VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÝ SERVER PRO VYSTAVENÍ WEBOVÉ SLUŽBY NA BANANA PI PRO

WEB SERVER FOR EXPOSING WEB SERVICE ON THE BANANA PI PRO

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Jaroslav Válka

AUTHOR

VEDOUCÍ PRÁCE — Ing. David Troják

SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Jaroslav Válka ***ID:*** 159527
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Webový server pro vystavení webové služby na Banana Pi Pro

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s volně dostupnými softwarovými webovými servery vhodnými pro hardwarovou platformu Banana Pi Pro.
2. Vyberte minimálně dva softwarové webové servery, kde minimálně jeden bude podporovat HTTP/2.
3. Vybrané webové servery nasaďte na hardwarovou platformu Banana Pi Pro.
4. Vystavte webovou službu za pomoci Node.js na obou vybraných serverech.
5. Proveďte porovnání z hlediska rychlosti a maximálního počtu současných připojení při provádění žádosti na webovou službu.

DOPORUČENÁ LITERATURA:

[1] GRIGORIK, Ilya. High-performance browser networking. Sebastopol, CA: O'Reilly, 2013. ISBN 978-1449344764.

[2] SHARMA, Rahul. NGINX High Performance. Birmingham: Packt Publishing, 2015. ISBN 978-1-78528-183-9.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: Ing. David Troják
Konzultant bakalářské práce: Ing. Juraj Jakubík

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Cílem této bakalářsképráce je vystavení webové služby na webovém serveru s využitím platformy Banana Pi Pro. Webový server, nasazen na hardwarové platformě Banana Pi Pro, je řešený formou open source softwaru, v tomto případě se jedná o webové servery Apache a Nginx. Na webový server je vystavena webová aplikace pomocí softwarového systému Node.js, který slouží jako platforma pro programovací jazyk Javascript. Praktická část projektu se zaměřuje na porovnání webových serverů z hlediska rychlosti a maximálního počtu současných připojení při provádění žádosti na webovou službu.

KLÍČOVÁ SLOVA

Apache, Nginx, Banana Pi Pro, Node.js, webový server, webová služba, HTTP/2, apache benchmark

ABSTRACT

The aim of this thesis is exposed web services on the web server using the Banana Pi Pro platform. Web server installed on a hardware platform Banana Pipra is designed in the form of open source software, in this case, the Web server Apache and Nginx. On the Web server is exposed to the web application using software system Node.js, which serves as a platform for the programming language Javascript. Is practically part of the project focuses on the comparison of Web sites in terms of speed and maximal number of simultaneous connections in the implementation of the request to the Web service.)

KEYWORDS

Apache, Nginx, Banana Pi Pro, Node.js, web server, web service, HTTP/2, ApacheBench, Chrome DevTools, jQuery, Javascript, JSONP

VÁLKA, Jaroslav *Webový server pro vystavení webové služby na Banana Pi Pro*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 42 s. Vedoucí práce byl Ing. David Troják

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Webový server pro vystavení webové služby na Banana Pi Pro“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Davidovi Trojákovi a konzultantovi bakalářské Ing. Jurajovi Jakubíkovi práce za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

V dnešní době již pořízení domácího webového serveru (řešení hardware a software v jednom celku) není drahou či hodně místa zabírající záležitostí. Při využití různých platforem a open source řešení je takové sestavení webového serveru velice levné a docela minimalistické. Výhodou oproti komerčním řešení či pronájmu webového serveru je naprostá možnost ovládání, nahlédnutí do zdrojových kódů, fyzická přítomnost serveru v domácnosti (možnost server kdykoliv fyzicky odpojit od sítě, či zkontrolovat funkčnost).

Cílem této bakalářské práce je porovnání dvou volně dostupných softwarových webových serverů, nasazených na hardwarovou platformu Banana Pi Pro, z hlediska rychlosti a maximálního počtu současných připojení při provádění žádosti na webovou službu, která bude vystavena za pomocí Node.js. Přičemž minimálně jeden ze serverů bude podporovat novou verzi protokolu HTTP - HTTP/2 a bude použit. Tato bakalářská práce se dělí na 4 části. První část se zabývá teorií. Druhá část prezentuje použité technologie, popisuje použité webové servery, platformu pro webovou službu a operační systém pro Banana Pi Pro. Třetí část je zaměřena na zapojení Banana Pi Pro, instalaci operačního systému, nasazení webových serverů a jejich konfiguraci spolu s vystavením webové služby. Poslední část popisuje způsob porovnání jednotlivých serverů a ukazuje konkrétní hodnoty porovnání.

1 TEORETICKÁ ČÁST

1.1 Webový server

Pod pojmem webový server si můžeme představit dvě věci: 1.Webový server jako program 2. Webový server jako počítač, na kterém je spuštěný webový server-program.

Webový server funguje na principu klient-server. V prostředí WWW se klient nazývá prohlížeč a server WWW server nebo webový server. Klient přeloží požadavek na získání určité stránky z webového serveru do jazyka protokolu HTTP, vytvoří spojení se serverem a požadavek odešle. Server trvale očekává požadavky na standardním portu 80 (naslouchá) a jakmile požadavek přijde, pak jej ověří, najde příslušnou stránku a zašle ji ve stejném protokolu klientovi. Prohlížeč obdrženou HTML stránku podle značek zformátuje a zobrazí. Nenajde-li server požadovaný soubor, zašle chybovou zprávu a spojení ukončí. Portem se rozumí číslo, které je charakteristické pro určitou službu a v operačním systému slouží k určení toho programu, který bude pakety obsahující určitý port zpracovávat. Takový program může být právě jeden.

Zkratka URL (Uniform Resource Locator) vyjadřuje formalizovaný způsob vyjadřování adresy určitého dokumentu. Skládá se ze způsobu přístupu k dokumentu, umístění serveru a cesty k dokumentu.[1]

1.2 Proxy server

Proxy server je server, který je umístěn mezi klientem a serverem a dělá prostředníka v jejich komunikaci. Proxy se tváří vůči klientovi jako server a přebírá jeho požadavek. Ten pak sám odešle serveru a také přebírá odpověď, kterou nakonec doručí klientovi. Proxy server se tady chová jako server při komunikaci s klientem a jako klient při komunikaci se serverem. Proxy server navíc může komunikaci, kterou obsluhuje měnit a zpracovávat (filtrování, cachování, statistika...). Proxy serverem je možné zabezpečovat síť (kontroluje přístup ven, lze použít místo NAT; povoluje pouze služby, které podporuje; klient nekomunikuje přímo s vnější sítí), urychlovat odezvu z vnější sítě pomocí cache, řídit přístup ke službám.

Aby klient využíval ke komunikaci proxy server je většinou potřeba jej nějak nastavit, případně upravit. Pokud všechny požadavky směřují automaticky na proxy, mluvíme o transparentním proxy serveru.[11]

1.2.1 Reverzní proxy server

Reverzní proxy server je principem velmi podobný proxy serveru. Sdružuje (resp. rozděluje) požadavky klientů, kteří jsou k němu připojeni. V praxi se nejčastěji využívá s několika připojenými servery. Všechna připojení z internetu směřující na některý ze těchto serverů jsou směrována přes tento reverzní proxy server, který buďto požadavek zpracuje sám nebo ho předá dál serverům.

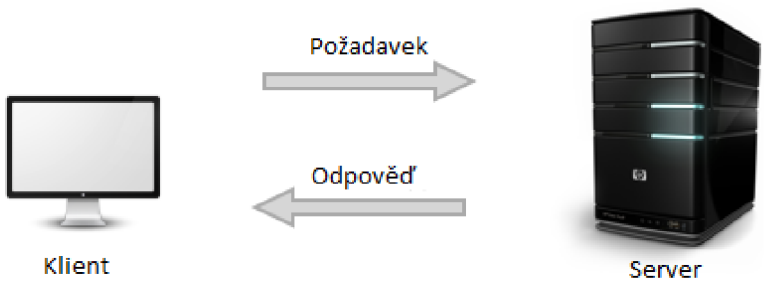
Reverzní proxy rozděluje vstupující provoz na několik serverů zachovávajíce jediný vnější rozhraní pro klienta. Naopak obyčejný proxy server se využívá pro provoz do vnější sítě – příkladem může být proxy na výstupu ze sítě ISP, který předává HTTP požadavky do internetu a díky cachování snižuje zátěž linky.

