GIAO THỨC TRUYỀN THÔNG COAP TRONG MẠNG CẢM BIẾN KHÔNG DÂY

Phạm Thành Nam*, Đào Mạnh Tuấn, Nguyễn Thị Thảo
Trường Đại học Công nghệ Thông tin và Truyền thông - ĐH Thái Nguyên

TÓM TẮT

Việc áp dụng công nghệ IPv6 trong mạng vô tuyến cá nhân tiêu thụ năng lượng thấp trong đó có mạng cảm biến không dây đã tạo điều kiện thuận lợi cho tiến trình tích hợp các mạng này vào mạng Internet. Song song với việc sử dụng IPv6 trong các mạng này, giao thức CoAP (Constrained Application Protocol) cũng được phát triển để hỗ trợ cho việc truy cập các tài nguyên dữ liệu của các mạng này bằng cách sử dụng các tính năng của công nghệ RESTful Web và từ đó giúp cho việc tích hợp các mạng này vào Web.

Bài báo của chúng tôi tập trung vào công nghệ RESTful WSNs. Trình bày về giao thức CoAP, làm nổi bật lên những điểm khác biệt chính giữa giao thức CoAP và HTTP đồng thời đưa ra các kết quả của một triển khai thực nghiệm đơn giản chỉ ra những ưu điểm của giao thức CoAP trong các điều kiện về mặt năng lượng so sánh với giao thức HTTP. Trong bài báo chúng tôi triển khai giao thức CoAP cùng với công nghệ 6LowPAN trong hệ điều hành Contiki 2.6 với các nút mạng cảm biến không dây Tmote Sky.

Từ khóa: *Mạng cảm biến không dây, Web applications, Web of Things, REST, CoAP*

GIỚI THIỆU

Những tiến bộ gần đây trong việc phát triển các ứng dụng trong lĩnh vực mạng cảm biến không dây (WSN) cùng với việc sử dụng công nghệ IP trong việc xác định địa chỉ nguồn tài nguyên mạng cảm biến đã tạo ra những thay đổi lớn trong mạng Internet. Hàng tỷ các thiết bị thông minh sẽ được kết nối vào mạng Internet dưới một dạng chung gọi là công nghệ Internet of Things (IoT). Công nghệ IoT sẽ cho phép kết nối giữa môi trường vật lý với môi trường số (mạng Internet), tạo ra các khả năng cùng các thách thức thú vị cho các miền ứng dụng đa dạng, chẳng hạn như: công-tơ đo năng lượng thông minh, giám sát sức khỏe thông minh, nhà tự động hóa...

Việc sử dụng công nghệ gán địa chỉ IP trên các thiết bị điện tử nhúng gần đây được đề xuất bởi liên minh IPSO (IP cho các đối tượng thông minh), liên minh các nhà viễn thông – công nghệ thông tin và các nhà phát triển trong lĩnh vực mạng cảm biến không dây. Cũng trong thời điểm này, IETF đã tiến hành chuẩn hóa giao thức IPv6 dành cho các mạng không dây cá nhân có mất mát và tiêu thụ năng lượng thấp (LLNs) trong đó có mạng cảm biến không dây, chuẩn này được gọi là 6LoWPAN [9]. Chuẩn mới này cho phép sử dụng địa chỉ IPv6 cho các mạng năng lượng thấp và các mạng có hiệu năng hạn chế, các mạng vô tuyến này theo chuẩn IEEE 802.15.4 [11]. Ngoài chuẩn 6LoWPAN, nhóm làm việc IETF cũng tiến hành chuẩn hóa một giao thức định tuyến mới cho các mạng có mất mát và tiêu thụ năng lượng thấp này gọi là giao thức RPL [10].

Một trong những ưu điểm chính của giao thức IP cho các mạng không dây tiêu thụ năng lượng thấp đó là cho phép sử dụng các kiến trúc dịch vụ Web chuẩn mà không cần phải sử dụng các gateway trung gian. Do đó, các thiết bị thông minh (smart objects) trong đó có các nút mạng cảm biến không dây không chỉ được tích hợp vào mạng Internet mà còn được tích hợp vào Web. Việc tích hợp này được gọi là công nghệ Web of Things (WoT). Ưu điểm của công nghệ WoT đó là các ứng dụng của các thiết bị thông minh này được xây dựng trên kiến trúc REST [1]. Trong một kiến trúc REST, một nguồn tài nguyên là một đối tượng trừu tượng được điều khiển bởi máy chủ và được nhận dạng bởi địa chỉ URI. Các nguồn tài nguyên của các đối tượng thông minh này được trình bày dưới các định

