

Zvýšenie efektivity OCSP komunikácie

Študent: Peter Brenkus

Vedúci práce: Ing. Norbert Varga

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Obsah

1. Uvedenie do problematiky

2. Implementované riešenie

3. Testovanie a výsledky

1.1 Digitálne certifikáty a OCSP protokol

- Základ bezpečnej komunikácie na internete
- Overenie identity serverov
- Môžu byť revokované
- Potreba zistiť ich stav v aktuálnom čase
- CRL alebo OCSP

1.2 Nedostatky OCSP

- Dodatočné sietové spojenie
- Dostupnosť OCSP respondérov
- Súkromie klientov

2. Implementované riešenie

2.1 Implementované riešenie

Zamerané na:

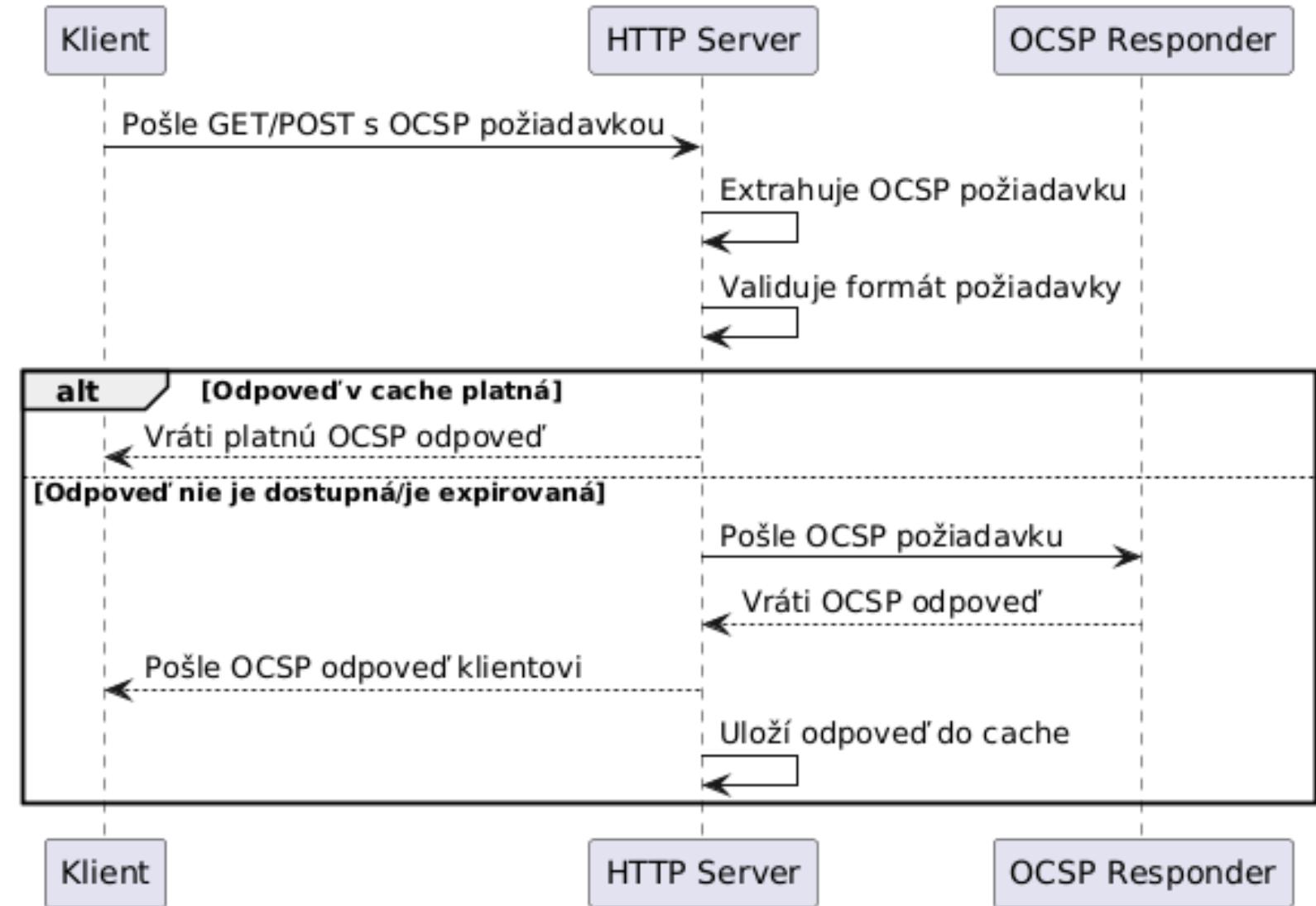
- zníženie času OCSP komunikácie
- zníženie záťaže OCSP respondérov

