The sad state of TLSv1.3 client support

This page constitutes random notes from my work day as an Atlassian product consultant, put up in the vague hope they might benefit others. Expect rambling, reference to unsolved problems, and plenty of stacktraces. **Check the date** as any information given is likely to be stale.

TLSv1.3 is relatively new and shiny, as fundamental web protocols go, and **TLS stacks are still working out the bugs**. In my experience:

- Back in April I had a client whose Jira connections to Slack would hang soon after server start.

  This turned out to be caused by a TLS 1.3 bug in Java 11. Jira loses track of the TCP connection state (leaving the TCP connection in CLOSE-WAIT forever) and blocks, thereby blocking the whole webhook thread pool:

  https://jira.atlassian.com/browse/JRASERVER-70780
  https://jira.atlassian.com/browse/JRASERVER-70189

  This was triggered by some change on Slack's end. Perhaps Slack switched their default from TLS 1.2 to 1.3 when they removed older TLS versions the previous month.
- Same client, 4 months later, suddenly experienced complete failure of Crowd to authenticate users against their LDAP.

  The relevant line in the stacktrace is:

  ```
  Caused by: org.springframework.ldap.CommunicationException: ldap.phx7.<redacted>.com:636; nested exception is javax.naming.CommunicationException: ldap.phx7.<redacted>.com:636 [Root exception is javax.net.ssl.SSLHandshakeException: extension (5) should not be presented in certificate_request]
  ```

  Once again some hurried Googling suggested this is a TLSv1.3 problem.

In both cases the solution was to avoid TLSv1.3 for now, by setting the Java flag:

```
-Djdk.tls.client.protocols=TLSv1.2
```

The first bug might also be avoidable by upgrading to Java 11.0.8 or higher. I haven't tested that yet though.

For now, I'm setting that flag in all instances I administer, until TLS servers and clients get their act together.