* Email: ptnam@ictu.edu.vn

dạng tùy ý như: XML hoặc JSON. Nguồn tài nguyên này được truy cập và thao tác bởi một giao thức lớp ứng dụng dựa trên các tương tác client/server và các thủ tục request/response. Kiến trúc REST không bị phụ thuộc vào một giao thức lớp ứng dụng cụ thể nào, tuy nhiên phần lớn các kiến trúc REST hiện nay đều dựa trên giao thức HTTP. Giao thức HTTP thao tác tài nguyên dựa trên các phương thức GET, POST, PUT,...

Kiến trúc REST cho phép chạy các ứng dụng IoT và các tương tác M2M (machine-to-machine) được phát triển trên đỉnh của các dịch vụ Web, có khả năng chia sẻ và sử dụng lại được. Các nút mạng cảm biến trở thành các nguồn tài nguyên trừu tượng được nhận dạng bởi các địa chỉ URI, được trình bày với định dạng dữ liệu tùy ý và được thao tác dữ liệu với cùng phương thức như HTTP.

IETF đã xây dựng một nhóm làm việc để tiến hành chuẩn hóa một kiến trúc REST trong các môi trường mạng bị hạn chế (constrained networks) gọi là CoRE Working Group. CoRE đã đưa ra một kiến trúc hình tháp các dịch vụ Web dành cho mạng các đối tượng thiết bị điện tử thông minh. CoRE đã định nghĩa một giao thức truyền dẫn dữ liệu giữa mạng các đối tượng thiết bị điện tử thông minh và Web dựa trên kiến trúc REST gọi là CoAP. CoAP bao gồm các chức năng giống với HTTP, các chức năng này được thiết kế lại để phù hợp với môi trường mạng có công suất xử lý và tiêu thụ năng lượng thấp trong các thiết bị điện tử nhúng nhỏ như là các nút mạng cảm biến. Để giao thức này phù hợp với công nghệ IoT và các ứng dụng M2M, một số chức năng khác được thiết kế dành riêng cho giao thức.

Cùng với công nghệ 6LoWPAN ngày càng trở lên khả dụng, công nghệ WoT sẽ ngày càng đóng vai trò quan trọng trong cộng đồng nghiên cứu. Đã có nhiều nghiên cứu khác nhau và kiến trúc REST/HTTP cho mạng cảm biến không dây gần đây được phát triển. Trong nghiên cứu [2] đã đề xuất một kiến trúc REST cho phép các thiết bị trực tiếp quảng bá dữ liệu của chúng. Trong [3], tác giả đề xuất một khung làm việc REST/HTTP áp dụng cho lĩnh vực nhà tự động (home automation). Trong [4] đã đề xuất một toolkit cho phép người dùng tạo ra các dịch vụ Web cung cấp bởi một thiết bị cụ thể và tự động khai phá chúng theo giao diện REST API. Các tác giả trong [5] chỉ ra phương thức các ứng dụng khác nhau có thể được dựng lên tại đỉnh của kiến trúc RESTful WSNs. Trong nghiên cứu [6] minh họa một triển khai thực tế của của kiến trúc RESTful WSN.

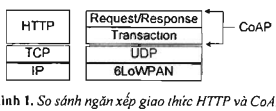
Có một đặc điểm chung là các nghiên cứu trước chỉ tập trung vào nghiên cứu kiến trúc RESTful WSN tuy nhiên chưa sử dụng giao thức CoAP. Nhóm CoRE gần đây mới bắt đầu tiến hành chuẩn hóa và xây dựng giao thức CoAP vì thế nó mới bắt đầu được xem xét đến trong các nghiên cứu về kiến trúc RESTful WSN.

Trong nghiên cứu này, chúng tôi trình bày về kiến trúc RESTful WSN dựa trên giao thức CoAP. Chúng tôi trình bày theo các phần sau: trong phần II chúng tôi giới thiệu về giao thức CoAP – lịch sử phát triển và các kiến trúc của giao thức; phần III chúng tôi trình bày triển khai của giao thức CoAP trong hệ điều hành Contiki đồng thời cũng trình bày cách triển khai giao thức HTTP trong hệ điều hành Contiki từ đó đưa ra các so sánh đánh giá; phần IV là kết luận của nghiên cứu.