1.3 HTTP

HTTP (HyperText Transfer Protocol) je protokol aplikační vrstvy, sloužící k přenosu dat v systému World Wide Web. World Wide Web (www) sestává dokumentů HTML uložených na webových serverech, které je prostřednictvím HTTP zpřístupňují klientům, webovým prohlížečům. Jedná se o protokol bezstavový. To znamená, že server si neudržuje žádné informace o svých klientech. Obdrží-li dotaz, odpoví na něj a tím pro něj skončila jedna ucelená HTTP transakce. Pokud zanedlouho dostane dotaz od téhož klienta, nedává si jej do žádných souvislostí s dotazem předchozím (přestože byl třeba vyvolán některým z odkazů na stránce, kterou tomuto klientovi před chvilkou odeslal). Veškeré stavové informace (např. adresu aktuální stránky) si musí pamatovat klient. Jedna WWW transakce se skládá z dotazu a odpovědi na něj. Klient naváže transportní spojení se serverem (standardním přenosovým protokolem pro Web je TCP z rodiny TCP/IP, tento krok tedy znamená navázání TCP spojení s programem, realizujícím server) a tímto spojením odešle dotaz. Server jej zpracuje a reaguje na něj patřičnou odpovědí. Poté zpravidla transportní spojení uzavře a tím celou HTTP transakci ukončí.

Protokol HTTP prošel během několika let své existence nezanedbatelným vývojem. Jeho první verze, označená 0.9, byla velmi primitivní. Umožňovala vlastně jen syrový přenos dat ze serveru ke klientovi a v současnosti je považována za zastaralou. Dnešním standardem je verze 1.0, definovaná v RFC 1945. Jejím nejdůležitějším přínosem je možnost doprovodit dotaz i odpověď řadou doplňujících informací. V nich lze uvést typ dokumentu, dobu jeho vzniku či schopnosti a požadavky klienta. Několikaleté používání této verze však odhalilo jisté slabiny a proto vývoj pokračoval verzí 1.1.

Verze 1.1 není takovým krokem vpřed, jako svého času 1.0. Nicméně přináší několik užitečných rozšíření. Zaměřují se na zvýšení efektivity HTTP, lepší práci



Obr. 1.1: Princip činnosti protokolu HTTP.

vyrovnávacích pamětí a serverů (proxy cache) a zdokonalenou podporu neanglických jazyků.[3]

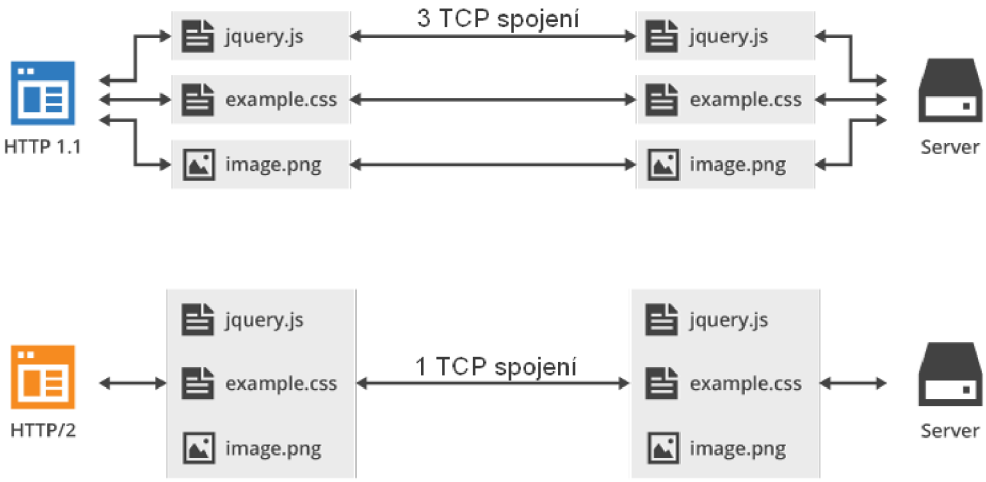
1.3.1 HTTP/2

Protokol HTTP/2 je druhá hlavní verze protokolu HTTP. Vychází z experimentálního protokolu SPDY, který byl vyvíjen pod záštitou společnosti Google. Ta oznámila ukončení jeho podpory právě ve prospěch nového protokolu HTTP/2. HTTP/2 specifikace je definována v RFC 7540 od května 2015. Protokol HTTP/2 obsahuje následující funkce či vylepšení:

- Binární protokol - Protokol HTTP/2 se od starších verzí liší například tím, že je binární. Rychleji se tedy parsuje i přenáší. Do textové reprezentace pro zobrazení v debuggerech se překládá.
- Komprese HPACK - HTTP/2 komprimuje i hlavičky a tedy i cookies. Používá na to vlastní algoritmus HPACK.
- Pipelining
- Multiplexing - HTTP/2 umí v jednom spojení na server poslat více požadavků, přičemž nezáleží na pořadí odpovědí. Jednotlivé části požadavků nebo odpovědí se mohou dokonce míchat mezi sebou.
- Server push - webový server může klientovi poslat data ještě dříve, než o ně požádá. Tím se opět zmenší počet nutných HTTP dotazů.

1.4 Webová služba

Webová služba je softwarový systém identifikovaný URI (Uniform Resource Identifier). Jeho rozhraní a vazby jsou definované a popsané pomocí XML. Definice může být objevená jinými softwarovými systémy. Tyto systémy jsou pak schopny vzájemně



Obr. 1.2: Princip multiplexingu.[9]

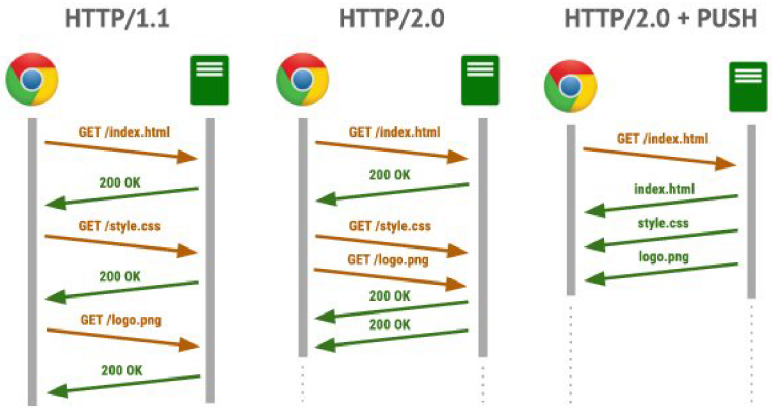
působit s webovou službou způsobem předepsaným v její definici a s použitím zpráv založených na XML a přepravovaných pomocí internetových protokolů.

Webové služby (WS - web services) představují posun od velkých monolitních struktur aplikací k modelu založenému na komponentech. Aplikace jsou v rámci tohoto modelu sestavené z malých stavebních prvků - jednotlivých funkcí. Pokud jsou tyto funkce umístěné na různých internetových serverech, označují se jako webové služby. Takto sestavené aplikace je možné snadno vytvořit, dynamicky modifikovat a měnit. Webová služba je souhrnný název pro skupinu technologií a metod, které spojují informační systémy prostřednictvím internetu a umožňují jim spolu efektivně komunikovat.[2]

Existují dva základní druhy webových služeb: klasické XML služby a RESTful služby.

1.4.1 SOAP

Vzájemná komunikace SOAP webových služeb je založená na dodržení tří základních standardů. Jsou jimi normy pro opis webových služeb WSDL, protokol vzájemné komunikace objektů SOAP založený na textovém formátu XML a seznam webových služeb UDDI.



Obr. 1.3: Princip funkce server push.[8]

1.4.2 REST

REST (Representational State Transfer) je architektura rozhraní, navržená pro distribuované prostředí. Pojem REST byl představen v roce 2000 v disertační práci Roye Fieldinga, jednoho ze spoluautorů protokolu HTTP. Proto nepřekvapí, že REST má s HTTP hodně společného.

REST je, na rozdíl od SOAP, orientován datově, nikoli procedurálně. Webové služby definují vzdálené procedury a protokol pro jejich volání, REST určuje, jak se přistupuje k datům.

Rozhraní REST je použitelné pro jednotný a snadný přístup ke zdrojům (resources). Zdrojem mohou být data, stejně jako stavy aplikace (pokud je lze popsat konkrétními daty). Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim, které jsou známé pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu.[10]

1.5 SSL

Secure Sockets Layer je protokol nebo spíše vrstva mezi vrstvou transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran.

Ustavení SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů – veřejný a soukromý. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem.

Během první fáze ustanovení bezpečného spojení si klient a server dohodnou kryptografické algoritmy, které budou použity.

