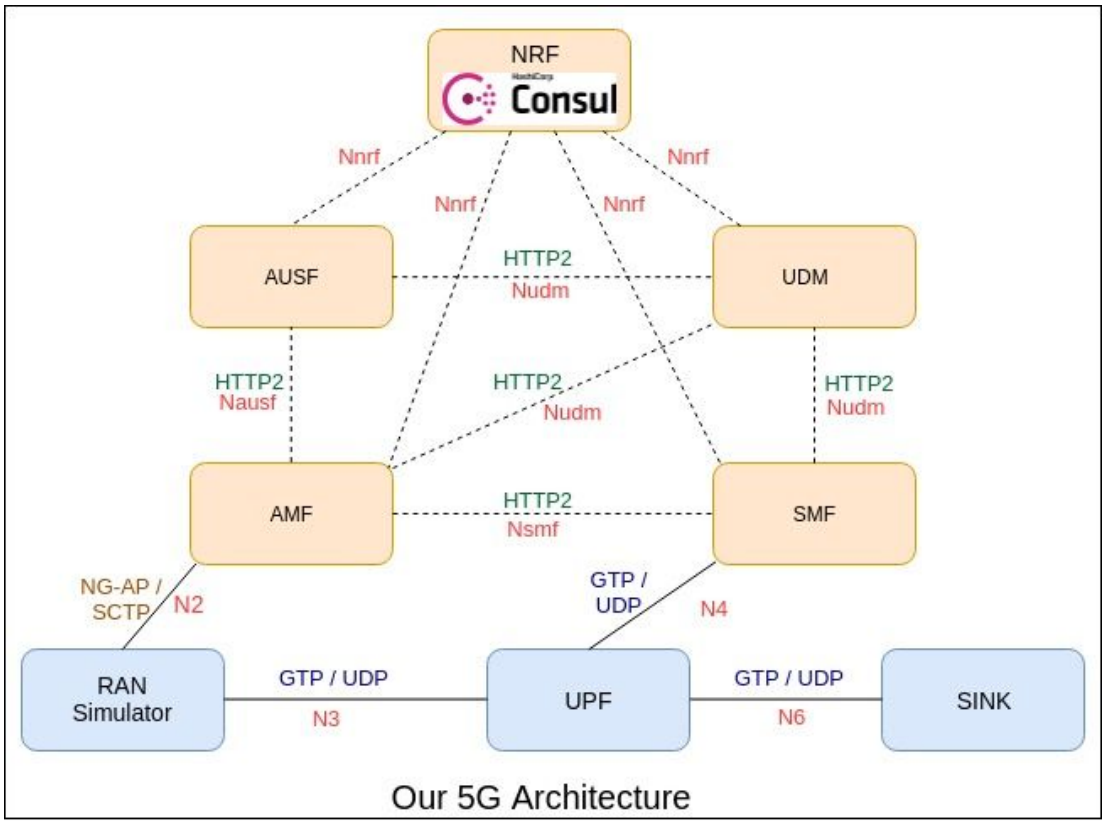
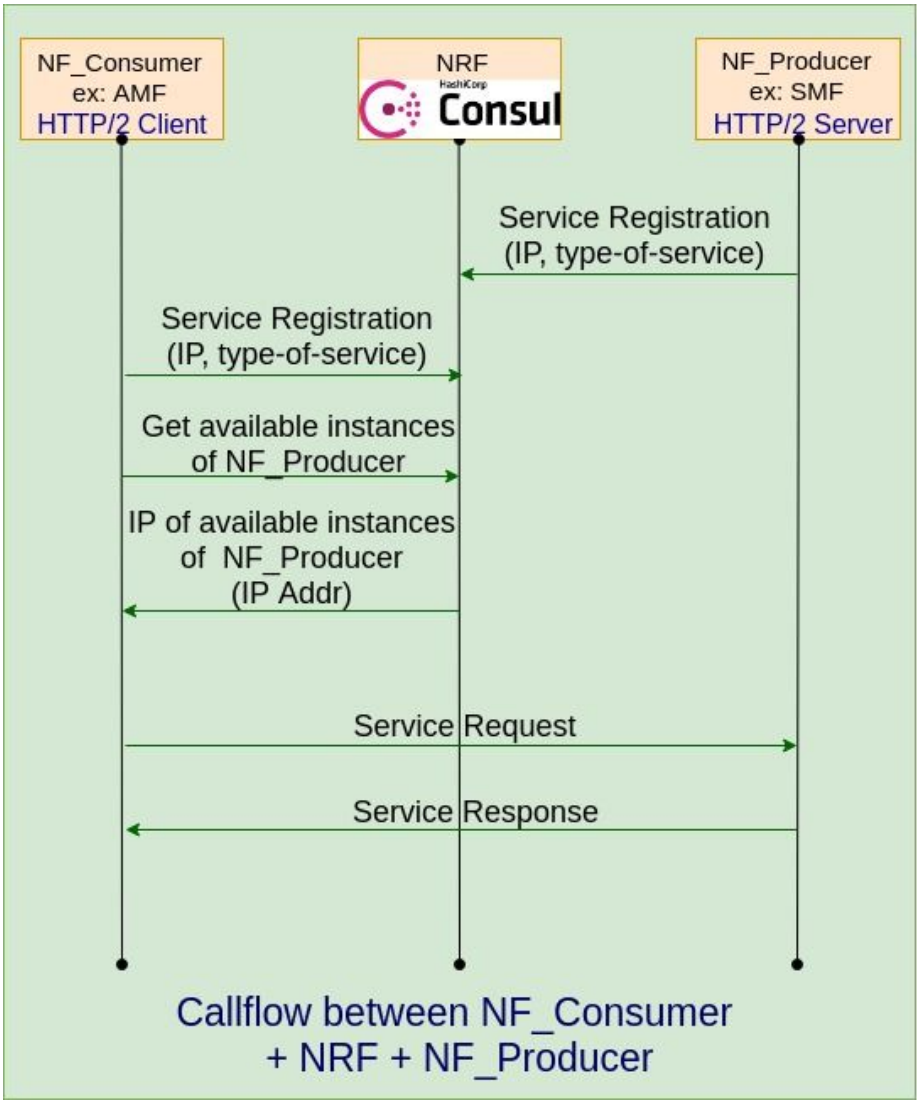
REST API based 5G Core Implementation

