به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

مبانی اینترنت اشیا

گزارش بخش‌های تئوری تمرین سری چهارم

نام و نام خانوادگی	محمدعلی کشت پرور
شماره دانشجویی	۹۷۳۴۰۲۲

فهرست گزارش سوالات

سوال ۱ -سوال اول ۳
سوال ۲ -سوال دوم ۵
سوال ۳ – سوال سوم ۷
سوال ۴ – سوال چهارم ۸

سوال ۱ – سوال اول

معیار	MQTT	HTTP	CoAP
معماری	Publish/subscriber	Client/server	Client/server – clienr/borker
لایه انتقال	TCP	TCP	UDP
Human readable	خیر	بله	خیر
اندازه هدر	۲ بایت	هدرهای زیادی دارد و ثابت نیست	۴ بایت
اندازه پیام	کوچک	بزرگ	کوچک
نوع شبکه	برای شبکه‌های محدود شده طراحی شده است که در آن محدودیت منابع داریم مانندIoT	برای شبکه اینترنت که در آن محدودیت‌های منابع نداریم یا کمتر داریم ساخته شده است.	برای شبکه‌های محدود طراحی شده است.
Single point of failure	این مشکل را دارد چون یک بروکر مرکزی برای کنترل دارد و با آسیب دیدن آن شبکه از کار می‌افتد.	نمی باشد	در مدل کلاینت سروری آن این مشکل را نداریم.
نوع ارتباط	آسنکرون	ارتباطش به صورت سنکرون است. یعنی کلاینت باید صبر کند تا سرور جوابش را بدهد. همین موضوع برای کاربردهای IoT آن را غیر مناسب می‌کند.	مدل آسنکرون است. چون از UDP استفاده می‌کند. یعنی کلاینت نیازی ندارد تا منتظر سرور بماند و این موضوع در مصرف بهینه انرژی کمک زیادی می‌کند.

نوع ارتباط	یک طرفه یعنی کلاینت باید شروع کننده ارتباط باشد و وقتی سرور چیزی برای کلاینت داشته باشد باید صبر کند تا کلاینت درخواستی به او بدهد.	ارتباط یک‌طرفه است.	ارتباطش یک‌طرفه نیست.
امکان همه پخشی	دارد	ندارد	دارد
تاخیر	تأخیر کمتری در ارسال دارد.	تأخیر زیادی در ارسال به دلیل حجم زیاد و ارتباط Reliable دارد.	تأخیر کمتری در ارسال دارد.
عملکرد در شبکه ها LLN	ضعیف	ضعیف	بسیار خوب
مکانیزم امنیتی	SSL/TLS	از روش‌های امنیتی SSL/TLSاستفاده می‌کند.	DTLS

MQTT و CoAP هر دو به عنوان پروتکل های IoT کاربردی هستند .MQTT در مواردی که داده‌های جریانی یا داده‌های مبتنی بر رویداد داریم ؛ مفید است . MQTT و CoAP برای برنامه‌های کم مصرف و محدود استفاده می‌شود، بنابراین آن را به گزینه‌ای مورد علاقه برای بسیاری از کاربردهای IoT تبدیل می‌کند. اما تفاوت های اساسی دارند. در حالی که پروتکل http برای کاربردهای IoT مناسب نمی باشد در واقع HTTP یک پروتکل ارتباطی مبتنی بر TCP/IP است که برای ارائه داده ها (فایل های HTML، فایل های تصویری، نتایج پرس و جو و غیره) در شبکه جهانی وب استفاده می شود.

CoAP در جایی کارآمدتر است که منابعی با پارامترهای متعدد وجود داشته باشد و client فقط به اطلاعات خاصی نیاز دارد. در اینجا ما می‌توانیم از پارامترهای query به همان روشی که از آنها در درخواست HTTP استفاده می‌شود ، استفاده کنیم. همچنین به دلیل استفاده از پروتکل UDP برای شبکه های LLN عملکرد خیلی خوبی دارد. پروتکل COAP به اندازه MQTT قابلیت اطمینان ندارد زیرا MQTT از TCP استفاده می کند که دارای مکانیزم های data retransmission و timeout و ... می باشد در حالی که COAP از

UDP استفاده می کند و acknowledgment برای هر بسته ندارد اما میتوانیم با ست کردن فلگی در هدر بسته COAP آن را برای این پروتکل فعال کنیم تا مانند TCP بعد از ارسال هر بسته ack را نیز ارسال کند.