Hlavná myšlienka:

- OCSP je nad HTTP
- cache OCSP odpovedí na úrovni HTTP servera
- odpovede pre GET aj POST metódy

2.2 Architektúra riešenia

- HTTP Server v jazyku Java (Jetty 11.0.15)
- Redis cache (Jedis 4.3.1)
- Práca s OCSP protokolom BouncyCastle 1.70
- Logback Core 1.2.11

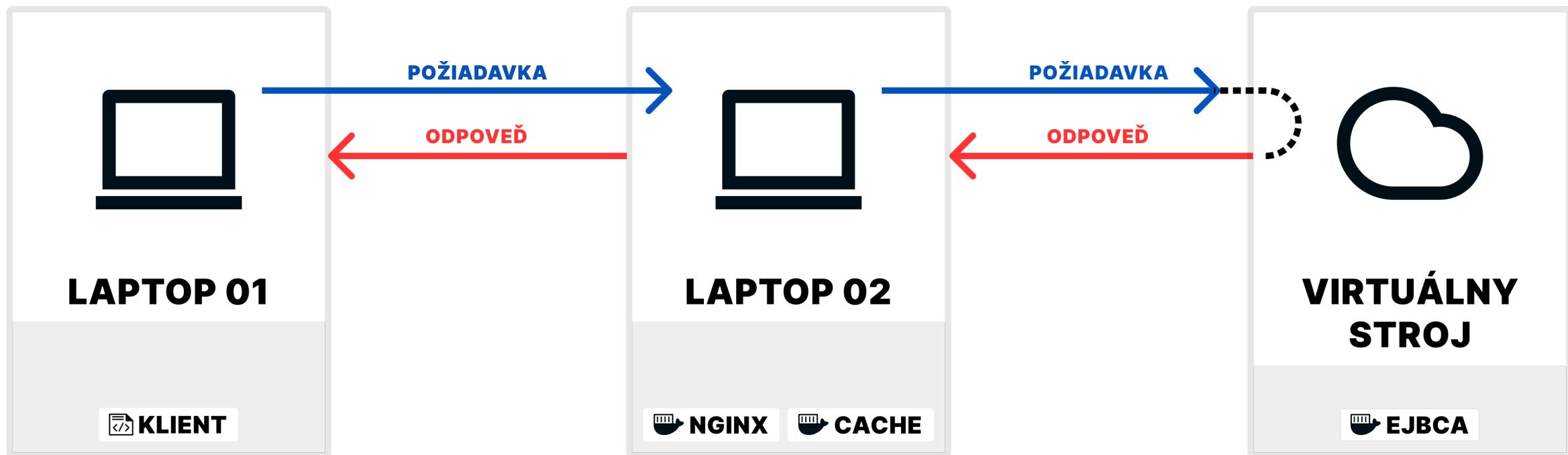


2.3 Najdôležitejšie vlastnosti

- Sériové číslo certifikátu = kľúč v cache
- Jedna odpoveď môže obslúžiť aj GET aj POST požiadavky
- Podpora unikátnych požiadaviek s *nonce*
- Podpora konfigurácie
- Robustnosť