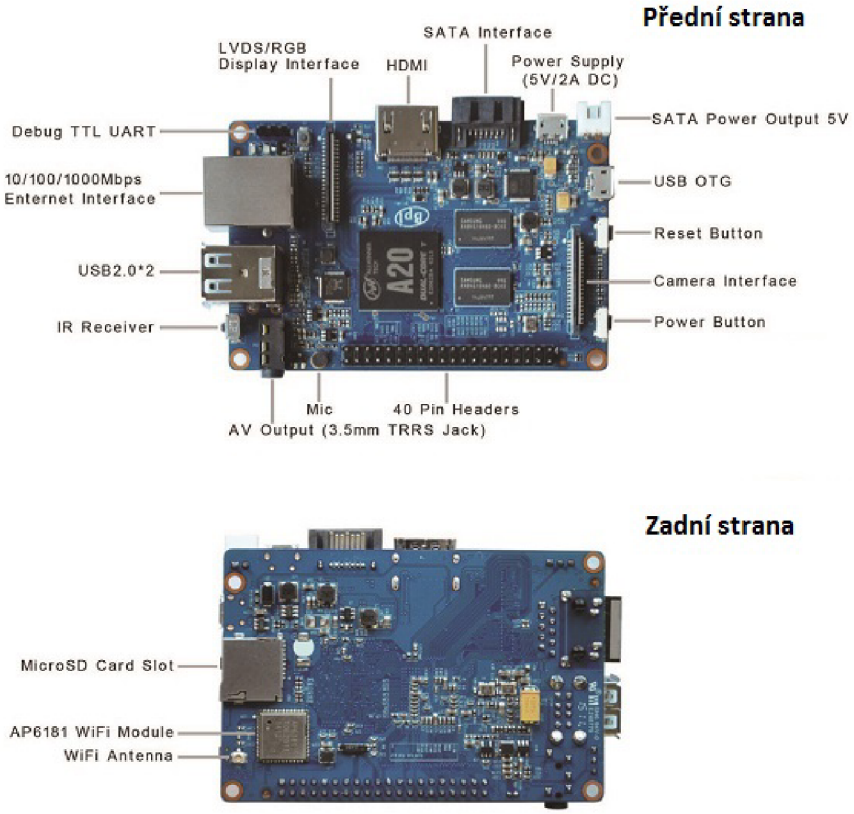
2 POUŽITÉ TECHNOLOGIE

2.1 Banana Pi Pro

Banana Pi Pro je jednodeskový počítač. Velice se podobá známějšímu a populárnějšímu Raspberry Pi. Mají podobnou architekturu, avšak Banana Pi Pro je mnohem výkonnější. Banana Pi Pro je vylepšeným modelem Banana Pi. Základ počítače tvoří dvoujádrový procesor 1 GHz ARM Cortex-A7 Dual Core s integrovaným GPU jádrem Mali400MP2 a obsahuje integrovaná operační paměť 1 GB DDR3. Deska neobsahuje žádnou interní paměť pro operační systém nebo ukládání souborů, naštěstí však nabízí možnost připojení microSD karty či SATA rozhraní pro připojení pevného disku. K připojení zobrazovací jednotky slouží konektor HDMI nebo kompozitní RCA. Zvukový výstup lze napojit skrze 3,5 mm JACK nebo HDMI. Na desce se nachází také integrovaný mikrofon. Oproti Raspberry Pi je součástí Banana Pi Pro ethernet port 10/100/1000 s konektorem RJ45 a na rozdíl od Banana Pi obsahuje bezdrátový modul WiFi s podporou standardu B/G/N. Základní deska obsahuje dva konektory USB 2.0 včetně jednoho konektoru USB OTG a jednoho konektoru USB micro sloužící pro napájení.

Celý počítač má rozměry 90 mm x 60 mm. Najdeme zde i sadu 40 programovatelných pinů pro připojení rozšiřujících modulů a konektor na připojení LCD dotykového panelu. Součástí desky jsou i tři tlačítka - zapnutí, reset a bootování. Deska obsahuje IR přijímač pro dálkové ovládání.

Základní deska počítače Banana Pi Pro je napájena pomocí napájecího adaptéru 5V s doporučeným výstupním proudem 2A.



Obr. 2.1: Banana Pi Pro.

2.2 Operační systém Raspbian

Raspbian je open source operační systém, založený na Debianu, který je optimalizován pro platformy jako Raspberry Pi či Banana Pi apod. Operační systém je sada základních programů a nástrojů, díky kterým naše platforma Banana Pro funguje. Raspbian poskytuje více než jen samotný operační systém, je dodáván s více než 35.000 balíčky, což představuje předkompilovaný software, zabalený ve formátu pro snadnou instalaci.[4]

2.3 Nginx

Nginx je open-source softwarový webový server a reverzní proxy, napsaný ruským programátorem Igorem Sysoevem. Pracuje s protokoly HTTP (i HTTPS), SMTP, POP3, IMAP a SSL. Zaměřuje se především na vysoký výkon, vysokou souběžnost

připojení a nízké nároky na paměť. Nginx je dostupný na Unixu, Linuxu a dalších variantách UNIXU pod BSD. V dnešní době existují i varianty pro Solaris, MAC OS a Microsoft Windows. Nginx je momentálně třetí nejpoužívanější webový server[1].

2.4 Apache

Patří mezi nejpopulárnější webové servery. Dle průzkumu[1] z listopadu 2015 zastupuje Apache celkových 37 % webových serverů a je tak stále na prvním místě. Jeho hlavní přednost je podpora velkého množství operačních sytému (GNU/Linux, BSD, Solaris, Mac OS X, Microsoft Windows a další). Podporuje velké množství funkcí, které mohou být rozšířeny pomocí modulů. Nejběžnějšími rozšířeními jsou moduly pro Perl, Python, Tcl či PHP. Podporuje také šifrování (SSL/TLS), proxy a zápisy do logů. Pokud jde o rychlost tohoto webového serveru, tak používá tzv. MultiProcessing moduly a může se tedy přizpůsobit potřebám systému, na kterém je spuštěn. Apache web server je šířen pod licencí Apache License 2.0, která stanoví, že tento program je volně šiřitelný a má slučitelnou licenci s třetí verzí GNU General Public License.[5]

2.5 JavaScript

JavaScript je programovací jazyk, který se používá v internetových stránkách. Zapisuje se přímo do HTML kódu, což je velká výhoda, protože je to jednoduché.

JavaScript je klientský skript. To znamená, že se program odesílá se stránkou na klienta (do prohlížeče) a teprve tam je vykonáván. (Protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta jdou už jen výsledky.)

2.6 JSONP

Javascript z bezpečnostních důvodů umí přímo načítat data jen z domény, ze které pochází zobrazovaná stránka. Celé skripty však může stránka z načítat i z jiné domény. Toho JSONP využívá. JSONP je jedním ze způsobů jak komunikovat asynchronně se serverem na pozadí klientské aplikace. Takové to aplikace se obecně označují za AJAX. JSONP představuje alternativu k použití XmlHttpRequest objektu pro vlastní asynchronní komunikaci. JSONP je zkratka z JSON in Padding. Česky by se to dalo přeložit nejspíš jako JSON v závorce. JSONP není všeobecně

[1] Dle průzkumu společnosti Netcraft

zavedená zkratka. Například společnost Google tuto metodu ve svých API označuje za json-in-script, což vcelku vystihuje podstatu této metody.[7]

2.7 jQuery

jQuery je javascriptová knihovna, která má za úkol usnadnit práci s javascriptem. Má podporu všech moderních prohlížečů, tudíž je to ideální univerzální nástroj na psaní webových aplikací pro klientskou stranu.

2.8 Node.js

Node.js je platforma pro vývoj webových aplikací v jazyce Javascript. Pojem platforma zahrnuje i webový server, není tedy nezbytně nutné žádný zvláštní webový server (jako je třeba Apache pro PHP) k Node.js pro provoz instalovat, nicméně v produkčním nasazení je lepší Node.js používat společně s jiným webovým serverem, který zajistí minimálně vyřizování požadavků na statické soubory, které by neměly být zpracovány přes Node.js. Node.js pracuje pouze v jednom vlákně tzv. single-thread, což je rozdíl oproti častějším případům, kdy webový server odpověď na každý požadavek zpracovává v jiném vlákně. Dále Node.js používá událostmi řízený (event-driven) neblokující I/O model. Všechny I/O operace v Node.js jsou napsány tak, aby server neblokovaly.[6]

2.9 ApacheBench

Server Apache obsahuje benchmark nástroj známý jako AB. Nástroj odesílá požadavky na server Apache a poskytuje statistické údaje o době odezvy serveru. Použitím toho nástroje můžeme vidět reakci serveru pod zatížením. AB odesílá uživatelsky nakonfigurované množství žádostí, včetně možností nastavení, kolik se těchto žádostí bude posílat zároveň. Kromě těchto dvou základních nastavení, poskytuje AB i další možnosti nastavení.

2.10 Chrome DevTool

Chrome DevTool je soubor ladících nástrojů vestavěných do internetového prohlížeče Chrome, které poskytují webovým vývojářům hloubkovou analýzu interních nastavení prohlížeče a webové aplikace.

3 PŘÍPRAVA PROSTŘEDÍ

V této kapitole jsou popsány jednotlivé kroky, jak byly jednotlivé webové servery a služby nakonfigurovány, včetně základního nastavení Banana Pi Pro.

