Timeouts and Deadlines

To abandon synchronous calls that run too long, use the select statement with time.After:

```
import "time"

c := make(chan error, 1)
go func() { c <- client.Call("Service.Method", args, &reply) } ()
select {
  case err := <-c:
    // use err and reply
  case <-time.After(timeoutNanoseconds):
    // call timed out
}
```

Note that the channel `c` has a buffer size of 1. If it were an unbuffered channel and the client.Call method took more than `timeoutNanoseconds`, the channel send would block forever and the goroutine would never be destroyed.

References

time.After: https://pkg.go.dev/time/#After

select: https://go.dev/ref/spec#Select_statements

blog post: https://go.dev/blog/2010/09/go-concurrency-patterns-timing-out-and.html