Bachelor's thesis

EFFICIENCY AND UTILIZATION OF VECTOR PACKET PROCESSING IN HIGH-SPEED NETWORKS

Ondřej Slavík

Faculty of Information Technology
Department of Computer Systems
Supervisor: Ing. Jan Fesl, Ph.D.
April 13, 2025

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Efficiency and utilization of Vector Packet Processing in high-speed networks
Student:	Ondřej Slavík
Supervisor:	Ing. Jan Fesl, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Networks and Internet 2021
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2025/2026

Instructions

Vector Packet Processing (VPP) je moderní softwarový framework, který umožňuje zpracování paketů ve vysokorychlostních sítích na úrovni uživatelského prostoru operačního systému. Významnou výhodou využití VPP by mělo být výrazné zvýšení propustnosti a snížení latence v rámci vysokorychlostní sítě. Zmíněné výhody VPP jsou primárně teoretické a zatím nebyly experimentálně dostatečně prokázány.

V rámci tvorby bakalářské práce postupujte dle níže uvedených kroků:
1) Nastudujte a popište detailně všechny principy, které VPP používá, jak je implementováno a jak lze VPP efektivně využívat.
2) Vytvořte testovací scénáře, které umožní srovnat efektivitu a cenu využití VPP oproti běžnému způsobu zpracování paketů na úrovni jádra operačního systému.
3) Po poradě s vedoucím práce realizujte infrastrukturu vhodnou pro reálné otestování VPP.
4) Na základě bodu 2) proveďte dostatečný počet měření (minimálně stovky) a srovnejte možný dosažitelný průtok, latenci a spotřebu el. energie s využitím resp. bez využití VPP.
5) Proveďte důkladný rozbor a diskuzi výsledků z předchozího kroku a explicitně uveďte nevýhody využití VPP, pokud nějaké budou.

Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 28 November 2024 in Prague.

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Slavík Ondřej. *Efficiency and utilization of Vector Packet Processing in high-speed networks.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2025.

I would like to express my sincere gratitude to my thesis supervisor, Ing. Jan Fesl, Ph.D., for his guidance, support, and valuable insights throughout the entire process of writing this thesis.
My thanks also go to the Silicon Hill club of the CTU Student Union for providing an inspiring environment and the technical resources that supported my work.
Finally, I would like to thank my family for their unwavering support, encouragement, and never ending patience during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Section 2373(2) of Act No. 89/2012 Coll., the Civil Code, as amended, I hereby grant a non-exclusive authorization (licence) to utilize this thesis, including all computer programs that are part of it or attached to it and all documentation thereof (hereinafter collectively referred to as the "Work"), to any and all persons who wish to use the Work. Such persons are entitled to use the Work in any manner that does not diminish the value of the Work and for any purpose (including use for profit). This authorisation is unlimited in time, territory and quantity.

In Prague on April 13, 2025

Abstract

Fill in the abstract of this thesis in English. Lorem ipsum dolor sit amet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Keywords Vector Packet Processing, Network benchmark, Energy efficiency, Linux network stack, Data Plane Development Kit

Abstrakt

Fill in the abstract of this thesis in Czech. Lorem ipsum dolor sit amet. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Klíčová slova enter, comma, separated, list, of, keywords, in, CZECH

Contents

List of Figures

List of Tables

List of code listings

List of abbreviations

DFA	Deterministic Finite Automaton
FA	Finite Automaton
LPS	Labelled Prüfer Sequence
NFA	Nondeterministic Finite Automaton
NPS	Numbered Prüfer Sequence
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	eXtensible Stylesheet Language Transformations
W3C	World Wide Web Consortium

Introduction

Modern high-performance network devices are usually proprietary systems that combine custom hardware, specialized operating systems, and tightly coupled software. While these solutions offer high throughput and reliability, they are typically expensive, inflexible, and slower to evolve due to their closed design and development model. Vector Packet Processing (VPP) is a high-performance network stack that operates at layers 2 to 4 of the ISO/OSI model. It was originally developed by Cisco Systems, Inc. (which is a world leader in networking) and open-sourced in 2016 under the Fast Data Project (FD.io), that is part of the Linux Foundation. VPP brings the ability to perform efficient, high-speed packet processing on common off-the-shelf (COTS) hardware, across a wide range of platforms and operating systems. Its open and flexible architecture opens the door to a new class of network applications that can be deployed and scaled more easily than traditional hardware appliances. In this way, VPP could represent a shift in the traditionally conservative networking world, echoing the "Mainframe to PC" revolution, where general-purpose systems replaced proprietary platforms, enabling broader innovation and accessibility.