3.1 Instalace OS Raspbian

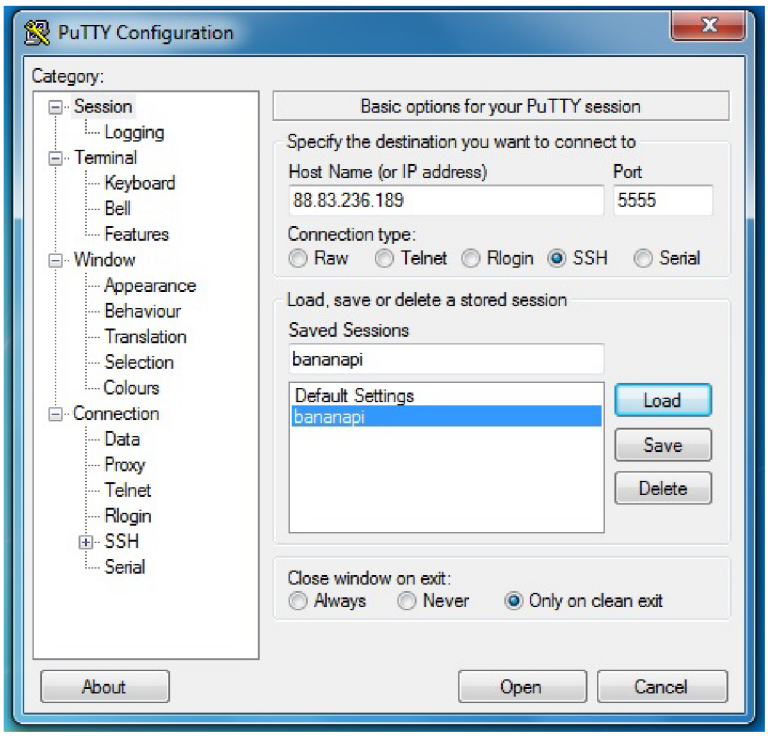
Pro tuto práci jsem si zvolil jako úložiště SD kartu - typ micro SDHC. Pro instalaci samotného systému je důležitá minimální velikost karty 2GB. Taky je důležité zvolit si kartu s dostatečně rychlým zápisem i čtením. V dnešní době je takovým standardem karta s rychlostí třídy 10. Po zvolení správné SD karty si stáhneme volnou distribuci námi použitého systému, Raspbianu, ze stránek www.bananapi.com, kde se nachází verze upravená pro potřeby naší platformy Banana Pi Pro ve formátu .img. Jelikož přípravu SD karty jsem prováděl v prostředí Windows 10, využil jsem program Win32DiskImager, který provede jak správné naformátování SD karty, tak instalaci OS z námi staženého souboru.

V programu zvolíme správný obraz a zvolíme SD kartu. Poté klikneme na tlačítko "Write", program nás upozorní na ztrátu dat, naformátuje kartu a provede zápis dat. Po dokončení zápisu se na kartě vytvořili dva oddíly. Jeden bohužel nevidíme, protože je naformátován jako ext4, což je linuxový souborový systém. Tím je příprava micro SD karty dokončena a systém je už připraven ke spuštění.

Nyní vložíme kartu do micro SD slotu na Banana Pi Pro. Připojíme klávesnici, myš, monitor a napájení. Počítač začne sám od sebe bootovat z SD karty. Po načtení systému vidíme plochu Raspbianu s předinstalovanými aplikacemi. Abychom mohli servery lépe testovat, nastavíme port pro SSH přístup (současný port 22 změníme na nějaký jiný, nepoužívaný port, např. 5555) a SSH restartujeme:

```
sudo nano /etc/ssh/sshd_config
/etc/init.d/ssh restart
```

Díky tomuto nastavení můžeme mít počítač připojený pouze k routeru a napájení a přistupovat k němu vzdáleně přes SSH pomocí programu PuTTY. Nasteví PuTTY je znázorněno na obrázku 3.1.



Obr. 3.1: Nastavení PuTTY.

3.2 Instalace a konfigurace serveru

Pro tuto práci jsem si zvolil nejčastěji používaný server Apache a třetí nejpoužívanější a taky volně dostupný server Nginx, který má zároveň oficiální podporu HTTP/2, tudíž jsem ho použil pro otestování HTTP/2.

3.2.1 Instalace a konfigurace Nginx serveru s HTTP/2

1. Nejdříve je nutné nainstalovat samotný Nginx:

```
apt-get install nginx
```

2. Nesmíme zapomenout, že Nginx server vytváříme s podporou HTTP/2, kde je zapotřebí vytvořit SSL certifikát:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
-keyout /etc/nginx/ssl/certificate.key
-out /etc/nginx/ssl/certificate.crt
```

3. Po nainstalování serveru a vytvoření SSL certifikátu se přesuneme do složky `sites-available`, kde vytvoříme nový konfigurační soubor pro Nginx:

```
1  //nazev serveru, vychozi slozky, port na kterem nasloucha a pridana
       podpora http2 a ssl
2  server{
3      root /var/www/;
4      listen 443 ssl http2;
5      server_name 192.168.0.107;
6      index index.html index.htm;
7
8
9  // pridani ssl certifikatu
10     ssl on;
11     ssl_certificate         /etc/nginx/ssl/certificate.crt;
12     ssl_certificate_key     /etc/nginx/ssl/certificate.key;
13 }
```

Listing 3.1: Konfigurační soubor pro Nginx server s HTTP/2

4. Poté ve složce `sites-enabled` vytvoříme pomocí příkazu:

```
ln -s /etc/nginx/sites-available/nazev_konfig_souboru
```

link, na vytvořený konfigurační soubor Nginx serveru. Díky tomuto linku budeme zde mít vždy stejný soubor, jako je ve složce `sites-available`. Tímto jsme si povolili virtuální server na daném portu.

5. Nyní máme server nakonfigurován a můžeme ho spustit:

```
service nginx start
```

Nginx server je spuštěný a naslouchá na portu 443.

3.2.2 Konfigurace Nginx serveru bez podpory HTTP/2

Pro Nginx server bez podpory HTTP/2 se musel vytvořit nový index soubor, kde skript posílá dotaz na druhou spuštěnou webovou službu(Node.js, port 8080), která je spuštěna bez podpory HTTP/2. Jinak vše proběhlo stejně jako v předchozí části.

```
1
2 server{
3     root /var/www/;
4     listen 8887;
5     server_name 192.168.0.107;
6     index indexbez.html indexbez.htm;
7
8 }
```

Listing 3.2: Konfigurační soubor pro Nginx server bez HTTP/2

3.2.3 Konfigurace Nginx jako reverzního proxy serveru s podporou HTTP/2

Nyní si vytvoříme reverzní proxy server, který nám žádosti typu GET přeposílá rovnou na webovou službu. Tudíž není zapotřebí žádný dotazovací skript, který by nám tyto žádosti posílal.

```
1
2 server{
3     listen 8886 ssl http2;
4     server_name 192.168.0.107;
5
6     ssl on;
7     ssl_certificate         /etc/nginx/ssl/certificate.crt;
8     ssl_certificate_key     /etc/nginx/ssl/certificate.key;
9
10    location / {
11    proxy_set_header Host             $host;
12    proxy_set_header X-Real-IP        $remote_addr;
13    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
14    proxy_pass "http://192.168.0.107:8888";
15
16    }
17
18 }
```

Listing 3.3: Konfig. soubor pro Nginx reverzní proxy server s HTTP/2

3.2.4 Konfigurace Nginx jako reverzního proxy serveru bez podpory HTTP/2

Podobně jako v předchozí konfiguraci si vytvoříme reverzní proxy server, jen s tou změnou, že vynecháme podporu HTTP/2.

```
1
2  server{
3      listen 8899;
4      server_name 192.168.0.107;
5
6      location / {
7      proxy_set_header Host             $host;
8      proxy_set_header X-Real-IP        $remote_addr;
9      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
10     proxy_pass "http://192.168.0.107:8080";
11
12     }
13
14 }
```

Listing 3.4: Konfig. soubor pro Nginx reverzní proxy server bez HTTP/2

3.2.5 Instalace a konfigurace Apache serveru bez podpory HTTP/2

1. Nainstalujeme Apache server:

```
apt-get install apache2 -y
```

2. Přesuneme se do složky `/etc/apache2/sites-available`, otevřeme konfigurační soubor `default` a nastavíme Apache server:

```
1 //nazev serveru, port
2 <VirtualHost *:8889>
3
4
5     ServerName 192.168.0.107
6     DocumentRoot /var/www/
7
8 </VirtualHost>
```

Listing 3.5: Konfigurační soubor pro Apache server

3. Server je připravený ke spuštění, proto ho restartujeme a tím dojde k jeho spuštění a nahrání nového konfiguračního souboru:

```
service apache2 restart
```

Apache server je spuštěný a naslouchá na portu 8889.

3.3 Vytvoření dotazovacího skriptu na web. službu

Ve složce `/var/www/` vytvoříme soubor index.html, který bude obsahovat dotazovací skript na webovou službu. Tento skript je napsán v jazyce Javascript s využitím knihovny jQuery. Protože jQuery má jistá bezpečnostní omezení, která nedovolí posílat žádosti na jiné domény, než figuruje sám skript, je nutné použít JSONP, který nám umožní tzv. cross-domain žádost.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Bakalarska prace</title>
5  </head>
6  <body>
7  <h1>Test</h1>
8  <script language="javascript" type="text/javascript" src="jquery.min.js">
9  </script>
10 <script>
11 function dotaznasluzbu() {
12  $.ajax({
13        type: 'GET',
14        url:'https://88.83.236.189:8888',
15        async: true,
16        dataType: "jsonp",
17        jsonCallback: "_testcb",
18        cache: false,
19        timeout: 1000,
20        success: function(data) {
21                $("#test").append(data);
22       },
23   });
24 }
25
26 for (i=0; i<10; i++){
27   dotaznasluzbu();
28 }
```

```
29 </script>
30 <span id = "test">Data</span>
31 </body>
32 </html>
```

Listing 3.6: Soubor index.html s dotazovacím skriptem

Z kódu je patrné, že používáme jazyk HTML a Javascript. Vytvořili jsme funkci `dotaznasluzbu`, která posílá požadavky typu GET na adresu, kde máme vystavenou webovou službu pomocí Node.js, pomocí JSONP. Odpověď se nám vrací v proměnné `_testcb`. Jestli funkce proběhla v pořádku, pak se nám pomocí proměnné `test` vypíšou vrácené data na plochu. Pro větší zatížení služby jsme použili cyklus `for`, který vykoná několik dotazů za sebou.

Pro servery, které se budou dotazovat bez použití HTTP/2, vytvoříme obdobný soubor indexbez.html, ve kterém se akorát budeme dotazovat na adresu, kde je vystavena webová služba bez použití HTTP/2.

3.4 Instalace Node.js a vytvoření webové služby

Dalším krokem bude nainstalovat Node.js a vytvořit webovou službu, která bude naslouchat na portu 8888.

1. Nejdříve si nainstalujeme Node Version Manager, který slouží ke správě instalací a všech verzí Node.js:

   ```
   curl https://raw.githubusercontent.com/creationix/
   nvm/v0.20.0/install.sh | bash
   ```

2. Zavřeme a znovu otevřeme terminál.
3. Nainstalujeme Node.js:

   ```
   nvm install 0.10
   ```

4. Přidáme podporu HTTP/2:

   ```
   npm install http2
   ```

   SSL certifikát již nemusíme vytvářet, použijeme ten, co jsme si vytvořili pro náš server Nginx.
5. Ve složce `/var/www/` vytvoříme webovou službu pomocí Node.js, která bude čekat na dotazy typu GET na portu 8888:

```
1 //Node.js webovy server
2 var fs = require('fs');
3 var options = {
```

```
4  key: fs.readFileSync('/etc/nginx/ssl/certificate.key'),
5  cert: fs.readFileSync('/etc/nginx/ssl/certificate.crt')
6  };
7
8  var http = require('http2');
9
10 http.createServer(options, function(request, response) {
11   if (request.method == 'GET') {
12     console.log('dotaz typu GET obdrzen');
13     request.on('data', function(chunk) {
14
15     }).on('end', function() {
16     response.writeHead(200, {'Content-Type': 'text/plain'});
17     response.end('_testcb(/'Text, ktery se vraci jako odpoved na dotaz
           typu GET/')');
18     })
19     } else {
20     response.statusCode = 404;
21     response.end();
22   }
23 }).listen(8888, "192.168.0.107");
24 console.log("Server naslouchа na portu 8888.") //vypis do terminalu
```

Listing 3.7: Konfigurační soubor pro Node.js webovou službu

6. Spustíme nové sezení:

```
screen
```

7. Stiskneme `Enter` a spustíme Node.js server:

```
node server.js
```

8. Sezení ukončíme stisknutím `Ctrl+a+d`

Tímto jsme na adresu https://192.168.0.107:8888/ vystavili webovou službu za pomocí Node.js, která naslouchá na portu 8888 a čeká na příchozí dotazy typu GET. Na tyto dotazy odpovídá testovacím textem. Do konzole si necháváme vypisovat stav webové služby, abychom věděli, zda služba funguje a jestli obdržela příchozí GET dotazy.

Nesmíme však zapomenout, že pro servery, které jsou nakonfigurovány bez podpory HTTP/2, musíme tuto službu vytvořit ještě jednou, taky bez podpory HTTP/2. To provedeme pouze odstraněním částí kódu, které se týkají HTTP/2 a zvolíme si port 8080, na kterém nám tato služba bude naslouchat.

4 POROVNÁVÁNÍ SERVERŮ A VÝSLEDKY

Nyní máme nakonfigurovaných 5 serverů:

- Nginx s podporou HTTP/2
- Nginx bez podpory HTTP/2
- Nginx jako reverzní proxy server s podporou HTTP/2
- Nginx jako reverzní proxy server bez podpory HTTP/2
- Apache bez podpory HTTP/2

Pro otestování serverů a webové služby je zapotřebí zvolit si správné testovací nástroje a metodiku testování. U našeho testování nás bude zajímat `Time per request` neboli průměrná doba k vyřízení požadavku. To je doba od odeslání žádosti typu GET po po vrácení odpovědi od webové služby. Pro všechny servery jsem zvolil zatížení 10 požadavky z jednoho připojení. To by nám mělo poskytnout dostačující průměr pro porovnání všech serverů.

Jako testovací nástroje jsem si vybral ApacheBench (AB) a Chrome DevTools. První zmíněný slouží k otestování reverzních proxy serverů, protože musíme vytvořit námi zvolený počet požadavků bez využití dotazovacího skriptu, jelikož reverzní proxy server přijaté žádosti přeposílá na námi určenou adresu, v tomto případě adresu, kde je vystavena webová služba. Druhý nástroj zachytává veškerou komunikaci a poskytuje nám detailní informace, proto nám poslouží u ostatních serverů.

4.1 Testování pomocí nástroje ApacheBench

Než začneme testovat s pomocí nástroje ApacheBench, je nutné si tento nástroj nainstalovat:

```
aptitude install apache2-utils
```

Nástroj AB má dva klíčové parametry, a sice celkový počet HTTP žádostí `-n` a počet současně posílaných žádostí `-c`. Výchozí hodnotou je u obojího jednička. Kromě těchto parametrů disponuje AB i dalšími parametry, které však v této práci nebyly využity.

Pokud tedy chceme otestovat webovou službu zasláním 10 žádostí z pouze jednoho připojení, vykonáme bychom následující příkaz:

```
ab -n 10 -c 1 https://88.83.236.189:8886/
```

Z výsledků benchmarku, který je na obrázku A.5 lze vyčíst spoustu informací. Vidíme, že byl opravdu využit Nginx s podporou HTTP/2, na jakou adresu jsme se dotazovali, i použití SLL protokolu. Dále se zde můžeme dočíst například, jak dlouho test probíhal, kolik bylo přeneseno dat, kolik požadavků bylo vyřízeno za

sekundu apod. Nás však zajímá hodnota u `Time per request`, kterou si zapíšeme do výsledné tabulky. Ten samý příkaz vykonáme i na adresu, kde nám naslouchá reverzní proxy server bez podpory HTTP/2.

```
ab -n 10 -c 1 https://88.83.236.189:8899/
```

Výsledek vidíme na obrázku A.4.

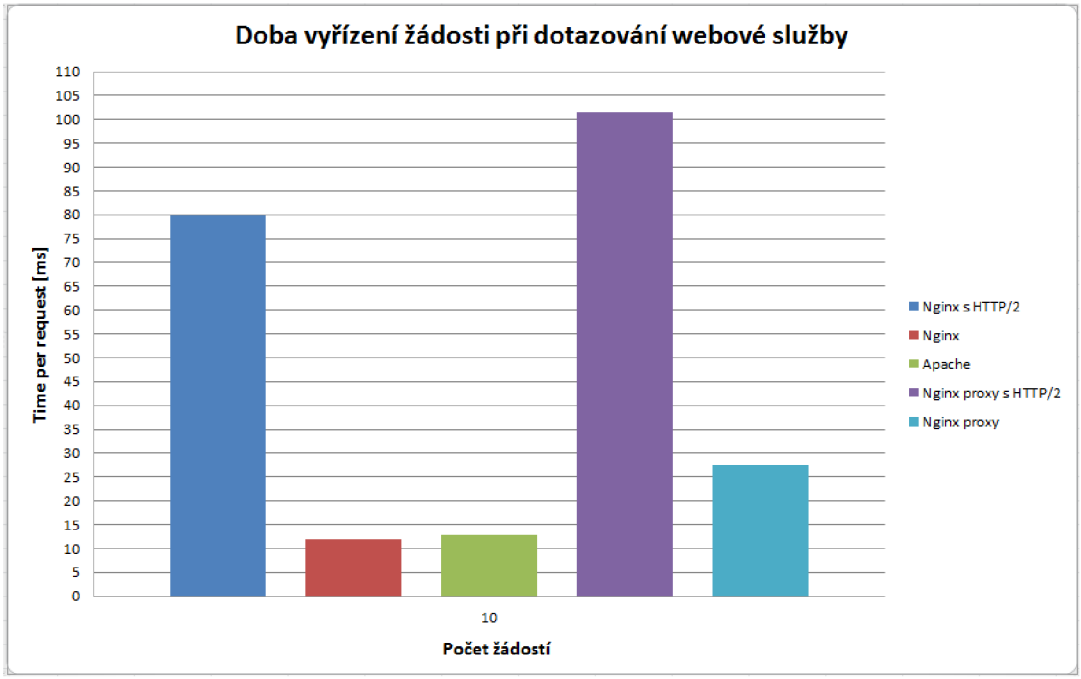
4.2 Testování pomocí nástroje Chrome DevTool