5G Architecture



The respective interfaces and communication protocols are shown in the above image.
The dotted lines are the control plane interfaces. The solid lines are the data plane interfaces.
Control Plane interfaces use HTTP/2, and the Data Plane interfaces use GTP tunnel and SCTP.
The control plane interfaces are named after the entity which plays the role of the server between the 2 entities.

Design Details REST API based 5GC



- We used nghttp library for implementing the HTTP/2 REST calls and Consul for the NRF.
- We replaced the reference point SCTP / GTP based communication channels with HTTP/2 REST calls on endpoints.
- As shown in the above diagram, all the NF instances register their IP address along with the type-of-service offered with the NRF (Consul).
- NF instances use Consul C++ client library github.com/oliora/ppconsul for communicating with Consul.
- When an NF_Consumer wants to access a service, it sends a service-discovery request for the Control Plane components to the NRF. The NRF replies back with the instance's IP address.

- The NF_Consumer caches this value and then communicates directly with the NF_Producer.
- The NRF API endpoints are shown in the table below.

Implemented HTTP/2 REST Calls

We have implemented the following services in our 5G control plane components (AMF, AUSF, SMF, UDM, NRF).
The data plane components (RAN, UPF, SINK) use the traditional socket based reference end-points.

Network Function	Service Name	Description	Service Endpoint	Known Consumers
SMF	Nsmf_PDUSession	This service manages the PDU Sessions and uses the policy and charging rules received from the PCF. The service operations exposed by this NF service allows the consumer NFs to handle the PDU Sessions.	(1) CreateSMContext	AMF