Since VPP was open-sourced only recently, it has not yet been widely adopted by the market, and there are only a limited number of academic studies on the subject. As a result, this area remains underexplored. This thesis aims to contribute to this field by evaluating VPP's[1] performance, with a particular focus on its electricity consumption. The findings could provide valuable insights for the industry and guide future research, especially in light of the increasing importance of energy efficiency, as highlighted in recent forecasts by ČEPS a.s. regarding the future of energy resources in the Czech Republic.

With the development of AI and the growing demand for high-resolution streaming services, it is highly likely that the demand for internet bandwidth

[1]The abbreviation VPP is also commonly used in academic literature to refer to a Virtual Power Plant.

will continue to rise. This will result in an increased need for network equipment capable of processing larger volumes of data more efficiently. Therefore, it is crucial to explore technologies like VPP that are capable to handle this growing demand and to explore their energy efficiency.

This thesis is divided into two parts: Theoretical and Practical. The Theoretical part presents the traditional approach to networking and packet processing, as well as an overview of how VPP is designed and the principles on which it operates. Additionally, it introduces the testing scenarios used. The Practical part describes the testing infrastructure, presents the results of various measurements, and provides an analysis of the findings.

Chapter 1

Theoretical part

1.1 "Vector Packet Processing (VPP) and Its Operating Principles

This section describes the fundamental principles behind the Vector Packet Processing (VPP) technology, which aims to enable efficient and high-performance network packet processing. VPP is built on modern programming and architectural principles that allow maximum utilization of contemporary hardware, particularly in parallel processing and memory access optimization.

The section begins with a brief description of traditional network traffic processing methods used by operating systems and their limitations in terms of performance and scalability. Following that, the architecture of VPP is explored in detail, explaining how packets are processed in vectors, the use of a node graph, and the various techniques that contribute to its high efficiency—such as I/O and compute batching, zero-copy methods, and lock-free multithreading. The purpose of this section is to provide a theoretical foundation for understanding how VPP operates and where it can be effectively used.

1.1.1 Introduction to traditional network traffic processing

A *network packet* is a basic unit of data transmitted over a network. It consists of a *header*, which includes control information such as source and destination IP addresses, and a *payload*, which carries the actual user data. Packets are routed independently through the network and reassembled at the destination. This structure allows efficient and reliable communication, even over complex or unreliable network paths.

Currently, packet processing works as follows: a packet arrives at the net-

work card, which then issues a system call (syscall) to the operating system for packet processing. The microprocessor must save the currently executing instruction, perform a context switch, locate the appropriate service routine in the interrupt vector table, and handle the packet processing. Once completed, it must restore the saved instruction, perform another context switch, and return to processing the interrupted program.

This system for operating peripherals was designed under the assumption that the peripherals would not request interrupts continuously, which is not the case with network devices that need to process large volumes of data split into small parts. This method requires the microprocessor to execute a significant number of instructions not directly related to packet processing. Chase et al. [1] discovered [1] that if MTU is 1500 bytes, then interrupt handling accounts for 20% - 25% of receiver packet-processing overhead. Another disadvantage of tradidtional packet processing is the inefficient handling of cache memory; the processing of the packets one by one in response to interrupts leads to frequent cache misses in both cache & inctruction caches.[2][2]

1.1.2 Vector Packet Processing: An Introduction

Vector Packet Processing (VPP) is a multi-platform network stack that operates at layers 2-4 of the ISO/OSI model and is developed by the FD.io project. It consists of a set of forwarding vertices arranged in an oriented graph and auxiliary software and provides out-of-the-box switch/router functionality. Unlike traditional network stacks, which run in the kernel, VPP operates in user space.

