دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

اینترنت اشیا

گزارش پروژه ی کارشناسی

محمد مهدی امینی

استاد

دکتر منشیی

۱۳۹۶

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل اول

CoAP

۱ـ۱ معرفی

Constrained Application Protocol (CoAP) یک پرتوکل Network-Orineted برای انتقال Docu-ment است که شباهت های بسیار به HTTP دارد. هدف از طراحی این پرتوکل استفاده ی آن در دستگاه هایی با توان پردازشی پایین و منبع تغذیه ی ضعیف و کم عمر بوده است. کاربردهای این پرتوکل اساسا به اینترنت اشیا مربوط می شود که از: smart energy, smart grid, building control, intelligent lighting control, industrial control systems, asset tracking, environment monitoring می توان به عنوان مثال یاد کرد.

بخش اول

مقایسه با HTTP

به دلیل شباهت بسیار زیاد CoAP با HTTP ، مقایسه ی این دو باعث مشخص شدن ویژگی های کلیدی CoAP خواهد شد.

۱ـ۱ـ۱ لایه ی انتقال

بر خلاف HTTP که در لایه ی انتقال از TCP استفاده می کند، فقط آدرس های Unicast را در حالت Synchrounos پشتیبانی و فقط با معماری Client/Server کار میکند؛ Coap از پرتوکل UDP استفاده می کند. در نتیجه سربار کنترل Congestion و نگه داری Connection حذف می شود، امکان ارسال پیام MultiCast فراهم می شود و همچنین تبادل پیام به صورت Asynchrounos ممکن می شود. در صورتی یک کاربرد خاص نیاز به Reliability در انتقال پیام داشته باشد، در لایه ی Application مکانیزهای مناسب در نظر گرفته شده است.

۱ـ۱ـ۲ مدل ارتباط Observe

این پرتوکل با اقتباس از مدل Request در پرتوکل HTTP امکان Observe کردن یک Resource را فراهم کرده. در واقع یک Flag به اسم Observe در بسته ی Request وجود دارد که اگر مقدار دهی شود، سرور بعد از ارسال پاسخ اولیه، به ارسال پاسخ ادامه می دهد(در واقع سرور ارتباط را قطع نمی کند). این امکان باعث ایجاد مکانیزمی شبیهPush-Notification می شود که سرور می تواند تغییر یک وضعیت مثل تغییر مقدار یک سنسور را به صورت آنی به ارسال کننده ی Request اطلاع دهد.(در پرتوکل CoAP نود های سنسور به صورت سرور شناخته می شوند و با ارسال Request با متدی مثل Get مقدار سنسور را برای درخواست کننده در پاسخ ارسال می کنند. این مساله در فضای کاری HTTP با توجه به سنگین و پیچیده بودن پرتوکل و ضعیف بودن نودهای کنترل کننده ی سنسور، تقریبا غیر ممکن و در واقع غیر قابل Scale است.).

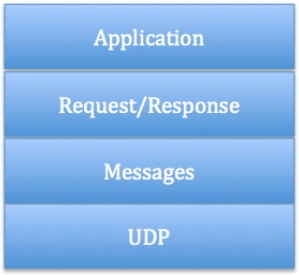
۱ـ۱ـ۳ کشف سرویس

در پرتوکل CoAP این قابلیت وجود دارد که دستگاه ها و نودهای فعال به عنوان موجودیت اینترنت اشیا، قابلیت ها و سرویس هایی که ارایه می دهند را به اطلاع دیگر موجودیت ها برسانند. به این منظور هر نود باید یک لیست از منابع و سرویس هایی را که در دسترس قرار داده به همراه توضیحات لازم (Meta-Data) تحت یک لینک (URL) استاندارد منتشر کند. دیگر نود ها با ارسال درخواست Get به این لینک، لیست را دریافت کرده و در صورت لزوم با استفاده از Meta-Data ی موجود در لیست سرویس مورد نظر را شناسایی و فراخوانی می کنند. این قابلیت به صورت پیش فرض در HTTP وجود ندارد و به برای این منظور از پرتوکل های Service Discovery مستقل استفاده می شود.