			(2) UpdateSMContext	AMF
			(3) ReleaseSMContext	AMF
AUSF	Nausf_UEAuthentication	The AUSF provides UE authentication service to requester NF. For AKA based authentication, this operation can also be used to recover from the security context synchronization failure situations.	(1) Authenticate	AMF
			(2) LocationUpdate	AMF
UDM	Nudm_UEAuthentication	1. Provide updated authentication related subscriber data to the subscribed NF consumer. 2. For AKA based	(1) GetAuthInfo	AUSF

		authentication, this operation can be also used to recover from security context synchronization failure situations. 3. Used for being informed about the result of an authentication procedure with a UE.	(2) SetLOCInfo	AUSF
	Nudm_UECM	Allow the NF consumer to update some UE context information in the UDM.	(3) UpdateInitialAttachInit	AMF
			(4) UpdateInitialAttach	AMF
			(5) UECtx/SetAuth	AMF
			(6) UECtx/SetSecurityCmd	AMF
			(7) UECtx/SetCrypt	AMF
			(8) UECtx//SetIntegrity	AMF
			(9) UECtx/HandleSecurityMode	AMF
			(10) UECtx/HandleLocationUpdate	AMF
			(11) UECtx/HandleCreateSessionBegin	AMF
			(12) UECtx/HandleCreateSessionEnd	AMF
			(13) UECtx/HandleAttachComplete	AMF
			(14) UECtx/UpdateAttachComplete	AMF
			(15) UECtx/HandleModifyBearer	AMF
			(16) UECtx/HandleDetach	AMF
			(17) UECtx/SetUPFInfo	AMF
			(18) UECtx/RequestSMFCreateSession	SMF
			(19) UECtx/UpdateSMFCreateSession	SMF

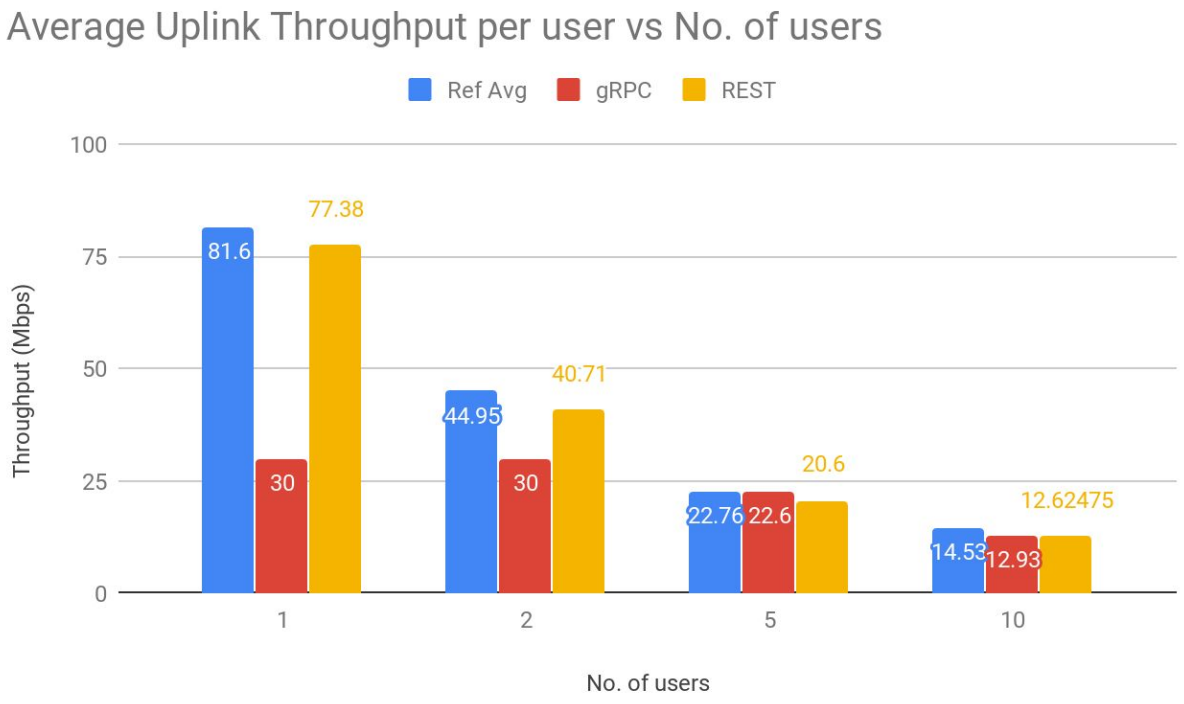
			(20) UECtx/RequestSMFModifyBearer	SMF
			(21) UECtx/RequestSMFDetach	SMF
NRF	ServiceRegister	When an instance boots up it registers its IP address with the NRF.	(1) /v1/agent/service/register	AMF, SMF, AUSF, UDM
	ServiceDiscover	When a NF_consumer wants to use a service it fetches Producer IP from NRF.	(2) /v1/agent/services	AMF, SMF, AUSF, UDM