In a traditional approach, packets are processed one by one. In contrast, VPP reads the largest available number of packets called vector from the network interface card (NIC) and processes the entire vector through a VPP node-graph one node at a time. Each node in this graph handles a specific part of the packet processing. This approach reduces cache misses and spreads fixed overhead costs across multiple packets, lowering the average processing cost per packet. Additionally, it allows VPP to take advantage of multiple cores, enabling parallel processing, which significantly improves overall performance.

Vector Packet Processing (VPP) runs on common off-the-shelf hardware (COTS), ensuring its broad compatibility and flexibility for deployment. It supports various architectures such as x86, ARM, and Power, and can be deployed on both standard servers and embedded devices. The design of VPP is agnostic to hardware, kernel, and deployment platform, meaning it can operate across a wide range of systems, including bare metal servers, virtual machines (VMs), and containers. This approach allows VPP to be deployed on widely available infrastructure without the need for specialized hardware.[3]

[1] kap. 3.3 obr. 6
[2] kap. 4.2

1.1.3 Techniques used in VPP

According to Linguaglossa et al.[4] VPP uses theese kernel-bypass techniques:

- **Lock-Free Multi-Threading (LFMT)** is a programming technique that leverages modern multi-core CPUs to increase system performance. In network applications, parallelism is achieved by running multiple threads in the same time. Ideally, the more threads are used, the better the system performance but only up to a saturation point beyond which additional threads bring no gainns. However, to reach this ideal performance, traditional synchronization mechanisms such as mutexes and semaphores must be avoided, as they introduce delays due to thread contention. Instead, lock-free architectures have to be used, allowing threads to operate independently without blocking each other. In the context of VPP this approach is enabled by hardware features like multi-queue NICs, which allow each thread to handle a distinct subset of traffic, ensuring efficient and parallel processing.
- **I/O batching (IOB)** is a key technique used in VPP. Instead of raising an interrupt for every incoming packet, the network interface card (NIC) collects multiple packets into a buffer and triggers an interrupt only when the buffer is full. This reduces the overhead caused by frequent context switching and interrupt handling. VPP typically uses poll-mode drivers, which collect packets in batches without relying on interrupts. Moreover, the batching technique is applied system-wide in VPP. This approach maximizes CPU efficiency, improves cache usage, and delivers stable, high-throughput performance even under heavy load.
- **Compute batching (CB)** is a technique that extends I/O batching to the processing phase itself. Instead of processing one packet at a time, network functions are designed to operate on entire batches of packets. This approach minimizes overhead from function calls (such as context switches and stack setup) and improves instruction cache efficiency. When a batch of packets enters a processing function, only the first packet might cause an instruction cache miss, while the rest benefit from already-warmed cache. Additionally it is possible to take advatage of instruction-level parallelism.
- **Receive-Side Scaling (RSS)** is a hardware-based technique used by modern NICs to distribute incoming packets across multiple RX queues. This enables parallel packet processing by allowing each queue to be handled by a separate thread, improving scalability and throughput. Packet assignment is typically done using a hash function over packet header fields (e.g., the 5-tuple).
- **Zero-Copy (Z-C)** is a technique used to eliminate unnecessary memory copying during packet processing. Instead of copying incoming packets

from the network interface card (NIC) to a separate buffer via system calls, the NIC writes packets directly into a pre-allocated memory region that is shared with the user-space application via Direct Memory Access (DMA). This allows the application to access packet data without invoking system calls or duplicating memory, significantly reducing CPU overhead.

- **Cache Coherence and Locality (CC&L)** are critical factors in the performance of modern software-based packet processing systems. In current COTS architectures, memory access has become a major bottleneck, which is mitigated by a multi-level cache hierarchy. Minimizing cache misses and maintaining data locality during packet processing is essential for achieving high performance and low latency.