سوال ۲ – سوال دوم

CoAP

از آنجایی که پروتکل CoAP از پروتکل UDP در لایه انتقال خود استفاده می کند ؛ پس قابلیت اطمینان پایینی دارد . این مورد را می توان به عنوان بزرگترین مشکل CoAP معرفی کرد . پروتکل UDP دریافت دیتاگرام ها در مقصد را تضمین نمی کند ، توانایی کنترل ازدحام ندارد همچنین هیچ تضمینی نمی کند که داده ها با همان ترتیب ارسالی در مقصد دریافت شوند . همه این موارد و به طور کلی قابلیت اطمینان در پروتکل TCP وجود دارد ولی پروتکل UDP این موارد را تضمین نمی کند. پس می توان نتیجه گرفت پروتکل CoAP نیز این مشکل را داراست . البته مدیریت این موارد نیاز به زمان و توان پردازشی بیشتری می باشد.

راه حل : از آنجایی که خیلی از کاربردهای IoT نیازی به خیلی از امکانات لایه TCP ندارد ، می توانیم بعضی از آنها مثل ack را لایه ی بالایی لایه انتقال یعنی لایه اپلیکیشن ایجاد کنیم . در پروتکل CoAP نیز در لایه اپلیکیشن برای تضمین دریافت در مقصد استفاده می شود . برای این منظور در CoAP چهار نوع پیام مختلف تعریف شده است. این دسته بندی صورت‌های Confirmable و Non-Confirmable و Acknowledgement و Reset می باشد. با این روش می توانیم مشکلات CoAP را تا حدودی برطرف کنیم . این مکانیزم در لایه اپلیکیشن اتفاق می افتد و زمانی که پیام از نوع Confirmable باشند دریافت بسته در مقصد، ACK باید ارسال شود. به این ترتیب در صورت که ACK دریافت نشود مجدد بسته ارسال می شود. علاوه بر این می توانیم duplicate در پیام‌ها را هم به کمک message ID تشخیص دهیم. در واقع مشکلات این پروتکل را می توان تا حدودی در لایه اپلیکشین مدیریت کرد ولی در صورتی که بخواهیم تمام ویژگی های TCP در لایه اپلیکیشن داشته باشیم ؛ بار محاسباتی بیشتری اضافه می کند و پیچیدگی های زیادی به لایه اپلیکیشن اضافه می شود.

از مشکلات دیگر CoAP میتوان به IP spoofing اشاره کرد. بدین صورت که شخص حمله کنند آدرس IP خود را با آدرس IP مقصد جاگذاری می کند و خود را به جای آن معرفی می کند.

راه حل : استفاده از TLS است اما بار محاسباتی اضافه می کند.