3. Testovanie riešenia

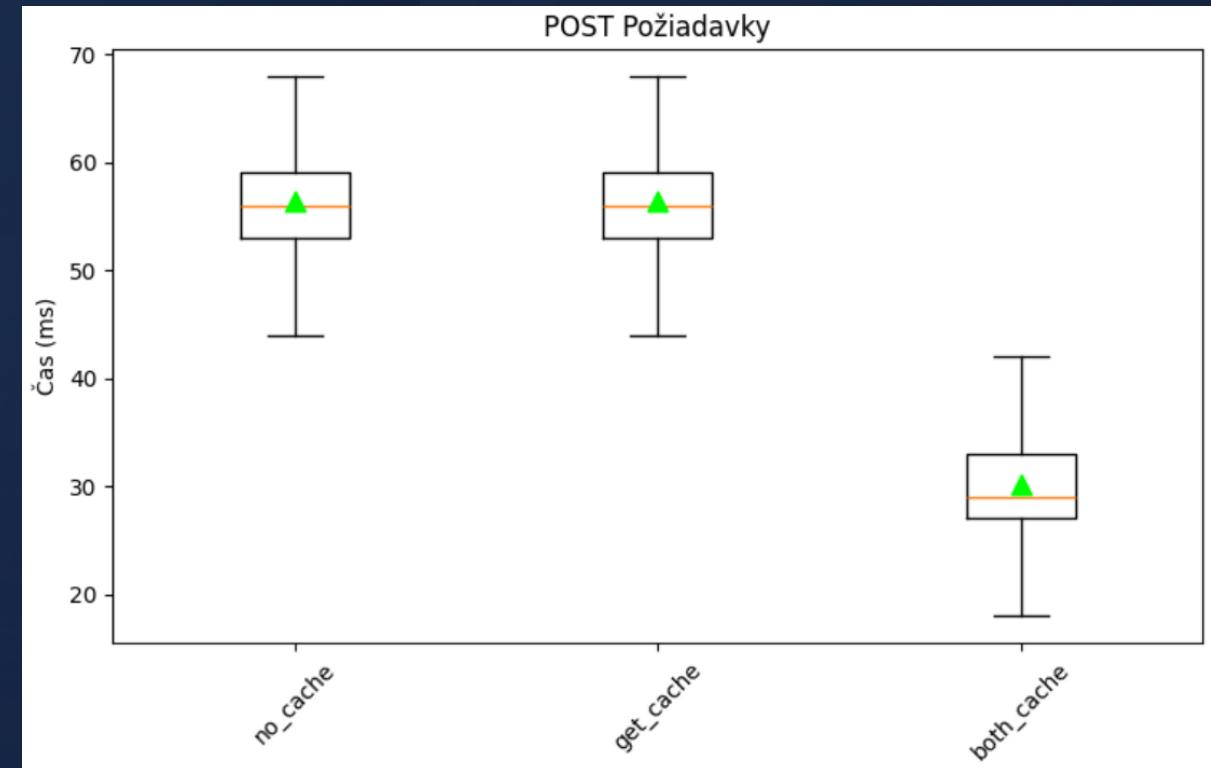
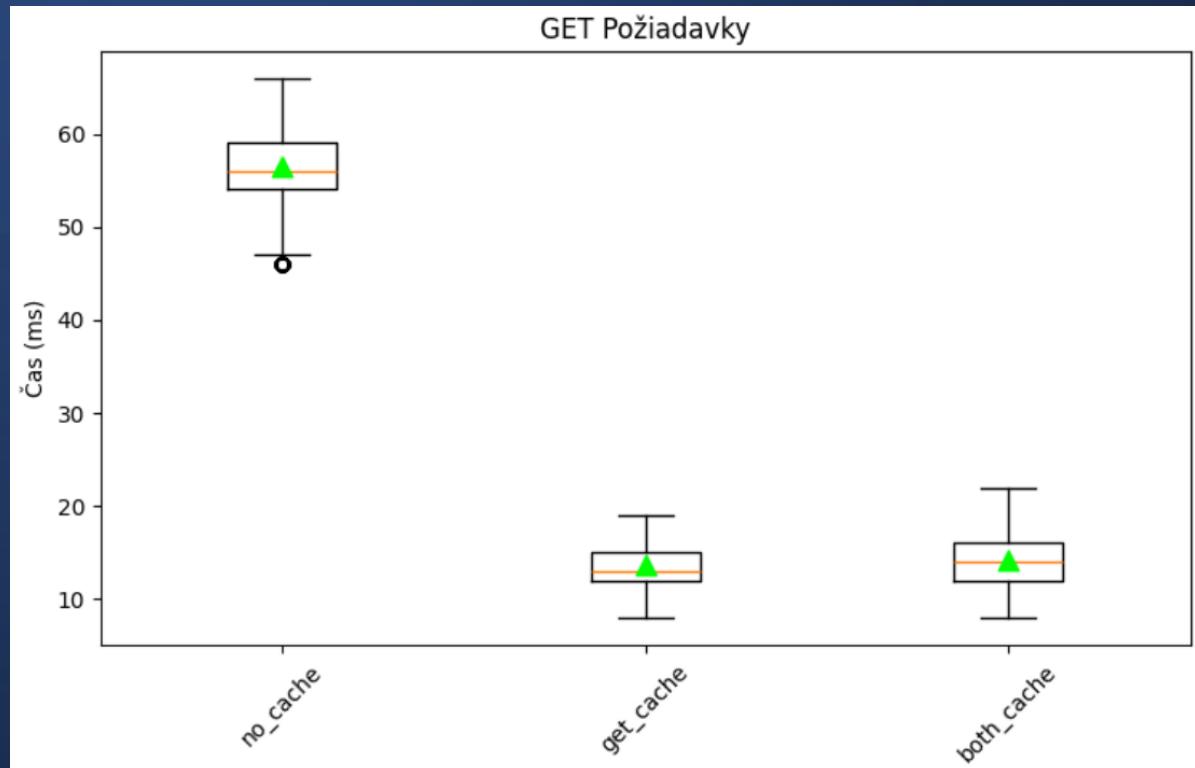
3.1 Testovacie Prostredie