- **Low-Level Parallelism (LLP)** refers to the ability to exploit the internal micro-architecture of modern CPUs, including multi-stage pipelines, arithmetic-logical units (ALUs), and branch predictors that help maintain pipeline efficiency. Well-optimized code can keep these pipelines full and execute multiple instructions per clock cycle, increasing overall throughput. Performance can be further improved by giving hints to the compiler – such as indicating the likely outcome of conditional branches – to reduce pipeline stalls. Vectorized packet processing and specific coding practices can take full advantage of these hardware features and VPP was specifically designed to taky advantage of LLP.

1.1.4 VPP Processing Graph and Graph nodes

At the core of VPP (Vector Packet Processing) lies the Packet Processing Graph, a directed graph composed of relatively small, modular & loosely coupled nodes. Each node is designed to perform a specific task and there are 3 types of them: *process*, *input* & *internal*. Process nodes do not participate in the packet forwarding graph; instead, they handle timers, events, and other background tasks within the VPP runtime. Input nodes are used for input of data and internal nodes are used for vector processing. Internal nodes also serves as output nodes. When a vector of packets is prepared by input node, it is then pushed through the internal nodes. During processing, the vector may be split if the batch contains packets of different protocols or types, as they may need to follow different paths through the graph When the original vector is completely processed, the process repeats. Illustration of this Processing Graph is shown in fig. 1.1.

Thanks to VPP's modular design, the processing graph is highly customizable and extensible. New nodes – referred to as plugins – can be easily added to implement specific functionality or repleace existing ones. Plugins are shared libraries that are loaded during startup of VPP, and they are not dependent on the VPP source code, allowing them to be developed independently. More-

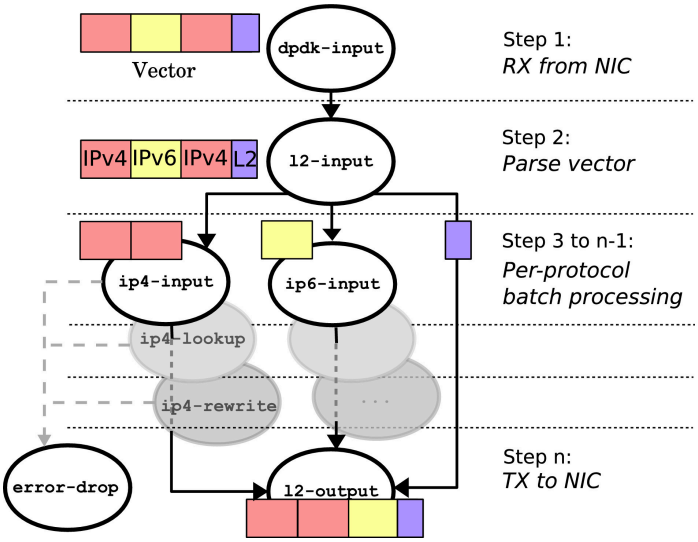

Figure 1.1 Picture showing the VPP Processing Graph [4]

over, existing nodes can be rewired to modify the packet processing logic when necessary.[4, 5, 6]

1.2 Implementation of Vector Packet Processing

1.2.1 DPDK and Its Role in VPP

The Data Plane Development Kit (DPDK) is an open-source collection of libraries and drivers designed to support high-speed packet processing in user space. It was initially developed by Intel in 2010 and is now maintained as a Linux Foundation project. DPDK provides a set of APIs and components that allow applications to bypass the kernel network stack and directly access network interface cards (NICs) through poll-mode drivers, significantly reducing the overhead associated with traditional packet handling mechanisms.[7]

The DPDK completely bypasses the kernel, communicating directly with the NIC. DPDK avoids the use of the kernel's system calls, instead handling its own I/O synchronization and memory management. DPDK employs a Poll Mode Driver (PMD) that uses busy-polling to retrieve, process, and deliver network packets to user-space applications without relying on interrupts. While this approach enhances performance by reducing latency, it also results in high CPU utilization, with the CPU usage on each core remaining close to 100% regardless of the network load.[8]