GIAO THỨC TRUYỀN THÔNG COAP

Vào tháng 3 năm 2010, nhóm làm việc IETF CoRE Group đã tiến hành chuẩn hóa giao thức CoAP. CoAP là một giao thức truyền dẫn dữ liệu Web được tối ưu dành cho mạng có tài nguyên hạn chế trong các ứng dụng IoT và M2M. CoAP dựa trên kiến trúc REST trong đó các nguồn tài nguyên được cung cấp bởi các máy chủ trừu tượng và chúng được xác định bởi các địa chỉ URI. Các tài nguyên có thể được thao tác theo các phương thức giống với giao thức HTTP: GET, PUT, POST và DELETE.

Giao thức CoAP không phải mang mục đích nén HTTP. Nó bao gồm một tập con các tính năng của giao thức HTTP nhưng đã được thiết kế lại để phù hợp cho các thiết bị có công suất xử lý và năng lượng tiêu thụ thấp như là các nút mạng cảm biến. Thêm vào đó, các cơ chế khác nhau đã được điều chỉnh và một vài chức năng mới đã được thêm vào để tạo ra một giao thức phù hợp với các ứng dụng IoT và M2M. Ngăn xếp giao thức HTTP và CoAP được minh họa theo sơ đồ hình 1.



Hình 1. *So sánh ngăn xếp giao thức HTTP và CoAP*

Sự khác nhau đầu tiên giữa HTTP và CoAP chính là lớp giao vận. HTTP dựa trên giao thức điều khiển chuyển phát TCP. Cơ chế điều khiển luồng của TCP không thích hợp với các mạng bị hạn chế về mặt công suất xử lý và năng lượng tiêu thụ thấp, cùng với chi phí truyền tin lớn, thời gian trao đổi dữ liệu ngắn. Ngoài ra TCP không hỗ trợ truyền multicast và tương đối nhạy cảm với các nút di động. Giao thức CoAP sử dụng phương thức truyền UDP do đó nó có chi phí truyền tin thấp và hỗ trợ việc truyền multicast.

Giao thức CoAP được tổ chức theo hai lớp. Lớp *Transaction* xử lý các bản tin đơn được trao đổi giữa các điểm đầu cuối với nhau. Các bản tin trao đổi trên lớp này gồm bốn loại: *Confirmable* (CON) - cần có bản tin phúc đáp Ack, *Non-confirmable* (NCON) - không cần bản tin Ack, *Acknowledgment* (Ack) - bản tin dùng để phúc đáp lại quá trình truyền bản tin CON, *Reset* – để biểu thị rằng một bản tin CON đã được nhận tuy nhiên nó bị lỗi trong quá trình xử lý. Lớp *Request/Response* chịu trách nhiệm cho việc truyền phát các yêu cầu và các đáp ứng dành cho việc thao tác và truyền dẫn tài nguyên mạng. Đây là lớp được thiết kế theo kiến trúc REST. Một yêu cầu REST thường là bản tin CON hoặc NCON, trong khi một đáp ứng REST thường liên quan tới bản tin Ack.

Việc sử dụng hai lớp này cho phép giao thức CoAP cung cấp cơ chế truyền tin cậy mà không cần sử dụng giao thức TCP. Trên thực tế, một bản tin CON được truyền phát lại sử dụng một thời gian chờ mặc định và tuân theo phân bố hàm số mũ giữa các lần truyền, cho đến khi bên nhận gửi lại bản tin Ack. Ngoài ra nó còn cho phép truyền không đồng bộ, đây chính là yêu cầu chính của các ứng dụng IoT và M2M. Khi một máy chủ CoAP nhận được một yêu cầu mà không thể xử lý ngay lập tức, nó đầu tiên thừa nhận việc tiếp nhận bản tin và gửi ngược trả lại đáp ứng. Các thẻ bài được sử dụng cho việc phối hợp truyền nhận các request/response trong truyền dẫn không đồng bộ.