بخش دوم

پیام ها

در لایه ی بالای لایه ی انتقال ، CoAP از ۳ زیر لایه انتزعی تشکیل شده که به ترتیب از پایین به بالا Message, Request/Response, Application هستند.

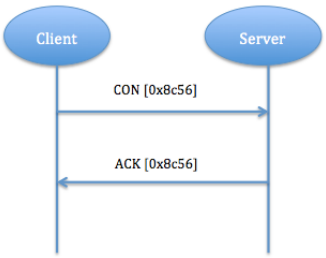


شکل ۱ـ۱: لایه های Coap به صورت انتزاعی

۱ـ۱ـ۴ لایه ی Message

لایه ی Message وظیفه ی مواجه شدن با UDP و تبادل داده به صورت Asynchronous را دارد. پیام های CoAP به واسطه ی این لایه به چهار نوع تقسیم می شوند. Con (Confirmable), NON (Non-Comfimable), ACK (Acknowlegdment), RST (Reset)

پیام هایی که به صورت Con ارسال می شوند باید توسط گیرنده با یک پیام ACK پاسخ داده شوند. اگر گیرنده نتوانست پیام را دریافت کند، یک پیام از نوع RST ارسال می کند. و به این ترتیب می توان Datagram های UDP را به صورت Reliable ارسال کرد(شکل ۱ـ۲) .

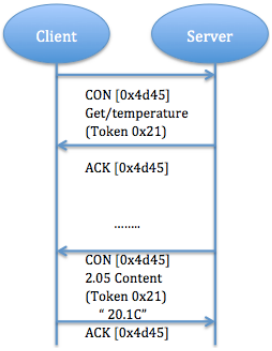


شکل ۱ـ۲: ارسال پیام به صورت Reliable

اگر پیام توسط فرستنده به صورت NON ارسال شوند هیچ پاسخی مبنی بر Acknowledge از گیرنده دریافت نمی کند. اما در صورتی که گیرنده در دریافت پیام با مشکلی مواجه شد یک پیام RST به عنوان پاسخ ارسال می کند.

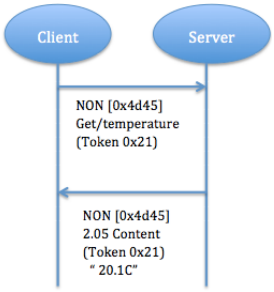
از ساختاری که ذکر شد برای پیاده سازی مدل های Request/Response استفاده شده به طوری که کلاینت می تواند Request خود را به صورت CON ارسال کند و پاسخ آن را در ACK دریافت کند. همچنین سرور می تواند پاسخ را به طور مجزا در قالب یک پیام CON ارسال کند(در این حالت ۲ پیام CON و ۲ پیام ACK

تبادل خواهد شد (شکل ۱ـ۳) .



شکل ۱ـ۳: مدل separate-request/respsone

حتی امکان ارسال request و respone در قالب NON وجود دارد که نه سرور نه کلاینت منتظر ACK نخواهند بود (شکل ۱ـ۴) .



شکل ۱ـ۴: مدل non-comfirmable-request/respsone

۱ـ۱ـ۵ ساختار پیام ها

پیام های CoAP در واقع داده های باینری هستند که بخش Data را در Datagram های UDP به خود اختصاص می دهند. طول بخش ثابت هدر پیام های CoAP ۴ بایت است. این هدر شامل یک بخش Token و یک بخش Options و یک بخش Payload همگی با طول های متغییر است.

Table 3 Message Format

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
Ver	T	OC	Code	MessageID
Token (if any, TKL bytes)…			
Options (if any)…			
Payload (if any)…			

شکل ۱ـ۵: ساختار پیام های CoAP

قسمت Code نماینگر نوع پیام(CON یا NON یا ...) است. قسمت MessageID نیز در فرایندهایی مثل ارسال ACK و RST کاربرد دارد.