DPDK is used in VPP for interfacing with hardware and handling Layer 1 (L1) functionalities. It is implemented as a plugin called *dpdk-plugin*.[4, 5]

1.2.2 Architecture of VPP Modules

1.2.3 Configuration and Startup

1.2.4 Plugins and Extensibility

Chapter 2
Pratical part

2.1 Building Infrastructure for Measurement

2.2 Test Scenarios & Results

2.2.1 Scenario #1a

2.3 Presentation and Analysis of Results

2.4 Ut enim ad minim veniam

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed,

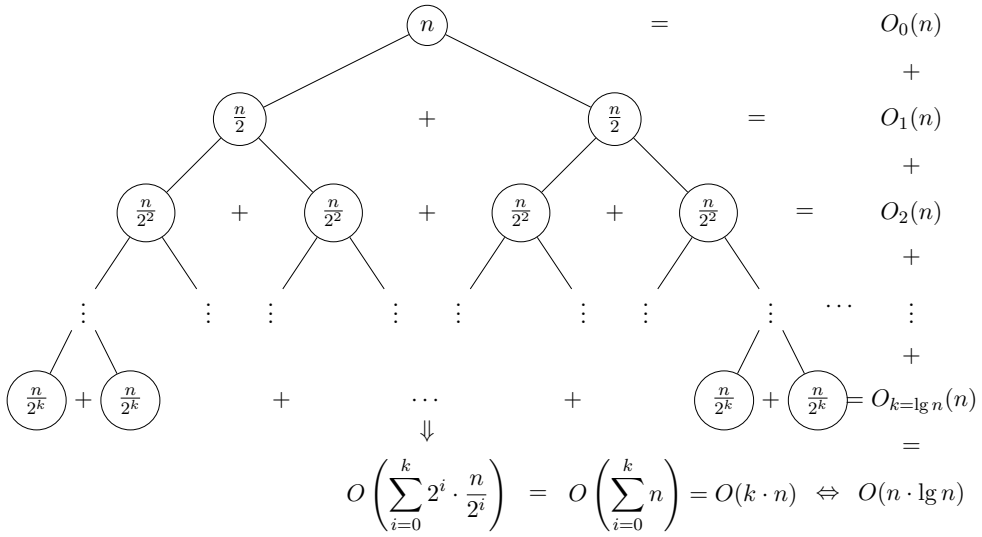

Figure 2.1 Lorem ipsum dolor sit amet

leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

2.5 Ut enim ad minim veniam

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

2.5.1 Ut enim ad minim veniam

Curabitur ligula sapien, pulvinar a vestibulum quis, facilisis vel sapien. Duis condimentum augue id magna semper rutrum. Aliquam ornare wisi eu metus. Fusce aliquam vestibulum ipsum. Vivamus ac leo pretium faucibus2.1.

- Ut enim ad minim veniam, quis nostrud
- Ut enim ad minim
- Ut enim ad minim veniam, quis
  - Ut enim ad
  - Ut enim ad
    - Ut enim
    - Ut enim
      - Ut enim
      - Ut enim

2.6 Class aptent taciti

Even though dark mode can be very nice for websites and apps, it does not look good when printed. Consider using white mode when print-screening dark-moded content. The contrast with the white page is too big, as seen in Figure 2.2.