MQTT

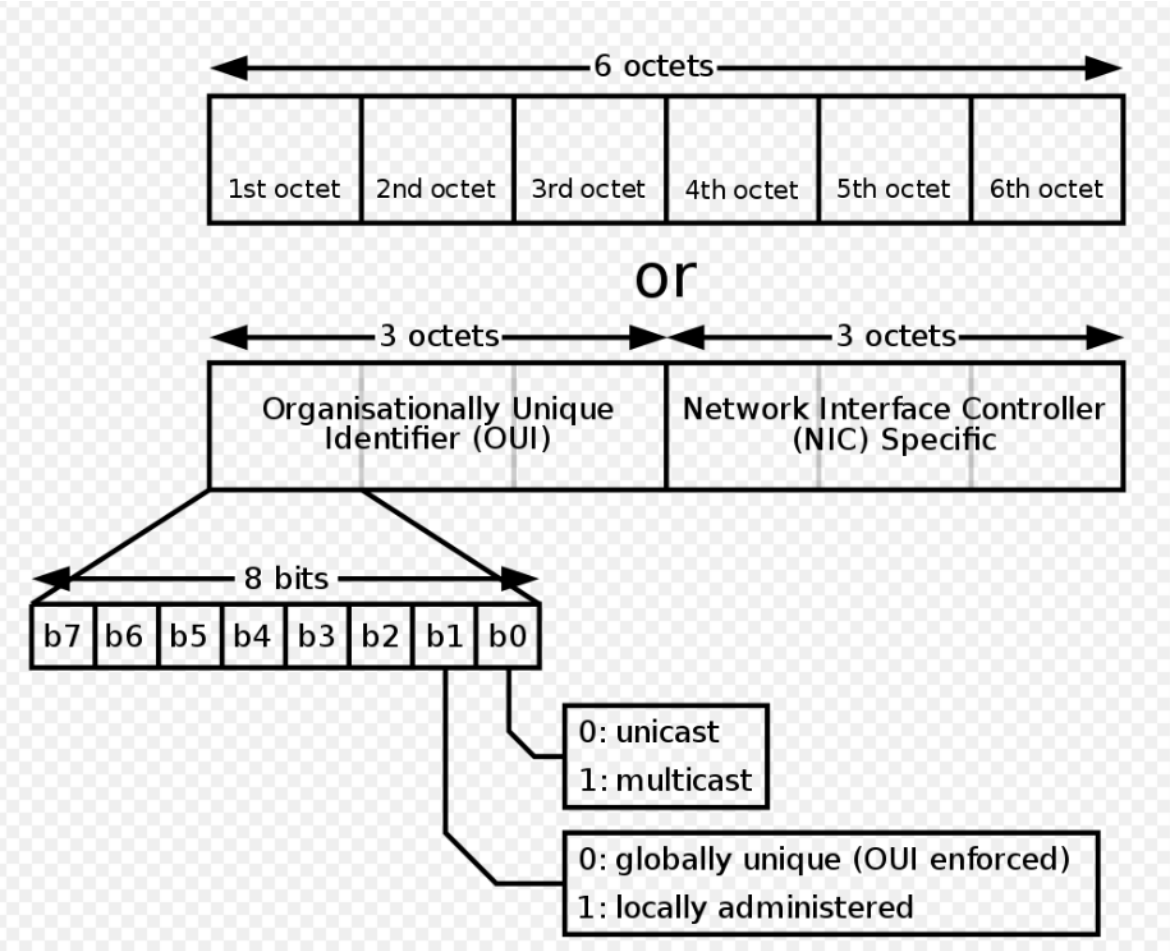
از بزرگترین مشکل این پروتکل می توان به استفاده از پروتکل TCP در لایه انتقال اشاره کرد . این پروتکل با توجه به این که قابلیت اطمینان بالایی دارد و رسیدن و ترتیب پیام ها در مقصد را تضمین می کند . علاوه بر این مکانیزم های کنترل خطا و ازدحام و ... دارد که باعث ایجاد بار محاسباتی زیادی می شود . TCP از پروتکل hand shaking استفاده می کند و قبل از ارسال داده باید کانکشن ایجاد شود (connection-oriented) . این موضوع بر مصرف باتری تأثیر می گذارد. علاوه بر این، دستگاه‌های متصل به TCP تمایل دارند سوکت‌ها را برای یکدیگر باز نگه دارند که نیاز به حافظه و توان پردازشی بیشتری دارد .

راه حل : برای حل این مشکل از پروتکل سبک تری مثل UDP در لایه انتقال استفاده می شود و بعضی از ویژگی های TCP را در لایه اپلیکشن ایجاد کنیم .

از مشکل دیگر MQTT میتوان به single point of failure بودن آن اشاره کرد. زیرا این پروتکل از مدل pub-sub برای تبادل پیام استفاده میکند و دو قسمت broker و client دارد. همه کلاینت ها داده ارسالی خود را در بروکر قرار می دهند و داده مورد نیاز خود را نیز از بروکر دریافت می کنند . بنابراین broker می تواند bottleneck ما باشد. برای رفع این مشکل میتوانیم به جای یک broker از چندین broker استفاده بکنیم که درصورتی که یکی از broker ها از کار بیفتد بقیه بتوانند کار آن را انجام دهند و سیستم تبادل پیام از کار نیفتد.