Một trong những mục đích của việc thiết kế giao thức CoAP là duy trì chi phí gửi tin nhỏ nhất có thể và hạn chế việc sử dụng phân mảnh. Giao thức HTTP luôn có chi phí gửi tin lớn. Điều này ngụ ý việc phân mảnh gói và suy giảm hiệu suất của các mạng LLN. Giao thức CoAP sử dụng một khối có độ dài cố định 4 byte nhị phân dùng cho header theo sau bởi các tùy chọn nhị phân. Một yêu cầu thông thường có độ dài header từ 10 – 20 bytes. Phần tiếp theo chỉ ra những ưu điểm nổi bật về chi phí truyền tin của giao thức CoAP so sánh với HTTP trong điều kiện các nút mạng sử dụng nguồn nuôi là pin.

Nguồn tài nguyên trên một máy chủ CoAP thường thay đổi liên tục theo thời gian, giao thức cho phép quan sát liên tục sự thay đổi của nguồn tài nguyên này từ một máy client. Điều này được thực hiện theo cách sau: máy client (máy quan sát) sẽ đăng kí nó với nguồn tài nguyên (đối tượng quan sát) bằng cách gửi một yêu cầu GET đến máy chủ. Máy chủ CoAP thiết lập một mối liên hệ quan sát giữa máy client và nguồn tài nguyên nó nắm giữ. Thời gian của việc duy trì mối liên hệ này được tuân theo thủ tục đăng kí [14].

Mặc dù giao thức CoAP vẫn còn đang trong quá trình phát triển nhưng đã có nhiều dạng triển khai khác nhau của nó. Hai hệ điều hành được biết đến nhiều nhất dành cho mạng cảm biến không dây đó là Contiki OS và Tiny OS đã hỗ trợ sẵn một dạng triển khai của giao

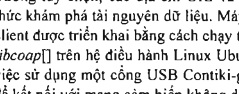
thức CoAP (chúng tôi sẽ giới thiệu chi tiết trong phần sau). Thêm vào đó, có hai thư viện mã nguồn mở được thiết kế đặc biệt dành cho mạng cảm biến không dây đó là: *licoap* được viết bằng ngôn ngữ C và *CoAPy* [20] được viết bằng ngôn ngữ Python.

ĐÁNH GIÁ SỰ TIÊU THỤ NĂNG LƯỢNG CỦA GIAO THỨC COAP

Mạng cảm biến không dây với các nút mạng thì nguồn năng lượng cung cấp cho các nút này chủ yếu là sử dụng pin. Việc sử dụng giao thức UDP làm giao thức giao vận và việc giảm kích thước tiêu đề header của gói tin truyền đã mang lại ý nghĩa quan trọng trong việc cải thiện công suất tiêu thụ và thời gian sống của pin trong mạng cảm biến không dây.

Để đánh giá hiệu năng của giao thức CoAP so sánh với giao thức HTTP, chúng tôi triển khai một thực nghiệm đơn giản. Đầu tiên chúng tôi tạo ra nhiều các yêu cầu dạng các dịch vụ Web giữa hệ thống máy client và CoAP server (chính là các nút mạng cảm biến). Tiếp sau đó là các máy client và HTTP server.

CoAP server được triển khai bởi các nút mạng cảm biến Tmote Sky chạy trên hệ điều hành Contiki với chuẩn 6LoWPAN/RPL của lớp mạng và giao thức CoAP ở lớp ứng dụng. Việc triển khai giao thức CoAP đã có sẵn trong hệ điều hành Contiki bao gồm hầu hết các đặc điểm của nó, ví dụ như: cú pháp và ngữ nghĩa các message (CON, NON-CON, ACK, RESET); các phương thức (GET, PUT, POST, DELETE...); các mã đáp ứng; các trường tùy chọn; các địa chỉ URI và phương thức khám phá tài nguyên dữ liệu. Máy CoAP client được triển khai bằng cách chạy thư viện *libcoap*[] trên hệ điều hành Linux Ubuntu với việc sử dụng một cổng USB Contiki-gateway để kết nối với mạng cảm biến không dây.