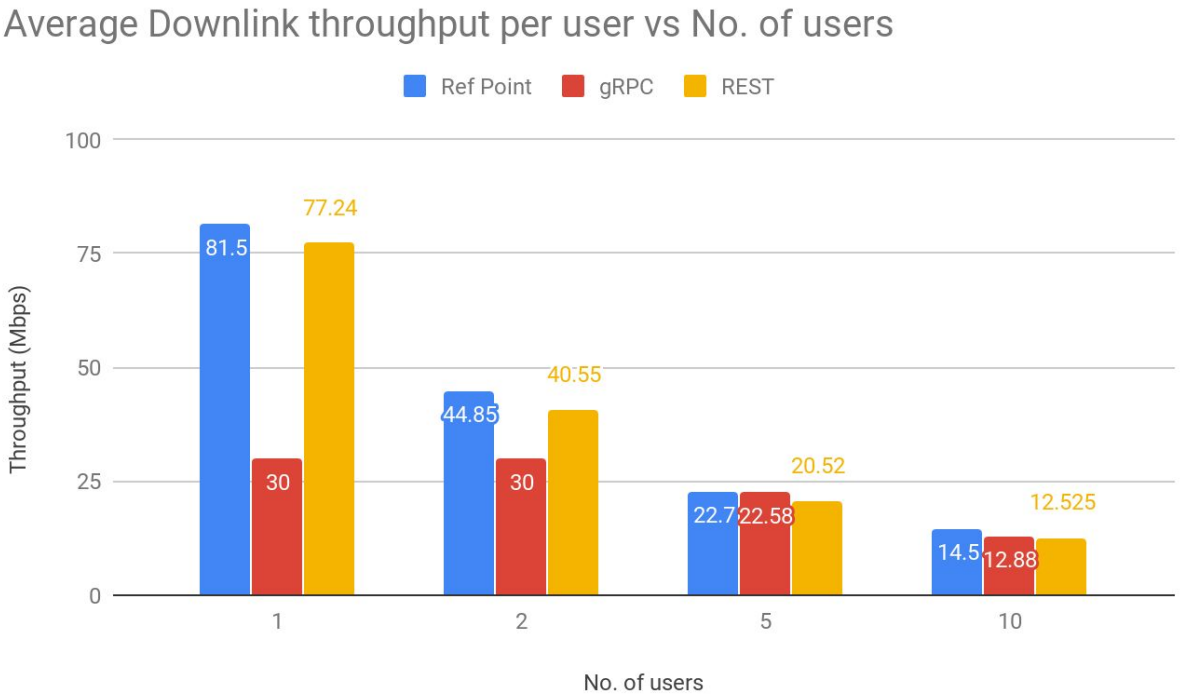
Task 1

Uplink and downlink Data throughput analysis for 1, 2, 5, 10 users for Ref Point vs gRPC-SBA vs REST.
We ran the REST code 5 times and plotted the average values. Ref Point and gRPC-SBA values are from previous assignments.

Test settings:

- For x users, x number of threads were created for all NFs.





