باسمه تعالی

تمرین اول درس برنامه نویسی وب

سیدعلیرضا میردامادی / 9729043

سوال 1)

- در **Http** نسخه **1.1** اتلاف پهنای باند کمتری از **http 1** وجود دارد
- **http 1** فرض میکند که هر سرور باید یک **IP** متمایز داشته باشد، در صورتی که **http 1.1** هدر هاست را ساپورت میکند
- **Connection** ها در **http 1** قابل استفاده مجدد نیستند، اما در **http 1.1** این امکان به این پروتکل افزوده شد
- در **http 2** میتوان بر بستر یک ارتباط **TCP**، چندین جریان داده را ارسال کرد، بدون آنکه هیچ منبعی منابع دیگر را مانع شود (مانند آنچه در **http 1.1** رخ میداد)
- در **http 3** از **QUIC** برای هندل کردن **stream** ها استفاده میشود، درحالیکه در **http 2** از **TCP** استفاده میشد (در لایه **transport**)
- در **http 3** زمان **handshake** بسیارکوتاه از و تشکیل یک نشست امن بسیار سریع تر از **http 2** انجام میگیرد
- **http 3** تنها میتواند بر یک بستر ایمن و رمزگذاری شده تشکیل شود، در حالیکه **http 2** به قبل میتوانست بدون **HTTPS** نیز پیاده سازی شود.

همانطور که اشاره شد، استفاده از QUIC که بر پایه UDP پیاده سازی شده است، به جهت ارتباط سریع تر و تجربه کاربری بهتر است.

سوال 2)

آ)

منظور از پروتکل های Stateless این است که این پروتکل ها هیچ تاریخچه/وضعیتی از کلاینت هارا نگهداری نمیکنند، که به تبع باعث پیچیدگی کمتر و عملکرد سریع تر و بهتر میشود؛ باید این نکته نیز ذکر شود که در برخی از کاربرد ها که نیاز به ثبت وضعیت کلاینت ها داریم، مجبوریم از راه حل هایی غیربهینه تر استفاده کنیم تا وضعیت نشست یا Session نگهداری شود.

اما پروتکل ها و سرویس های stateful کاملا تراکنش ها و نشست هارا ضبط میکنند و به هر کاربر یا کلاینت بر حسب تاریخچه متفاوتش واکنش متفاوت نیز میدهد؛ برخلاف Stateless که به همه یک واکنش نشان میدهد.

ب)

HTTP یک پروتکل stateless است، که برای حل مشکلات آن از راه حل های زیر بهره میگیرد:

- **Cookies**
- **JSON Web Tokens**

به کمک این روش ها HTTP میتواند در کاربرد های گفته شده، اطلاعات مرتبط با کلاینت هارا ذخیره کند و هنگام مراجعه هرکاربر، پاسخ متناسب را به او بدهد.

سوال 3)

آ)

در Http شروع کننده ارتباط همیشه client است، بنابراین در کاربرد های alerting و chat ها، شرایطی پیش می آید که سرور نیاز است داده ای را به کلاینت ها بفرستد (message) و یا آنها را از موضوعی آگاه کند (alerting)، در اینگونه کاربرد ها چون ارتباط بین سرور وکلاینت پس از پاسخ سرور بسته میشود، با چالش مواجهیم.

ب)

- **برای حل آن میتوان از persistent connections استفاده کرد، بدین معنی که ارتباط بین کلاینت و سرور پس از response سرور بسته نمیشود، و سرور میتواند در زمان نیاز به کلاینت داده هایی را ارسال کند، این ارتباط تا زمانی که سرور صلاح بداند باز میماند.**
- **راه دیگر نیز استفاده از server-send event ها است، بدین معنی که از یک شی EventSource به منظور ساخت یک callback استفاده شود، که وقتی سرور پیام جدیدی ارسال کند اجرا میشود، این تکنیک server push نیز نامیده میشود.**
- **راه حل آخر نیز web Socket ها هستند. بدین طریق میتوان یک رابطه دو طرفه بین سرور وکلاینت ایجاد کرد که هر کدام در هر زمان میتوانند به یکدیگر پیام بفرستند، این تکنیک در چت ها بسیار استفاده میشود.**

سوال 4)

- **Protocol: ss**
- **User: Asghar**
- **Password: 1234!!**
- **Host domain: ss.myproxy.com**
- **Port: 1234**
- **Fragment: shadowSocks1**
-

سوال 5)

1. **503 Service Unavailable**: چراکه یکی از سرویس ها موقتا دچار مشکل شده است
2. **403 Forbidden**
3. **301 Moved Permanently**
4. **429 Too many Requests**
5. **204 No Content**: چراکه توکن در هدر فرستاده شده و پاسخ سرور بدنه ندارد.
6. **403 Forbidden**

سوال 6)

1. **Reverse proxy**: برای **load balancing** و **caching** که از دید کاربر پنهان است و سمت سرور عمل میکند
2. **Reverse proxy**: برای **load balancing** که به وضوح از دید کاربر پنهان است.
3. **Forward Proxy**: در سمت کلاینت پیاده سازی میشود و دسترسی به سرویس را محدود و مشروط میکند