Tento nástroj je součástí prohlížeče Chrome, takže pokud ho již nemáme, musíme si ho nainstalovat. Spuštění toho nástroje se provádí pomocí klávesové zkratky `F12`, kdy se nám v pravé části prohlížeče zobrazí námi požadovaný nástroj. Podle obrázku A.3 vidíme, že po zadání adresy našeho požadovaného serveru do adresní řádky prohlížeče, dostaneme pěknou analýzu všech provedených žádostí, které vidíme v záložce Network. Nás nejvíce zajímá hodnota `Time` při volání webové služby `?callback=_testcb&_`, která znázorňuje dobu k vyřízení požadavku. Všech 10 hodnot si zapíšeme a uděláme průměr. Stejně pokračujeme i u ostatních serverů. Když se ještě jednou zaměříme na obrázek A.3 a A.2, tak si můžeme všimnout rozdílu ve velikosti u odpovědi webové služby. To je způsobené právě binárním přenosem, který podporuje HTTP/2. Dále je ještě patrná jedna funkce HTTP/2 a tou je jedno TCP spojení, přes které šly všechny odpovědi. Lze to zaznamenat ve sloupci connection id, hned vedle sloupce time.

4.3 Výsledky testování

Všechny hodnoty `Time per request` naměřené u jednotlivých testovaných serverů jsme si zapsali a vynesli do grafu 4.1, ze kterého je patrné, jak které servery dopadly. Nejlépe si vedl Nginx server bez podpory HTTP/2 a těsně za ním se umístil Apache server taky bez podpory HTTP/2. Naopak servery, které podporovaly HTTP/2 dopadly nejhůře. Podle mě je to způsobeno použitím SSL certifikátů, protože se musí nejprve ustavit SSL spojení a až pak se začné posílat žádost typu GET. U reverse proxy serverů taky zaznamenáváme větší zpoždění oproti ostatním serverům. To je pravděpodobně způsobeno tvorbou nové HTTP hlavičky.

Určitě jiným nastavením serverů a použití jiné metodiky testování by se dosáhlo jiných výsledků. Taky je důležité upozornit, že jsme testovali dotazování se na REST webovou službu, která byla vystavena na jiném portu, než samotný server.



Obr. 4.1: Výsledky testování.

5 ZÁVĚR

Cílem bakalářské práce bylo seznámit se alespoň s dvěma open source webovými servery, přičemž minimálně jeden bude podporovat HTTP/2. Tyto servery posléze nasadit na Banana Pi Pro, přičemž budou schopny provádět žádosti na webovou službu, která bude vystavena pomocí Node.js. Poté tyto servery otestovat na rychlost provádění žádosti na webovou službu.

Veškeré cíle bakalářské práce byly úspěšně splněny.

Pro práci jsem si vybral dva open source softwarové servery, Apache a Nginx. Vybrané servery jsem nastudoval a úspěšně nasadil na platformu Banana Pi Pro, kdy jsem vytvořil celkem 5 virtuálních serverů. Tyto servery jsem pomocí ApacheBench benchmarku a Chrome DevTools otestoval při dotazování se na webovou službu, která se mi povedla vystavit za pomocí Node.js. na který se dotazovali za pomocí skriptu, na který jsem použil nastudované poznatky o Javascriptu, JSONP a jQuery. Nejlépe dopadl Nginx bez použití HTTP/2. Nejhůř Nginx reverzní proxy server s využitím HTTP/2. To je pravděpodobně způsobeno použitím SSL certifikátu. Při psaní semestrální práce jsem se pokoušel vytvořit html stránku, ze které bych se dotazoval pomocí javascriptu na webovou službu. Bohužel se mi to nepodařilo a rád bych tuto funkci zprovoznil do bakalářské práce. Dále bych se rád zabýval v bakalařské práci testováním více serverů a jejich podrobnějším porovnáváním.

LITERATURA

[1] PINKAS, Otakar. Webové servery. *Ikaros* [online]. 2011, [cit. 2015-12-04]. Dostupné z URL: <http://ikaros.cz/webove-servery>.

[2] HOROVČÁK, Pavel. Webové služby - třetí generace internetu. *SystemOnLine.cz* [online]. 2003, [cit. 2015-12-04]. Dostupné z URL: <http://www.systemonline.cz/clanky/webove-sluzby-treti-generace-internetu.htm>.

[3] SATRAPA, Pavel. HTML v příkladech. *Softwarové noviny* [online]. 1997, [cit. 2015-12-04]. Dostupné z URL: <http://www.nti.tul.cz/~satrapa/docs/wwwprikl/html9.html>.

[4] *Raspbian* [online]. 2015, [cit. 2015-12-04]. Dostupné z URL: <https://www.raspbian.org/>.

[5] About Apache. *Apache HTTP Server Project* [online]. [cit. 2015-12-04]. Dostupné z URL: <http://httpd.apache.org/ABOUT_APACHE.html>.

[6] NEŠETŘIL, Jakub. JavaScript na serveru: Začínáme s Node.js. *Zdroják.cz* [online]. 2010, [cit. 2015-12-04]. Dostupné z URL: <https://www.zdrojak.cz/clanky/javascript-na-serveru-zaciname-s-node-js/>.

[7] HOŘČICA, Adam. Co je to JSONP? *Adam's Notepad* [online]. 2010, [cit. 2016-05-27]. Dostupné z URL: <http://notepad.jslab.net/clanky/co-je-to-jsonp.html>.

[8] CIOTTA, Giuseppe. Getting to know HTTP/2.0 with the mosaic demo. *Giuseppe Ciotta Blog* [online]. 2015, [cit. 2016-05-27]. Dostupné z URL: <https://giuseppeciotta.net/getting-to-know-http20-with-the-mosaic-demo.html>.

[9] HODSON, Ryan. HTTP/2 For Web Developers. *CloudFlare* [online]. 2015, [cit. 2016-05-27]. Dostupné z URL: <https://blog.cloudflare.com/http-2-for-web-developers/>.

[10] MALÝ, Martin. REST: architektura pro webové API. *Zdroják.cz* [online]. 2009, [cit. 2016-05-27]. Dostupné z URL: <https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>.