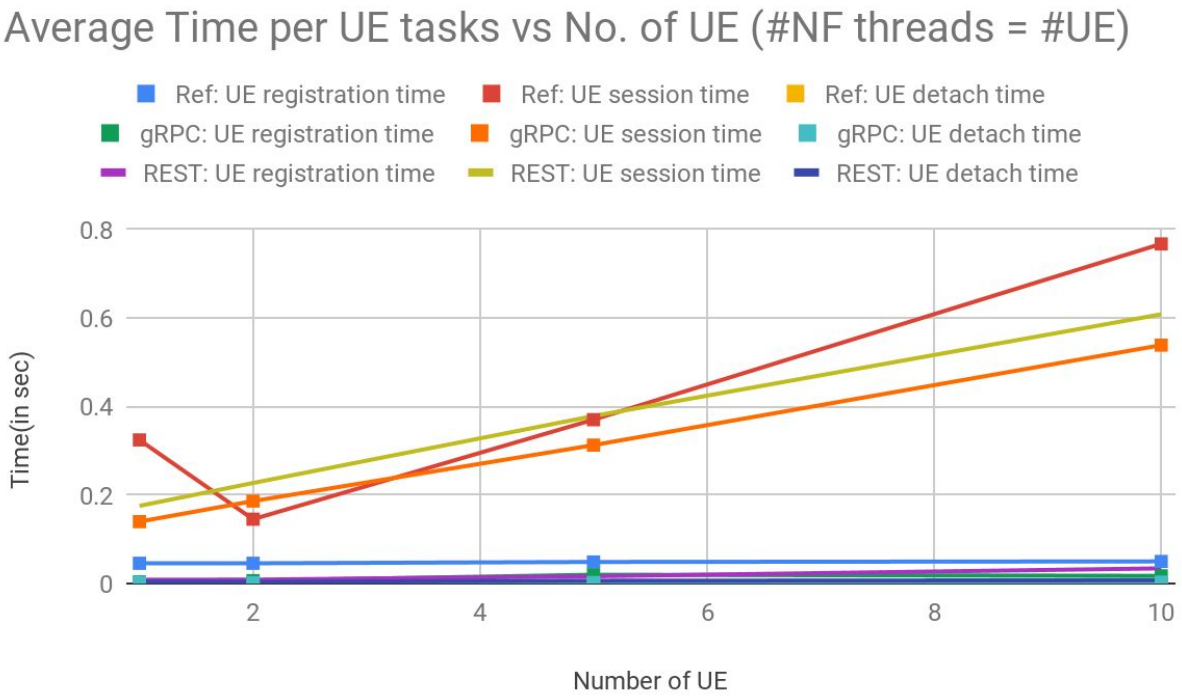
Uplink and Downlink observations:

- Throughput per user decreased with more number of users, as they have to share the band.
- Ideally, for 5 and 10 users, the throughput should be near 5x and 10x less for each user compared to 1 user. But it seems to be slightly higher.
    - With this, we can see that we are achieving higher overall throughput with more users.
    - This might be because a single user is not able to pump data at enough rate into the network, whereas more users are able to overcome that.
- We calculate uplink & downlink throughput for the data plane, which does not use REST, hence no observation on the throughput based on REST api changes.
- Throughput of REST is near to Ref Point in all cases. The small difference might be because of system conditions are the time of testing as the changes are only in the control plane.
    - REST is also similar to gRPC except the anomaly for 1 and 2 UE.

Task 2

For this task we have plotted the average time taken for UE registration, UE session creation and UE detach per transaction, keeping the number of threads in the different NFs equal to the number of UE. We ran the REST code 5 times and plotted the average values.

Average Time per UE tasks vs No. of UE (#NF_threads = #UE)				
No. of UE	1	2	5	10
UE registration time	0.0084844	0.0085872	0.01635618	0.03411612
UE session time	0.1753326	0.2269646	0.3782406	0.60794
UE detach time	0.0042086	0.003658	0.00617906	0.00723634



Observations:

- Session creation takes the maximum time out of all 3 tasks. For a small number of UE, Ref-Based is faster.
  But as the number of UE increases, Ref-based becomes slower than all 3. gRPC is the fastest and REST lies between Ref & gRPC.
  Reason: Since Session creation takes the maximum time, serializing and de-serializing payload plays an important role.

