시 메모리 사용량을 프로파일링한 그래프이다. 가장 위 선 그래프부터 기존 JerryScript, 16KB 세그먼트 할당기법, JerryScript내 실제 메모리 사용량을 의미한다. 동적 세그먼트 할당 기법을 사용하면 16KB 씩 계단식으로 낭비가 없도록 메모리를 할당하는 것을 관찰할 수 있으며 기존 정적 메모리 할당 방식에 비해 약 40% 메모리만을 사용하는 것을 확인하였다. IoT.js의 메모리 사용량을 포함한 총 메모리 사용량 부분에서는 각 애플리케이션 평균 42%의 메모리를 사용한 것을 확인하였다.

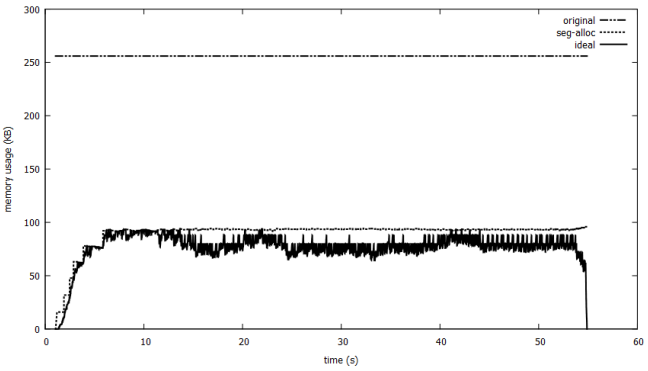

그림 3 Echo 런타임 메모리 사용량

4.3 성능 분석

그림 4는 세그먼트 크기 별 Echo 애플리케이션의 실행 시간을 측정한 그래프이다. 세그먼트 할당 방식을 사용할 경우, 앞에서 언급한 압축 연산 오버헤드가 발생하여 성능이 최대 29% 감소하는 것을 확인할 수 있으나 세그먼트 크기가 커질수록 할당받는 세그먼트 수가 줄어들어 오버헤드가 21%까지 감소하는 것을 확인하였다. 사물인터넷 환경에서는 고성능의 워크로드가 동작하지 않으므로 성능에 대한 이득보다 메모리 사용량을 줄이는 것이 더욱 중요하다. 그러므로 사물인터넷 환경에서 메모리 사용량 감소를 감안하면 무시할만한 성능 감소라 할 수 있다.

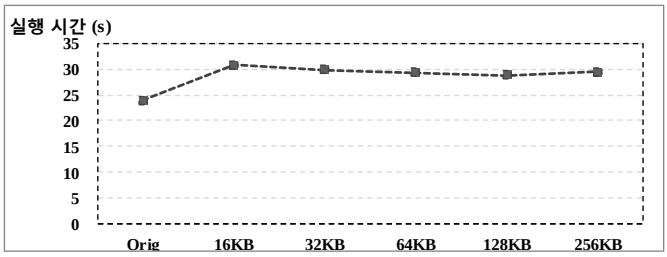

그림 4 Echo 애플리케이션 실행 시간

5. 결 론

본 연구에서는 사물인터넷에서 사용되는 경량 자바스크립트 엔진을 사물인터넷 장치에서의 사용성 측면에서 분석하고 저성능 사물인터넷 장치에 적합한 자바스크립트 엔진의 힙 메모리 할당 방식을 제안하였다. 제안한 기법은 오픈소스인 JerryScript에 구현하여 상용 애플리케이션을 모방한 자바스크립트 애플리케이션 위에서 동작하여 효과를 검증하였다. 그 결과, 기존 자바스크립트 엔진에 비해 평균 42%의 메모리만을 사용하도록 메모리 최적화를 하였다.

6. 참고 문헌

[1] Gavrin, Evgeny, et al. "Ultra lightweight JavaScript engine for internet of things." Proc of the SPLASH. ACM, 2015.

[2] 이혜민, 신동군. "사물인터넷을 위한 개방형 소프트웨어 플랫폼 설계." 한국정보과학회 학술발표논문집 1492-1494, 2015.

[3] Dongig Sin, Dongkun Shin. "Performance and Resource Analysis on the JavaScript Runtime for IoT Devices." ICCSA. Springer. 2016.