3.2 Testovanie času odozvy

- Priemerná rýchlosť spracovania jednej požiadavky
- Bez Cache – Nginx Cache – Implementovaná cache
- Rôzne časy dňa (6:00, 12:00, 18:00, 24:00)
- 1000 požiadaviek – 50/50 GET/POST

3.2 Výsledky



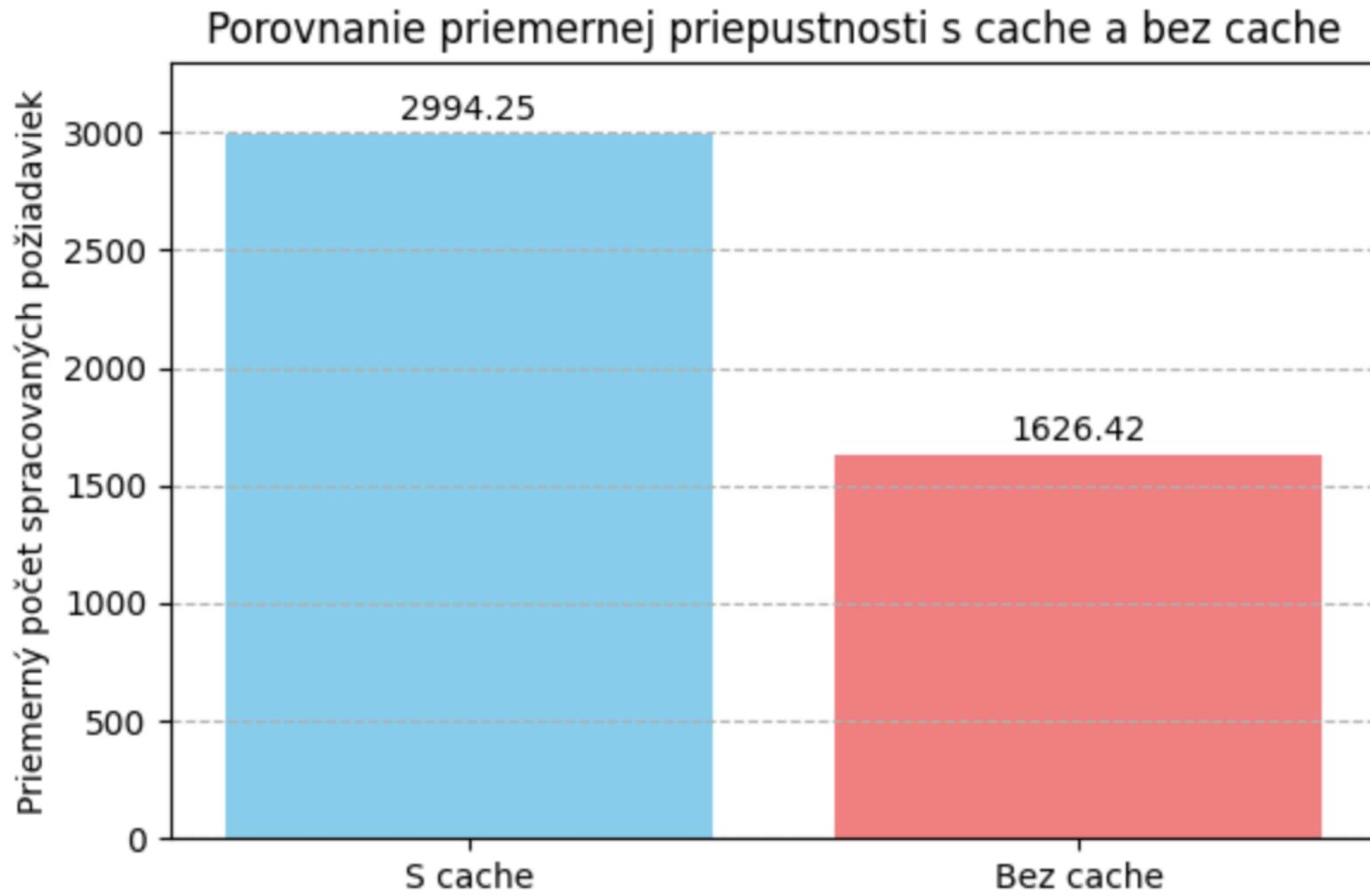
- Pokles o 46.5 %

3.3 Testovanie priepustnosti

- Priemerný počet spracovaných požiadaviek za daný čas
- Bez cache – Implementovaná cache
- Rôzne časy dňa (6:00, 12:00, 18:00, 24:00)
- 4 paralelné POST požiadavky
- 30 sekúnd, tri opakovania

3.3 Výsledky

- Nárast o 84.1 %



Zhodnotenie

- Cache pre OCSP požiadavky
- Kľúč = sériové číslo certifikátu
- Cieľ = zníženie záťaže OCSP respondérov
- Výsledky = lepšia odozva a priepustnosť

1. Aká doba uloženia odpovede v cache je "rozumná"? Treba vychádzať z toho, že počas uloženia odpovede v cache môže prísť žiadosť o odvolanie certifikátu

RFC6960 o poli „nextUpdate“:

- The time at or before which newer information will be available about the status of the certificate.
- If nextUpdate is not set, the responder is indicating that newer revocation information is available all the time

```
// Určenie TTL podľa nextUpdate ak je nastavený, inak default