Since gRPC uses protobuf, it is computationally least expensive to serialize and deserialize. gRPC is built on top of HTTP/2 and is further optimized in terms of latency and CPU utilization. Hence gRPC is fastest in Session Creation.

REST is faster than Ref-based because it uses HTTP/2 features like:

- HTTP/2 is binary, instead of textual.
- HTTP/2 is fully multiplexed.
- It uses header compression HPACK to reduce overhead.
- It reduces additional round trip times (RTT).

Ref-based uses SCTP for communicating, thereby making it slower than HTTP/2.
For 10 users the gap between gRPC and REST is lesser than REST and Ref-based. This is because gRPC is an optimized version of HTTP/2, whereas Ref-based uses a completely different technique (which is slower).

- Session creation time per user increases with the number of users.
  Reason: This is because session creation is the most computationally expensive out of the 3 tasks. Since all UE threads are mapped to an SMF thread and are all running on the same docker container, this leads to more contention, thereby leading to more delay with more users.

- Detach is the fastest task out of the three tasks.
  Reason: This is because it involves a smaller payload and requires lesser computation to complete it.

- UE Registration & Detach time per user is almost the same for a different number of concurrent users.
  Reason: Registration and detach are computationally very light tasks. Since each UE request is mapped to an AMF thread the request is serviced very quickly. Hence the increasing number of users does not increase contention as drastically as SMF.

Task 3

For this task we have plotted the average time taken for UE registration, UE session creation and UE detach per transaction. We ran the REST code 5 times and plotted the average values. We have considered two different scenarios for this task:

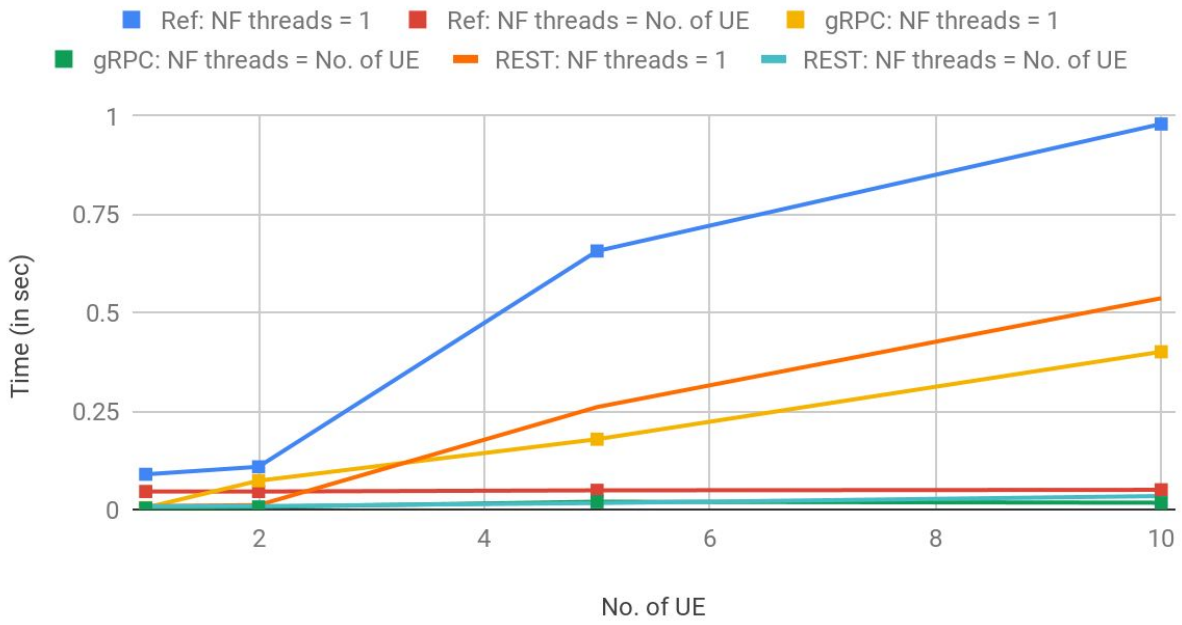
1. Number of NF_threads = 1
2. Number of NF_threads = Number of UE

