```
class HelloService : public Service {

    override onCreate() {
        registerService Handler([illegible]);
    }

    override onBind() {
        ;
        ;
    }

    override onUnbind

    override onRebind

    override onDestroy
```

Registred a Service Handler while creating the Service,

Whether to register during onCreate or onBind

```
private class ServiceHandler(message msg){
    NotificationManager n
    n.register(msg)
}
```

onCreate can be implicitly called so that any app/activity can actually straightaway call bind. If no such Service thread is created, then the bind function automatically creates the Service thread.

Every app has used unbind. Then, onDestroy is called automatically. Background job would completely stop. Other apps which want to use it might notice that nothing has happened in the background.

Background jobs are run with weaker priority on the weaker / less powerful cores. They consume memory. It is possible that Android's framework actively stops an app from creating more services.

$\Downarrow$

You should not utilize Services for critical work.

Payment $\Rightarrow$ Not as a service. Otherwise data might become inconsistent, which is undesirable

---

A second scheme of creating background tasks is to use WorkManager class and schedule tasks using it.

Services don't run in Threads. So with a lot of tasks, they might become unresponsive. (Small amount of computation per service).

Most services tend to use notifications and
{network resources / sensors}

⇓

Continuous access to these resources.

These Resources can be expensive to use; both in monetary cost and power consumption.

⇓

{Minimize the amount of probes on the HW as much as possible.

During mediation by the service, you can coalesce / merge the requests ; across multiple apps including messages and emails. Cannot be done for any real-time application, (watching video or listening to audio)

⇒ they consume more power.

Can we move some computation of the Services into a server machine ??

$\Rightarrow$ move the computation to a server, and thus save energy; ~~or~~ can it be useful?

$\Rightarrow$ Yes; it is often done; but suffers from privacy challenges.

Is there something that can be done?
Federated learning is one strategy.

Latency aspect $\Rightarrow$ Services typically do not require very low latency.
$\Rightarrow$ Not very good for reliable applications.

Smartphones are integrated with cloud services. $\Rightarrow$ X86-64 architecture

ARM architecture

$\Rightarrow$ Why is this supposed to keep improving user experience ??

$\Rightarrow$ Minimum QOS (quality of service) for wireless networks

[in the worst case; what is the minimum

bandwidth?

How is the problem of different architectures handled?

$\Rightarrow$ Translate machine code from ARM to x86-64.

$\Downarrow$

Not in runtime $\Rightarrow$ happen a priori

{ Phone's APK can activate a service on the cloud's version of APK.

$\Rightarrow$ Google Translate, etc.