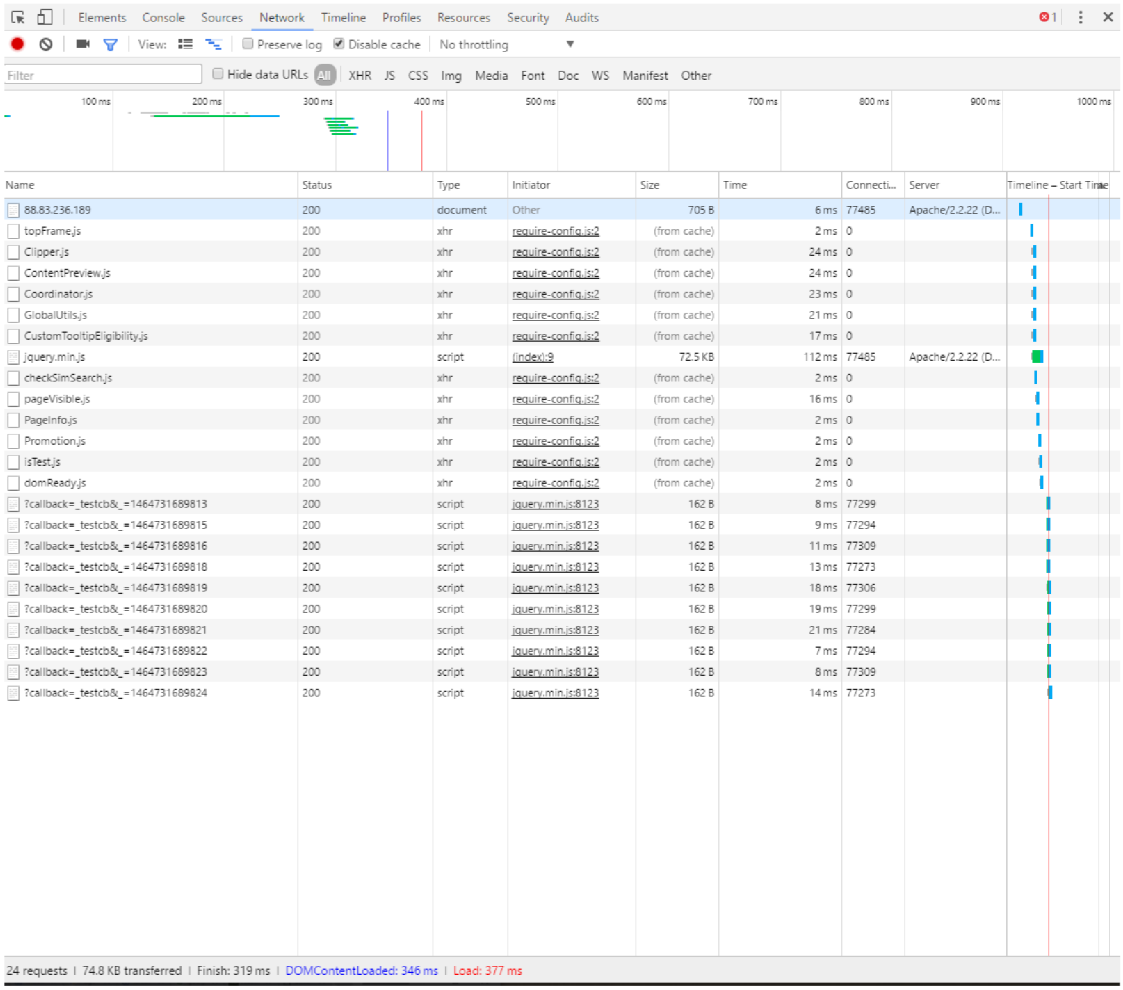
[11] PROKEŠOVÁ, Hana. Proxy server. [online]. 2003, [cit. 2016-05-27]. Dostupné z URL:
<https://www.fi.muni.cz/~kas/p090/referaty/2003-jaro/skupina10/proxy.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

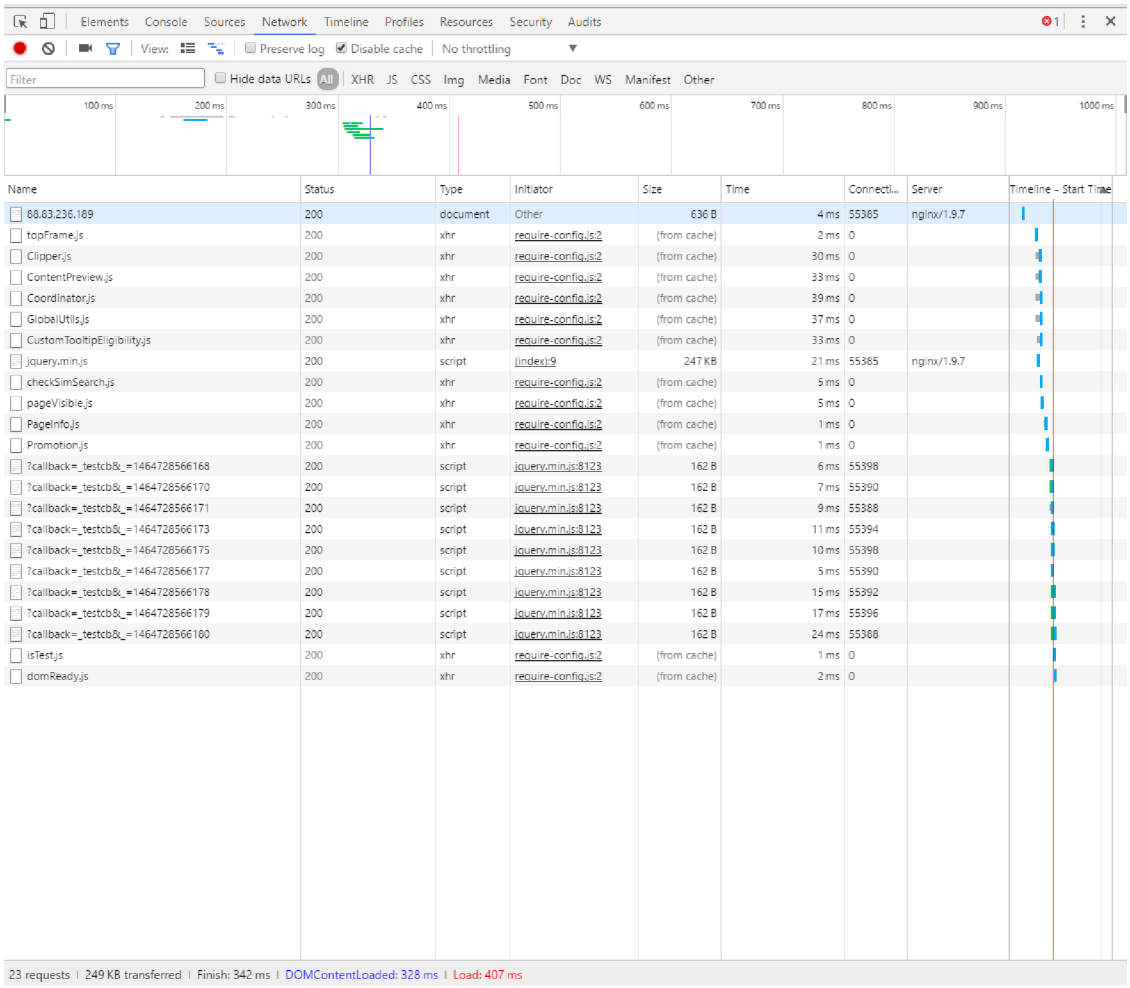
HTTP	HyperText Transfer Protocol
URL	Uniform Resource Locator
NAT	Network Address Translation
SPDY	pronounced speedy
URI	Uniform Resource Identifier
XML	Extensible Markup Language
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
WSDL	Web Services Description Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
JSONP	JavaScript Object Notation with Padding
SSH	Secure Shell
REST	Representational State Transfer