Average Registration Time per UE vs No. of UE				
No. of UE (Registration)	1	2	5	10
NF_threads = 1	0.0084844	0.01111056	0.2601328	0.5366366
NF_threads = No. of UE	0.0084844	0.0085872	0.01635618	0.03411612

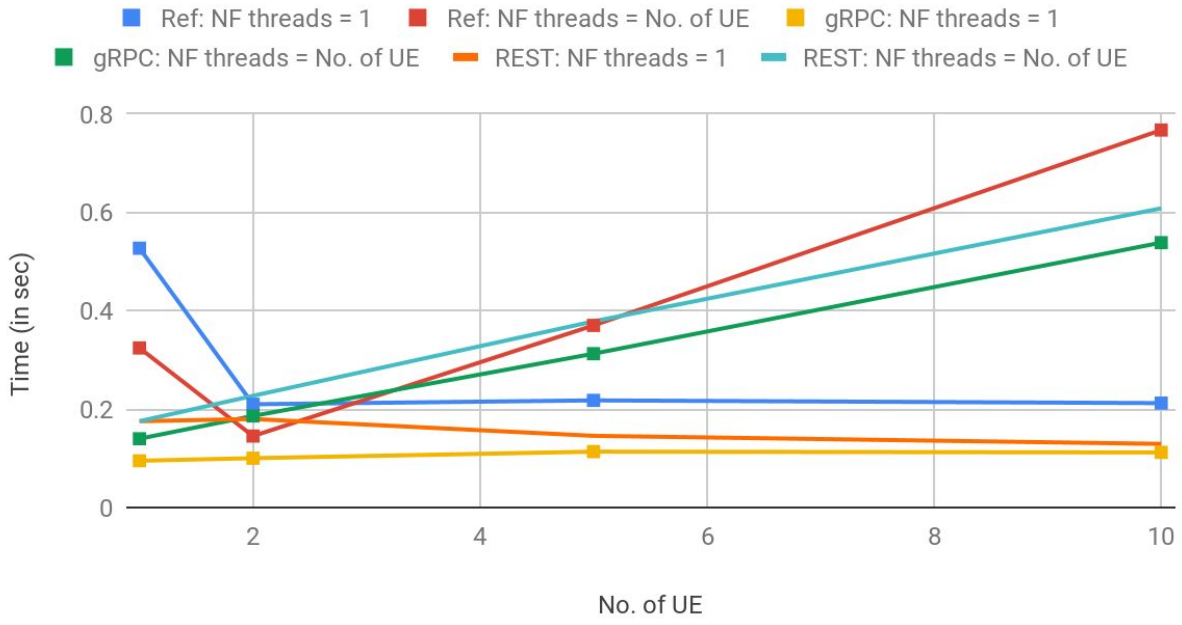
Average Session Creation Time per UE vs No. of UE				
No. of UE (Session)	1	2	5	10
NF_threads = 1	0.1753326	0.1798534	0.1456166	0.1292704
NF_threads = No. of UE	0.1753326	0.2269646	0.3782406	0.60794