Hình 2. *Mô hình thực nghiệm đánh giá giao thức CoAP và HTTP*

Máy chủ HTTP server cũng tương tự là các nút mạng cảm biến Tmote Sky và cũng được triển khai trong hệ điều hành Contiki. Việc triển khai giao thức CoAP và HTTP được chúng tôi triển khai trên hệ điều hành Contiki 2.6 và dựa trên thư viện coap-07 internet-draft. Máy HTTP client được tạo ra bằng cách thay thế thư viện *libcoap* [18] bằng *cURL* [19], nó là một dạng cửa sổ dòng lệnh để tạo ra các chức năng của HTTP. Trong cả hai thực nghiệm này, máy client sẽ thăm dò tài nguyên tại máy server sau mỗi khoảng thời gian 10 giây trong vòng 20 phút bằng cách gửi các *request* về dữ liệu nhiệt độ và độ ẩm của nút mạng cảm biến đang nằm giữ. Khi sử dụng giao thức CoAP thì một request sẽ tuân theo định dạng sau: GET *coap://[<node_ip_address>]:<port_number>/readings*, trong đó *node_ip_address* chính là địa chỉ IPv6 của nút mạng cảm biến (thông thường sẽ dưới dạng địa chỉ IPv6 rút gọn); *port_number* là địa chỉ cổng của nút (với CoAP thông thường là 5683; *readings* chính là tài nguyên dữ liệu mà máy client gửi yêu cầu (trường hợp này là nhiệt độ và độ ẩm). Khi sử dụng giao thức HTTP thì yêu cầu của nó sẽ tuân theo định dạng sau: GET *http://[note_ip_address]:<port_number>/readings* trong đó ý nghĩa các tham số giống như trong trường hợp giao thức CoAP.

Trong cả hai trường hợp, máy server sẽ trả về các đáp ứng chính là tài nguyên của nút mạng cảm biến trong một tệp JSON (Java Script Object Notation). JSON là một đối tượng dạng các ký tự dành cho việc trao đổi dữ liệu giữa client/server. Một ví dụ về tài trọng payload của đáp ứng trả về như sau: *{"sensor":"0212:7400:0002:0202", "readings":{"hum":31, "temp":23.1}}*, trong đó nút mạng cảm biến được xác định bằng bốn nhóm cuối trong địa chỉ IPv6, *hum* là dữ liệu độ ẩm và *temp* là dữ liệu nhiệt độ.

Giao thức CoAP cũng hỗ trợ việc định dạng dữ liệu theo chuẩn khác như là XML (Extendsible Markup Language). Tuy nhiên, sự rườm rà và việc phân tích cú pháp phức tạp

của XML dẫn đến ngôn ngữ này không thích hợp dành cho các đối tượng là các nút mạng cảm biến.

Bảng 1 minh họa sự so sánh giữa giao thức HTTP và CoAP trong điều kiện về số byte được truyền trong mỗi phiên truyền, công suất tiêu thụ và thời gian sống của pin. Cả hai giao thức này đều được triển khai trên các nút mạng cảm biến Tmote Skype với nguồn nuôi là Pin.

Công suất tiêu thụ được tính toán sử dụng công cụ Energest, đây là một công cụ cho phép tính toán công suất tiêu thụ của nút cảm biến Tmote Sky[16].

Bảng 1. *So sánh giữa giao thức COAP và HTTP*

Application Protocol	Bytes per-transaction	Power	Lifetime
CoAP	154	0.744mW	151 days
HTTP	1451	1.333 mW	84 days

Các tham số đánh giá bao gồm *Bytes per-transaction* là số byte truyền trong một phiên, *power* là công suất tiêu thụ, *lifetime* là thời gian hoạt động của pin theo mỗi giao thức. Như trong bảng 1, một phiên truyền HTTP có số byte tốn gấp 10 lần so với phiên truyền CoAP. Điều này đạt được là do trong CoAP đã nén tệp tiêu đề header. Việc truyền dẫn với số byte lớn hơn này đưa ra một suy luận là giao thức HTTP sẽ sử dụng nhiều hoạt động cho việc truyền/nhận tới nút mạng cảm biến hơn và tiêu thụ công suất lớn hơn (1.33 mW trong HTTP so với 0.74 mW trong CoAP). Nhờ sử dụng lược đồ công suất tiêu thụ mà ta có thể ước tính được thời gian hoạt động của pin trong HTTP là 84 ngày ngắn hơn nhiều so với CoAP là 151 ngày.