سوال ۳ – سوال سوم

برای شناسایی دستگاه در اکوسیستم IoT می توانیم از mac address استفاده کنیم . از mac address در سوییچ های شبکه نیز می توانیم برای شناسایی دستگاهای متصل استفاده کنیم . به طور مثال پروتکل ARP در لایه دیتالینک از mac address برای پیدا کردن دستگاه مورد نظر استفاده می کند. برای ارتباطات گره به گره در اینترنت از آدرس دهی IP استفاده می شود . در IoT نیز از روش های گوناگونی به منظور شناسایی دستگاه ها استفاده می شود . می توان به عنوان به شبکه لورا که از شناسه DevEUI استفاده می کند اشاره کرد . mac address دارای ۴۸ بیت می باشد که برای هرگره در کارت شبکه است ، در واقع با استفاده از آن با اینترنت ارتباط برقرار می‌کند. آدرس‌های MAC عمدتاً توسط سازندگان دستگاه تخصیص داده می‌شوند و بنابراین به عنوان آدرس سخت‌افزار یا آدرس فیزیکی نیز نامیده می‌شوند. این شناسه منحصر فرد است و نمی توان آن را تغییر داد .



۲۴ بیت ابتدایی نشان دهنده شناسه شرکت سازنده کارت شبکه است که در واقع این شناسه توسط IEEE به شرکت‌های سازنده اختصاص یافته است. این شناسه در واقع به عنوان OUI شناخته می‌شود. ۲۴ بیت بعدی در واقع کارت شبکه را مشخص می‌کند که توسط شرکت تولید کننده و به صورت منحصر به فرد برای هر کارت شبکه می‌باشد.

بیت b0 آدرس دهی چندپخشی و تک پخشی را تعیین می کند و بیت b1 آدرس دهی universal و local را تشخیص می دهد.

سوال ۴ – سوال چهارم

چنانچه برای هر سنسور یک مدل اطلاعاتی در نظر بگیریم ؛ کار تجزیه و تحلیل داده ها بسیار سخت می شود و همچنین با اضافه شدن سنسور جدید پیچیدگی های بیشتری در سمت تحلیل داده ها ایجاد می شود . پس روش مناسبی نمی باشد .

SenML برای ساخت برنامه های عمومی نیازی به متا دیتا ندارد می تواند از مقدار و تایپ برای طیف وسیعی از کاربردها استفاده کند. زمانی که منابع متعدد یا مقاصد در یک پیام واحد استفاده می‌شوند، نام‌های منحصربه‌فرد آن‌ها را از هم متمایز می‌کند . مقادیرSenML امکان بهینه‌سازی مقادیر پایه را فراهم می‌کند .

SenML فرمتی را برای انتقال اطلاعات سنسور تعریف می‌کند که اطلاعات سنسورها و عملگرها را به صورت یک لیست ساده در قالب JSON ، شی باینری مختصر (CBOR) و یا XML تبادل می کند. این روش انتقال، مدل داده مشترکی را در نظر می گیرد. یک سنسور ساده، مانند یک سنسور دما، می تواند از یکی از این انواع قالب های گفته شده (json, xml, …) در پروتکل های مختلف مانند HTTP یا پروتکل CoAP به منظور انتقال اندازه گیری های سنسور یا پیکربندی آن استفاده کند.

SenML طوری طراحی شده است که پردازنده هایی با قابلیت های بسیار محدود می توانند به راحتی اندازه گیری سنسور را در مدیا خود انجام دهند همچنین اندازه گیری سنسور به روشی نسبتا کارآمد در یک سرور تجزیه کننده داده ها می تواند انجام شود . SenML می تواند برای انواع مدل های جریان داده استفاده شود. ب

فیلد های مورد استفاده در SenML در جدول زیر آمده است .

Name	Label	CBOR Label	JSON Type	XML Type
Base Name	bn	-2	String	string
Base Time	bt	-3	Number	double
Base Unit	bu	-4	String	string
Base Value	bv	-5	Number	double
Base Sum	bs	-6	Number	double
Base Version	bver	-1	Number	int
Name	n	0	String	string
Unit	u	1	String	string
Value	v	2	Number	double
String Value	vs	3	String	string
Boolean Value	vb	4	Boolean	boolean
Data Value	vd	8	String (*)	string (*)
Sum	s	5	Number	double
Time	t	6	Number	double
Update Time	ut	7	Number	double