int ttlToUse = CACHE_TTL;
if (CACHE_ENABLED) {
    try {
        OCSPResponseParser.parseOCSPResponse(ocspResponseBytes);
        Date nextUpdate = OCSPResponseParser.getNextUpdate(ocspResponseBytes);
        if (nextUpdate != null && !cacheHit) {
            long ttlMillis = nextUpdate.getTime() - System.currentTimeMillis();
            int ttlSeconds = (int) (ttlMillis / 1000);
            if (ttlSeconds > 0) {
                ttlToUse = ttlSeconds;
                logger.debug("Používam TTL z nextUpdate: {} sekúnd", ttlToUse);
            }
        }
    }
}
```

2. Aké certifikáty posielal klient na overenie- boli to certifikáty platné alebo odvolané? Tento fakt môže ovplyvniť výsledky testovania priepustnosti.

- Testovanie prebehlo s jedným platným certifikátom
- Z pohľadu cache nezáleží na stave – prosté ukladá odpovede
- Na miere zaplnenia (počte rôznych certifikátov) tiež nezáleží - cache je heš tabuľka s prístupom $O(1)$

3. Aký je rozdiel medzi spracovaním požiadavky GET a POST?

- Z pohľadu mojej aplikácie sa spracúvajú rovnako
- Pomocou objektov z Bouncycastle knižnice pre OCSP dáta sa spracováva obsah OCSP správ