KẾT LUẬN

Trong bài báo này chúng tôi đã trình bày về giao thức truyền thông CoAP, một giao thức lớp ứng dụng được thiết kế dành cho các mạng tiêu thụ năng lượng thấp LLNs với kiến trúc REST. Giao thức CoAP cung cấp việc thao tác nguồn tài nguyên tương tự như giao thức HTTP. Ngoài ra, giao thức CoAP còn hỗ trợ đầy đủ các chức năng thông thường của các ứng dụng IoT và M2M, như là truyền multicast, truyền thông không đồng bộ. Không giống với HTTP, CoAP sử dụng phương thức truyền dẫn UDP do đó sẽ giảm chi phí của gói tin. Bài báo này của chúng tôi cũng minh họa một so sánh đánh giá về mặt năng lượng tiêu thụ và số byte sử dụng cho việc truyền tin giữa giao thức CoAP và HTTP. So sánh của chúng tôi được tiến hành trên các nút Tmote Sky. Và được triển khai trên hệ điều hành Contiki 2.6. Kết quả của so sánh cho thấy giao thức CoAP sử dụng số byte để truyền dữ liệu ít hơn và có thời gian hoạt động của pin các nút mạng cảm biến lâu hơn so với việc sử dụng giao thức HTTP.

TÀI LIỆU THAM KHẢO

1. Roy Fielding, Architectual Styles and the Design of Network-based Software Architectures, chapter 5.
2. Dawson-Haggerty, S., et al. sMAP – a Simple Measurement and Actuation Profile for Physical Information. In Proceedings of 8th ACM Conference on Embedded.
3. Kovatsch, M., et al. Embedding Internet Technology for Home Automation. In Proceedings of IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2010.
4. Mayer, S., et al. Facilitating the Integration and Interaction of Real-World Services for the Web of Things. In Proceedings of Urban Internet of Things – Towards Programmable Real-time Cities (UrbanIOT), 2010.
5. Guinard, D., et al. A Resource Oriented Architecture for the Web of Things. In Proceedings of Internet of Things 2010 International Conference (IoT), 2010.
6. Castellani, A. P., et al. Architecture and Protocols for the Internet of Things: A Case Study. In Proceedings offFirst International Workshop on the Web of Things (WoT), 2010.
7. Shelby, Z. Embedded Web Services. IEEE Wireless Communications, pp. 52-57, December 2010.
8. Atzori, L., et al. The Internet of Things: A survey. Computer Networks, pp. 2787-2805, October 2010.
9. Kushalnagar, N, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919.

10. Vasseur, J. P., and Dunkels, A., Interconnecting Smart Objects with IP: The Next Internet. Morgan Kaufmann, 2010.
11. Shelby, Z., and Bormann, C., 6LoWPAN: The Wireless Embedded Internet, Wiley, 2009.
12. Trifa, V., et al. Design and Implementation of a Gateway for Web-based Interaction and Management of Embedded Devices. In Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE), 2009.
13. Shelby, Z., et al. Constrained Application Protocol (CoAP). Internet-Draft. draft-ietf-core-coap-07.
14. Hartke, K., et al. Observing Resources in CoAP. InternetDraft. draft-ietf-core-observe-01.
15. Eggert, L., Congestion Control for the Constrained Application Protocol (CoAP). Internet-Draft.draft-eggertcore-congestion-control-01.
16. Dunkels, A., et al. Demo abstract: Software-based sensor node energy estimation. In Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys), 2007.
17. Castellani, A., et al. Best Practice to map HTTP to COAP and viceversa. Internet-Draft. draft-castellani-core-http-coapmapping-00.txt.
18. http://libcoap.sourceforge.net/
19. http://curl.haxx.se/
20. http://coapy.sourceforge.net

SUMMARY

CONSTRAINED APPLICATION PROTOCOL IN WIRELESS SENSOR NETWORK

Pham Thanh Nam*, Dao Manh Tuan, Nguyen Thi Thao
College of Information and Communication Technology - TNU

Our paper focuses on RESTful Web technology in WSNs. Description CoAP protocol and its architecture, highlighting the main differences between the protocols HTTP and CoAP and give the results of an experiment carried out simply the advantages of CoAP protocol in terms of energy compared with the HTTP protocol. In the paper we implement CoAP protocol with 6LoWPAN technology in Contiki OS 2.6 for the Tmote Sky wireless sensor network motes
Keywords: *wireless sensor network, web application, Web of Things, REST, CoAp*

Ngày nhận bài: 07/12/2014; Ngày phản biện: 22/12/2014; Ngày duyệt đăng: 05/3/2015
Phản biện khoa học: *TS. Nguyễn Toàn Thắng – Trường Đại học Công nghệ Thông tin & Truyền thông - ĐHTN*

* *Email: pnam@ictu.edu.vn*