Individuální projekt

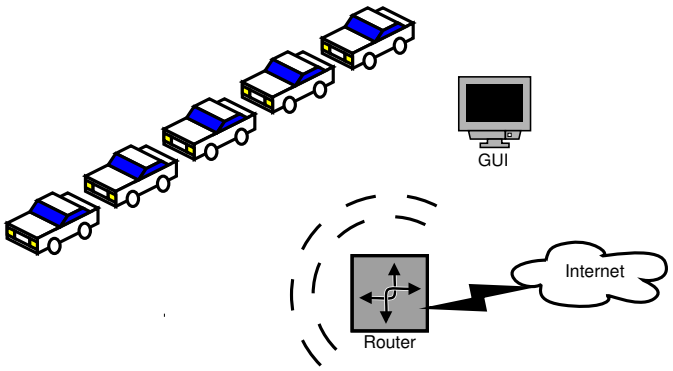
Šimon Wernisch

I. Úvod

V řízených systémech, kde jednotlivé podsystémy jsou distribuované a k přenosu dat dochází na sdíleném médiu (síti) s přepojováním paketů, je nutně přítomno proměnné dopravní zpoždění (ang. time delay) a nedeterminismus způsobený chybami přenosu [1]. Tyto jevy mohou i v dobře navržených řídících systémech způsobit chybnost chování, která může vést k výpadku, poškození zařízení a újmě na zdraví přítomných osob. Z pohledu teorie řízení může dopravní zpoždění destabilizovat systém a proměnné dopravní zpoždění činí systém časově proměnný (ang. time-varying nebo time variant), čímž značně komplikuje návrh [2]. Pro návrháře je tedy zajímavé, ne-li nutné znát rozsah dopadu dopravního zpoždění a možné následky.

Tento projekt je postaven na experimentálním modelu Oddělení pokročilých algoritmů pro řízení a komunikace, vytvořeným v rámci projektu Slotcar platooning. Model je skupina autodráhových autíček, vybavených počítačovými deskami Raspberry Pi se systémem Linux a Java Runtime Environment. Řídící program, psaný v jazyce Java, umožňuje vozidlům po nastavení působit autonomně. Veškerá komunikace mezi autíčky probíhá po bezdrátové síti WiFi. Nastavení je možné skrz grafické rozhraní programu spuštěném na počítači, který je s auty ve společné síti (1). Vozidla mají senzory vzdálenosti vepředu i vzadu, regulovatelný motor s měřením rychlosti a STM procesor pro regulaci.

Cílem projektu je zajištění synchronizace času mezi všemi zařízeními, zavedení časových značek (ang. timestamps) do implementace řídícího programu a analýza sítě.



Obrázek 1. Experimentální model autonomních autíček

II. Synchronizace času

Mají-li jednotlivé uzly sítě vyměňovat informace o čase m.j. pro znalost zpoždění mezi vznikem a zpracováním dat, je nutné zajistit synchronizované hodiny všech uzlů před spuštěním následných operací. Čas hodin je možné popsat jako

$$t^S = t + \delta^S$$

$$\delta^S = \delta_p^S + \rho^S t$$

kde t je skutečný čas, t^S je čas hodin S, δ^S je posun hodin (ang. skew), δ_p^S je pevná složka posunu a ρ^S je rozptyl (ang. drift). Rozptyl se mezi hodinami liší podle nedokonalosti a teplotní závislosti hodinových krystalů. Pevná složka posunu je způsobená rozdílně nastaveným časem při spuštění, např. při ztrátě napájení a bez bateriově napájeného RTC obvodu dojde k resetu času na začátek epochy.

Synchronizaci je možné řešit hardwarově či softwarově, avšak hardwarové řešení je na modelu pohybujících se dráhových aut nepraktické. Softwarové algoritmy staví na opakované výměně času svých hodin a odhadu rozdílu posunů. Tyto výměny jsou také ovlivněny proměnným zpožděním. Algoritmy tedy musí být vůči zpoždění odolné. Odhad rozptylů je nad rámec projektu, avšak důsledek rozptylu je, že synchronizace času není jednorázová akce a po jisté době je potřeba ji opakovat. Ukazuje se, že zdroj hodin v Raspberry Pi má rozptyl téměř konstantní a za den vznikne chyba zhruba 4 sekund [3]. NTP je schopno tento rozptyl opravit. Režie (ang. overhead) spojená se synchronizací času za chodu není významná.