SEZNAM PŘÍLOH

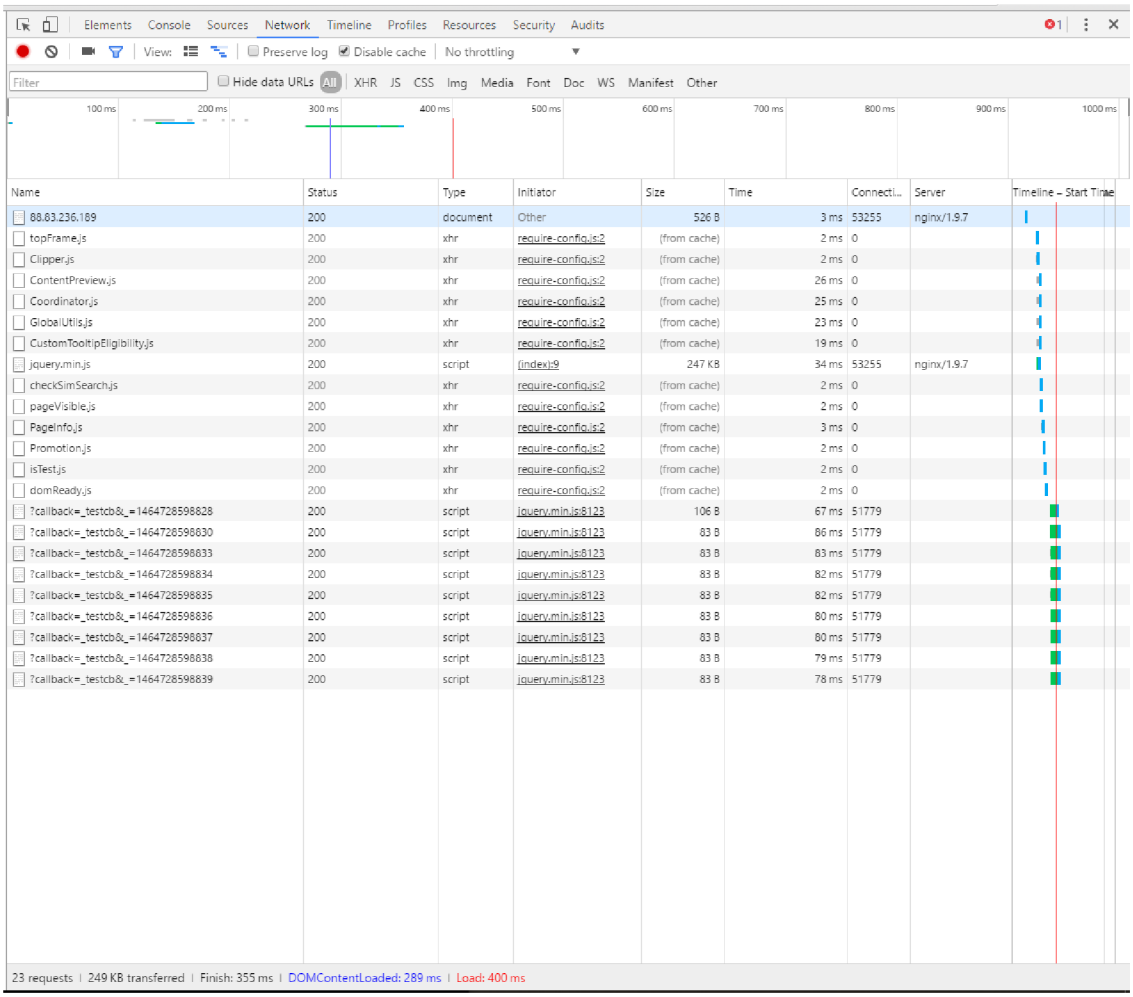
A SCREENSHOTY Z TESTOVÁNÍ SERVERŮ S VYSTAVENOU WEBOVOU SLUŽBOU



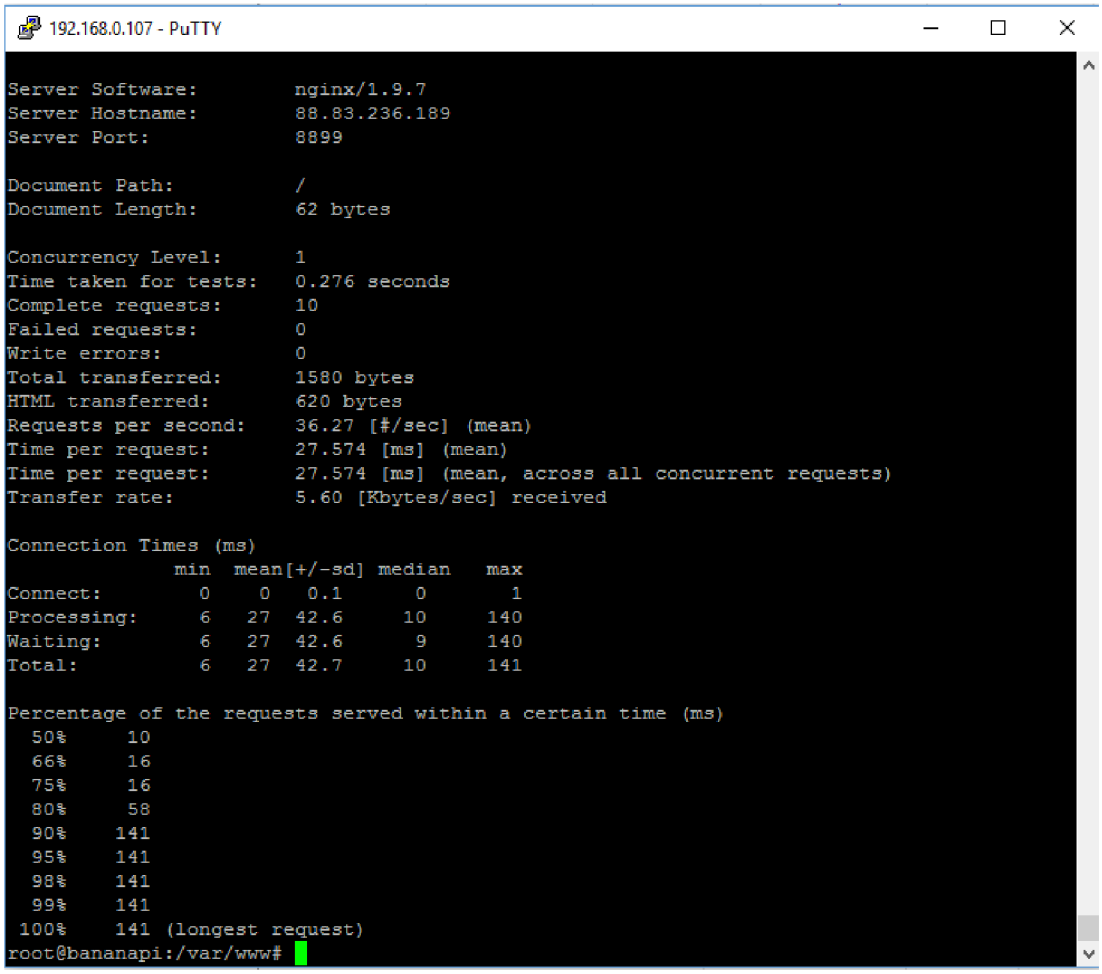
Obr. A.1: Testování Apache serveru pomocí Chrome DevTools.



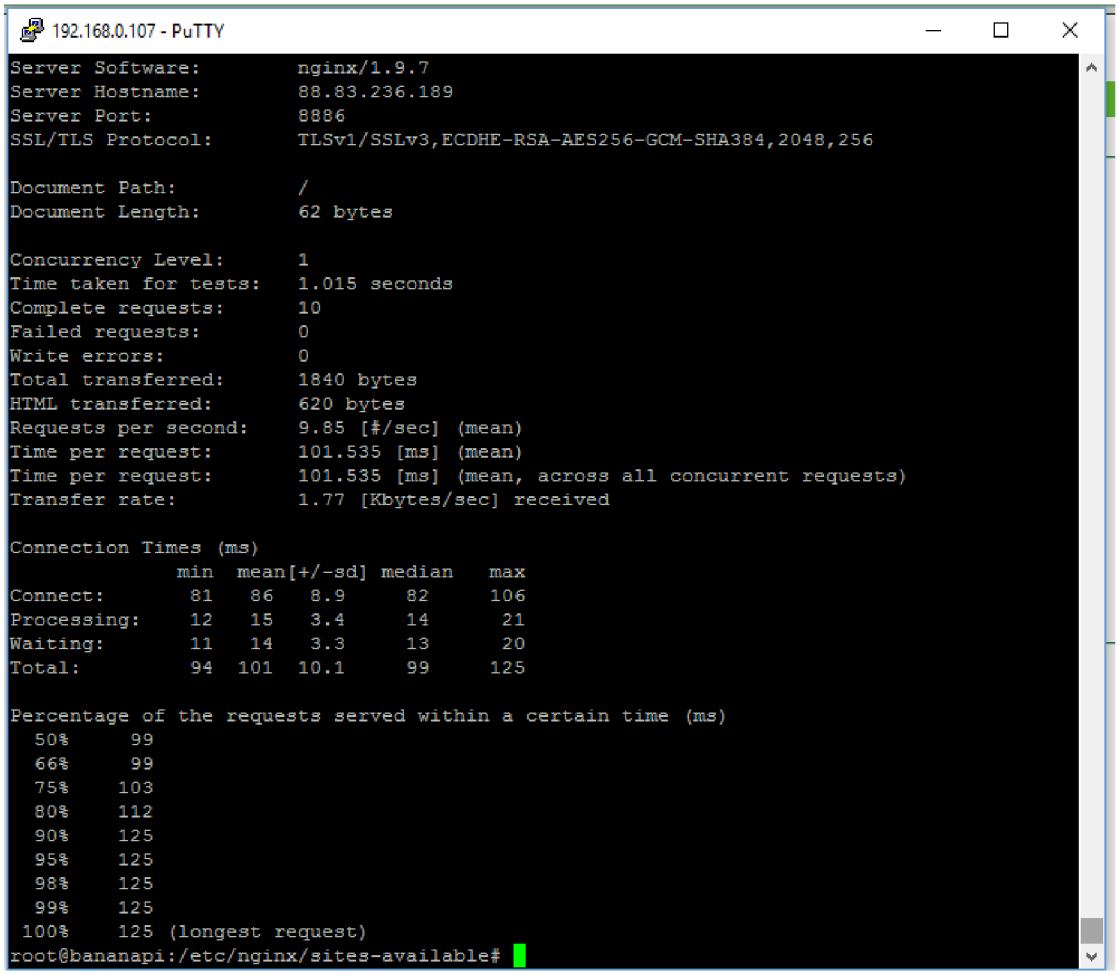
Obr. A.2: Testování Nginx serveru pomocí Chrome DevTools.



Obr. A.3: Testování Nginx serveru s podporou HTTP/2 pomocí Chrome DevTools.



Obr. A.4: Testování reverzního proxy serveru Nginx pomocí nástroje Apache benchmark.



Obr. A.5: Testování reverzního proxy serveru Nginx s podporou HTTP/2 pomocí nástroje Apache benchmark.

B OBSAH PŘILOŽENÉHO CD

- Elektronická verze bakalářské práce - xvalka00.pdf
- Zdrojové kódy webové služby - ve složce `Webová služba`
- Zdrojové kódy dotazovacího skriptu - ve složce `Dotazovací skript`
- Konfigurační soubory jednotlivých serverů - ve složce `Servery`