بخش سوم

امنیت

با توجه به اینکه اینترنت اشیا در جنبه های حساسی از زندگی تاثیر گذار خواهد بود، نیاز به امن بودن فضای تبادل اطلاعات در این حوزه کاملا جدی است. به این منظور CoAP امنیت را از طریق بکار گیری پرتوکل DTLS فراهم کرده است. DTLS به نوعی نمونه ی همزاد SSL/TLS است که در لایه ی انتقال از پرتوکل UDP بهره می برد. به این ترتیب مناسب CoAP و محدودیت های فضای اینترنت اشیا است.DTLS تمامی ۳ نیاز اصلی در حوزه ی امنیت یعنی محرمانگی انتقال داده ها، اهراز هویت و اطمینان از صحت داده های دریافتی را، به صورت End-to-End و بدون اعمال سربار هدرهای رمزنگارانه در لایه های پایین تر شبکه، ارایه می کند.

بخش چهارم

اینترنت اشیا

پرتوکل CoAP اساسا برای استفاده در حوزه ی اینترنت اشیا طراحی شده. در طول گزارش به جنبه های مختلف طراحی این پرتوکل پرداختیم. در اینجا به اختصار لیستی از ویژگی هایی که این پرتوکل را برای اینترنت اشیا مناسب کرده ذکر می شود:

* ۱: معماری این پرتوکل بر اساس معماری RSETful بنا شده در نتیجه مقادیر سنسورها و سرویس های عملگرها میتوانند در قالب یک آدرس URL با متد های Post, Get و ... در دسترس قرار گیرند.

* ۲: امنیت End-to-End به واسطه ی استفاده از پرتوکل DTLS فراهم شده که این پرتوکل در لایه ی اپلیکیشن فعالیت می کند و در لایه ی انتقال از UDP بهره می برد. در نتیجه با کمترین سربار محاسباتی و کمترین سربار شبکه امنیت مورد نیاز فراهم می شود.

* ۳: تبادل پیام به صورت Asynchrounos (غیر همزمان) امکان پذیر است در نتیجه CPU های دستگاهایی که در طرفین ارتباط قرار دارند Cycle ها خود و انرژی منبع تغذیه را صرف انتظار کشیدن برای دریافت پاسخ نمی کنند. هر زمان که پاسخ به بدست نود رسید، CPU به واسطه ی یک CallBack نرم افزاری از در دسترس قرار گرفتن آن آگاه شده و پردازش های لازم را روی آن انجام می دهد.

* ۴: اندازه کم هدر و کم بودن پیچیدگی Pars کردن آن باعث مصرف کمتر Cycle های Cpu و مصرف کمتر Ram خواهد شد.

* ۵: پشتبانی از URI و Content-type و Accept که باعث می شود نوع داده های مختلف به راحتی به واسطه ی این پرتوکل قابل انتقال باشند.

* ۶: توانایی های کش کردن و پروکسی کردن باعث می شود دستگاه های مبتنی بر CoAP بتوانند به راحتی با سیستم های مبتنی بر HTTP صحبت کنند.

* ۷: امکان به کاری گیری Reliability در لایه ی اپلیکیشن در صورت نیاز.

* ۸: پشتیبانی از آدرس های Unicast و .Multicast

* ۹: کم بودن حجم پیام ها و تعداد آنها پرتوکل را برای محیط هایی که هزینه ی پهنای باند بالایی دارند،به گزینه ی مناسبی تبدیل کرد است.

کتاب‌نامه

[1] Washer, Peter. *Learning Internet of Things*. Packt, 2015.

[2] CoAP. http://coap.technology.

[3] Cliff Brake, IOT Protocols: MQTT vs CoAP vs HTTP. http://bec-systems.com/site/1200/iot-protocols-mqtt-vs-coap-vs-http, 2014.

[4] Xi Chen, Constrained Application Protocol for Internet of Things. http://www.cs.wustl.edu/ jain/cse574-14/ftp/coap/index.html, 2014.

[5] eclipse.org, MQTT and CoAP, IoT Protocols. https://eclipse.org.