A. NTP

Network time protocol je protokol navržený k synchronizaci času po síti. Je to jeden z dnes nejdéle používaných internetových protokolů a jeho referenční implementace, s více než 360 000 řádky kódu, je schopna přesně udržovat správný čas i v případě výpadku synchronizačních serverů. Synchronizace probíhá mezi serverem a klientem. Klient má pevně nastavené servery, kterých se smí tázat, a vybírá si z nich ten nejvhodnější. Servery jsou klasifikovány topologickými vrstvami zvanými straty, server synchronizovaný vůči přesným externím hodinám je stratem 1, server synchronizovaný vůči serveru stratu 1 je stratum 2 atd.

Po internetu je NTP schopen synchronizovat čas s přesností desítek milisekund, po lokální síti s přesností desetin milisekund [4]. Toho využijeme v síti modelu dle obrázku (1) - vytvoříme NTP server v routeru, který slouží zároveň jako DHCP server a AP. Čas tohoto serveru může být synchronizován vůči jinému serveru mimo lokální síť, ale také nemusí. Pro autíčka to není důležité a promítne se pouze do záznamů (ang. log) určených pro lidské oko. V autíčku spustíme NTP klient a nastavíme jej na pevnou IP adresu routeru. Tím dosáhneme symetrické a spolehlivé cesty mezi synchronizovanými dvojicemi klient-server. Takto nastavená lokální NTP topologie vede k tomu, že

po připojení autíčka do sítě dojde k přijetí serveru a úspěšné synchronizaci během sekund s přesností pod $1\,\mathrm{ms}$ mezi členy sítě. V jiném případě, kdy lokální server neběží a auta se připojují k veřejným NTP serverům, byla změřena doba k synchronizaci i 10 minut. Přesností pod $1\,\mathrm{ms}$ myslíme rozdíl časů v síti.

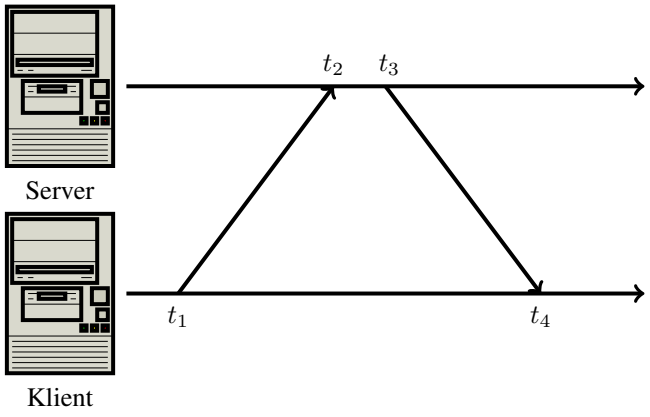
Referenční NTP je stále doporučená implementace. Do budoucna je však možné přejít na úspornější implementace SNTP, Chrony či budoucí náhradu referenční implementace, Ntimed.

B. Round-trip delay synchronizace hodin

V rámci projektu byla započata implementace základního algoritmu pro zjištění posunu hodin klienta, použitého v NTP implementaci [5], přímo do řídícího programu. Algoritmus začíná výměnnou zobrazenou v obrázku 2. Přeposílá se paket o velikosti čtyř časových značek, které jsou postupně do paketu vyplňovány. Z těchto časů poté lze vypočítat posun δ pomocí vzorce

$$\delta = \frac{t_2 - t_1 + t_3 - t_4}{2}$$

a korigovat jedny hodiny. Vypočtený posun je zatížen chybou a iterováním lze chybu snižovat pod jistou hranici. Algoritmus neřeší rozptyl a je tedy potřeba opakovat po uplynutí doby několika minut.



Obrázek 2. Měření obousměrného zpoždění

Oproti použití NTP je synchronizace zabudována do programu a jeho chod je tímto synchronizován ve smyslu závislosti procesu řízení na procesu synchronizace času. Pokročilost a kvalita NTP je však prokázána a pro model dostačující a výhody vlastní implementace neopodstatňují pokračování.

III. Časové značky

Znalost času vzniku síťového paketu umožňuje určit stáří obsažené informace při zpracování porovnáním s aktuálním časem hodin. Toto stáří je právě dopravní zpoždění od vzniku informace po její zpracování a lze jej využít při kalkulacích či rozhodování.

V programu je definována nejvyšší úroveň dat vložená do nižších vrstev paketu podle obrázku 3. Rámec linkové vrstvy má minimální velikost bez zavedení časových značek 81 bajtů. Toto číslo je součtem velikostí v tabulce I. Zavedením časové značky se zvětšila velikost o 4 bajty, tedy 6% zvětšení rámce datového přenosu. 4 bajty časové značky s přesností 1 milisekundy pojmou téměř 50 dní, což je mnohem déle než běžná doba běhu modelu.