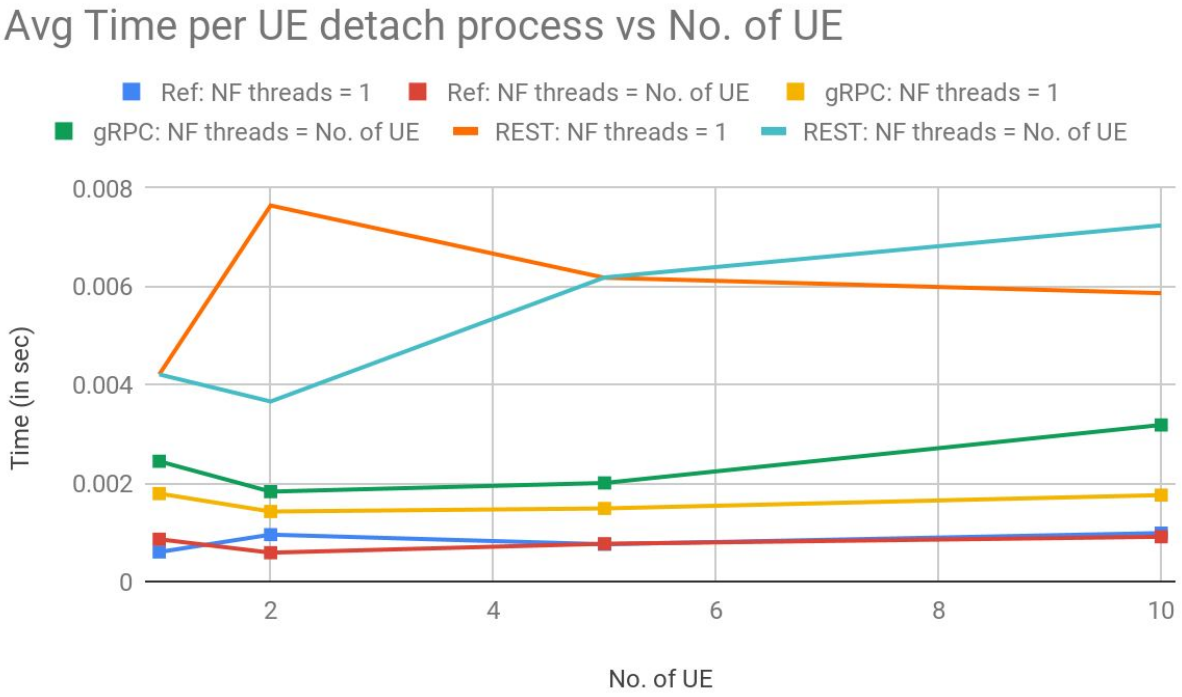
Average Detach Time per UE vs No. of UE				
No. of UE (Detach)	1	2	5	10
NF_threads = 1	0.0042086	0.0076408	0.00617288	0.00585984
NF_threads = No. of UE	0.0042086	0.003658	0.00617906	0.00723634

Avg Time per UE registration vs No. of UE



Avg Time per UE session creation vs No. of UE





Overall Observations for task 3:

- For UE registration, when NF_threads = No. of UE, there is not much difference in the time taken by all 3 approaches. The reason explained above.
  In both the cases, UE Registration is slowest for Ref-based and fastest for gRPC based. UE registration with gRPC is more than twice faster than Ref-based. The reason is explained above.
  But all 3 are slower than the corresponding NF_threads = No. of UE because when NF_threads = 1, the UE needs to wait before it is allowed to be serviced.

- For UE session creation, gRPC was the fastest and Ref-based was the slowest for both less and more number of threads at each NF. The reason is explained above.
  NF_threads = 1 was faster than NF_threads = No. of UE in Session creation.
  Reason: For NF threads = 1, Registration is the bottleneck whereas, for NF_threads= No. of UE, Session creation is the bottleneck.
    - This might be because when 1 user is registered, it goes into session creation phase. Whereas rest of the users are still stuck at the registration phase. Hence the user which finished registration doesn't have much contention for session creation.

- For NF_threads = 1, For UE detach, gRPC was slower than Ref
    - This is might be because: Detach is a very fast process and the data to be transmitted is very small. So the extra overhead of gRPC processing tends to be overkill for this task.

- And with such small values, there are chances of noise in the data.
  - OS scheduling time, etc.

Task 4

The comparison between REST and Ref Point and gRPC is done inside the description of Task 1, 2 and 3 above.

PLAGIARISM STATEMENT <Include it in your report>

We certify that this assignment/report is our own group's work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other students if we become aware of it.

Names of group members:

Date: 11 March 2019.

Signature:

GV

(Ganesh Vernekar)

HA

(Harsh Agarwal)