Čas je strojově reprezentován jako doba od začátku epochy, která pro Javu je o půlnoci 1. 1. 1970 UTC. V implementaci použijeme jinou epochu, která bude jednotná pro všechna auta a pro kterou dojde k chybám spojených s přetečením číselné reprezentace času ve známých okamžicích. Takovou epochou může být předchozí půlnoc. Takto zavedená „plovoucí“ epocha je pro autíčka shodná po zajištění synchronizovaného času. K přetečení informace času v 4-bajtovém celém čísle nedojde kvůli obnovování epochy a jediný problém může vzniknout při přenosu paketu během této obnovy.

Tabulka I
Velikosti rámců OSI vrstev a vlastních vrstev bez vložených dat

Vrstva	Velikost (bajtů)
CarPacket (bez značky)	5
WIFIFrame	3
UDP [6]	8
IPV4 [7]	20
MAC [8]	46

Implementace časových značek v jazyce Java s přepočítáním na novou epochu je ve třídě slotcar.clock.CarClock. Využívá knihovny pro datum a čas z Java SE 8. Třída poskytuje statické metody, které jsou dále použity v programu. Rozšířena byla třída slotcar.network.CarPacket, která reprezentuje data z obrázku 3, o časovou značku. Časová značka je vytvořena při instancování paketu

```
CarPacket packet = new CarPacket(service.
    getServiceNum(), logAddr, this.myLogAddr,
            payload, CarClock.getTimestampMilli())
```

a přečtena je při události přijetí paketu

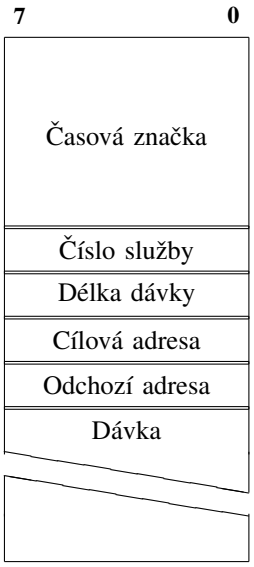
```
CarPacket packet = new CarPacket(e.getPacket());
LocalDate timestampDate = CarClock.
    getDateFromTimestampMilli(packet.milliSent);
Duration controlDelay = Duration.ofMillis(CarClock.
    getTimestampMilli()-packet.milliSent);
```

Tento controlDelay představuje zpoždění, tedy součet dob mezi vznikem zprávy, odesláním, přijetím a zpracováním. Zajisté by se dalo čas změřit přímo při odeslání a ihned při přijetí. Takový změřený časový úsek by reprezentoval čistěji zpoždění způsobené sítí a určily by se všechny mezičasy zmíněného sledu událostí. Jejich znalost však není zajímavá sama o sobě a vystačíme si s jejich součtem.

IV. Analýza

A. Měření zpoždění

V předchozí sekci bylo popsáno, jakým způsobem měříme dopravní zpoždění. Podíváme-li se na logaritmické histogramy



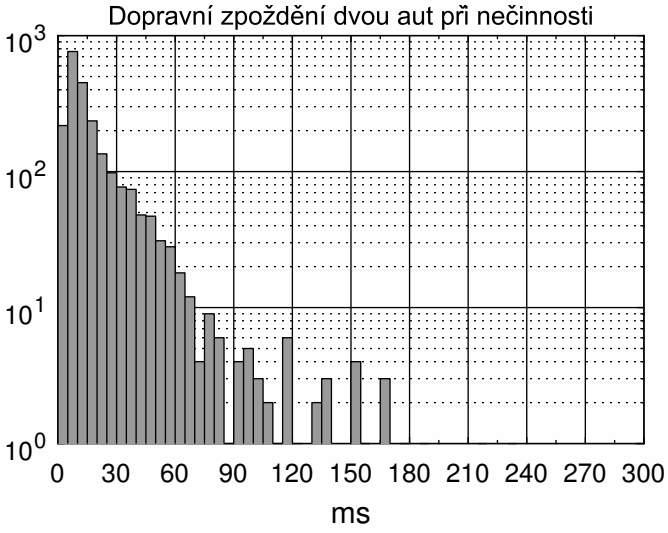
Obrázek 3. Struktura dat vkládaných do WiFi rámce

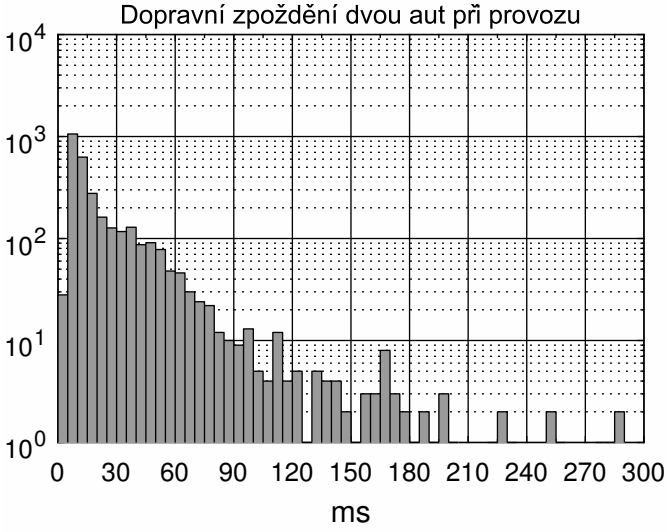
v obrázku 4, můžeme dojít k několika závěrům. Většina zpráv se ke zpracování dostane během 0 až 60 milisekund. Vyšší zpoždění však existují četně do 200 milisekund a existují i jedinci s delším zpožděním.

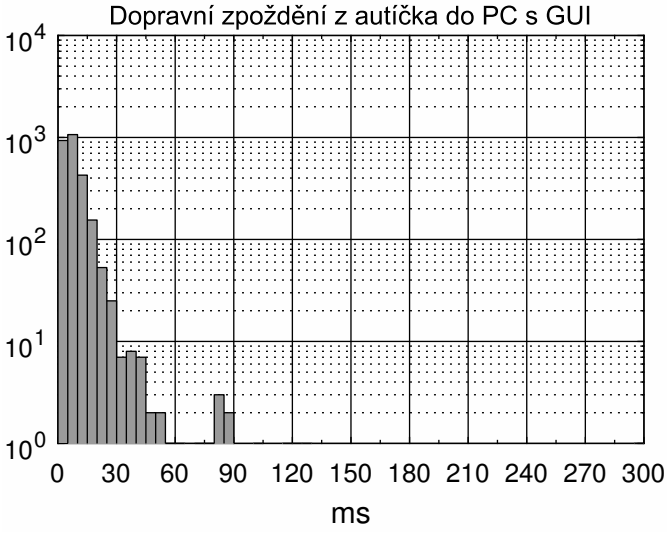
Důvody pro náhodnost dopravního zpoždění jsou [2]

- Čekání na vyprázdnění vlastní fronty zpráv
- Přeposílání zpráv v důsledku odhalení chyby přenosu
- náhodné čekání při detekci kolize (CSMA/CA)

Všechny tyto jevy jsou častější při vyšším zatížení a to můžeme pozorovat v prostředním histogramu. Zde je histogram posunut doprava a začínají se objevovat zpoždění výrazně vyšší. Dá se očekávat, že s dalším zatížením budou tyto změny výraznější. Posledním pozorováním je, že přenos z autíčka do počítače má celkově mnohem kratší zpoždění. To ukazuje na možný omezený výkon Raspberry Pi a jeho WiFi modulu.







Obrázek 4. Histogramy dopravního zpoždění s logaritmickou osou

REFERENCE

[1] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 2000. [Online]. Available: http://itec.hust.edu.cn/~liuwei/common-comnet/textbook/Comnet-A%20Top-Down%20Approach,%203rd%20Edition.pdf

[2] J. Nilsson, "Real-time control systems with delays," 1998.

[3] R. Bergsma. (2013) How accurately can the raspberry pi keep time? [Online]. Available: https://blog.remibergsma.com/2013/05/12/how-accurately-can-the-raspberry-pi-keep-time/

[4] D. Mills *et al.* (2012) Network time synchronization research project. [Online]. Available: https://www.eecis.udel.edu/~mills/ntp.html

[5] D. Mills, "Internet time synchronization: the network time protocol," University of Delaware, Tech. Rep., 1991. [Online]. Available: https://www.eecis.udel.edu/~mills/database/papers/trans.pdf

[6] J. Postel, "User datagram protocol," RFC Editor, Tech. Rep., 1980. [Online]. Available: https://www.rfc-editor.org/rfc/rfc768.txt

[7] ——, "Internet protocol," RFC Editor, Tech. Rep., 1981. [Online]. Available: http://www.rfc-editor.org/info/rfc791

[8] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," 2012.