2.6.1 Class aptent taciti

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies

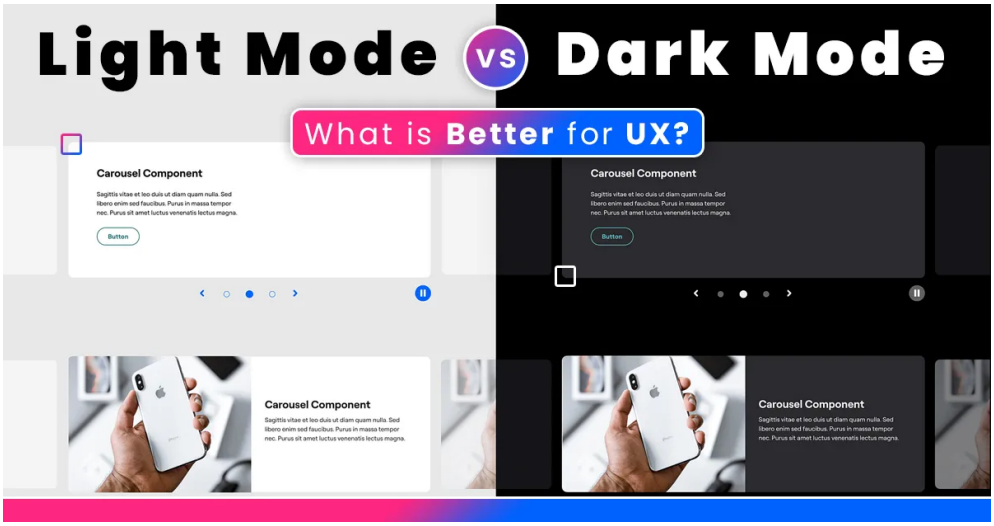

Figure 2.2 White and dark mode comparison. [9]

non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

1. Ut enim ad minim veniam, quis nostrud
2. Ut enim ad minim
3. Ut enim ad minim veniam, quis
   a. Ut enim ad
   b. Ut enim ad
      i. Ut enim
      ii. Ut enim
         A. Ut enim
         B. Ut enim

```
#include<stdio.h>
#include<iostream>
// A comment
int main(void)
{
    printf("Hello␣World\n");
    return 0;
}
```

Code listing 2.1 Zbytečný kód

2.7 Ut enim ad minim veniam, quis nostrud

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Nulla non arcu lacinia neque faucibus fringilla. Vestibulum erat nulla, ullamcorper nec, rutrum non, nonummy ac, erat. Aliquam erat volutpat. Proin pede metus, vulputate nec, fermentum fringilla, vehicula vitae, justo.[1] Etiam dictum tincidunt diam. In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero. Nulla est. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Aenean id metus id velit ullamcorper pulvinar. Fusce wisi. Integer lacinia. Aliquam id dolor. Pellentesque pretium lectus id turpis. Suspendisse sagittis ultrices augue. In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero. Sed ac dolor sit amet purus malesuada congue. [10]

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci. Duis viverra diam non justo. Curabitur ligula sapien, pulvinar a vestibulum quis, facilisis vel sapien. Duis condimentum augue id magna semper rutrum. Aliquam ornare wisi eu metus. Fusce aliquam vestibulum ipsum. Vivamus ac leo pretium faucibus. [11]

2.7.1 Ut enim ad minim veniam, quis nostrud

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Nulla non arcu lacinia neque faucibus fringilla. Vestibulum erat nulla, ullamcorper nec, rutrum non, nonummy ac, erat. Aliquam erat volutpat. Proin pede metus, vulputate nec, fermentum

[1] Ut enim ad minim veniam, quis nostrud exercitation.

Typ	Prostředí	LaTeXovská zkratka	TeXovská zkratka
Text	`math`	`\(...\)`	`$...$`
Displayed	`displaymath`	`\[...\]`	`$$...$$`

Table 2.1 Zadávání matematiky

fringilla, vehicula vitae, justo. Etiam dictum tincidunt diam. In laoreet, magna id viverra tincidunt, sem odio bibendum justo. [12]

Nulla est. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Nullam feugiat, turpis at pulvinar vulputate, erat libero tristique tellus, nec bibendum odio risus sit amet ante. Aenean id metus id velit ullamcorper pulvinar.

2.7.1.1 Class aptent taciti

▶ **Definiton 2.1** (Optional label)**.** *Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.*

▶ **Example 2.2.** Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.

Nadpis 5. úrovně

▶ **Theorem 2.3.** *Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.*

Proof. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci. □

Level 5 heading

▶ **Corollary 2.4.** *Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.*

▶ **Proposition 2.5.** *Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.*

▶ Note 2.6. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.

▶ Remark 2.7. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.

▶ **Lemma 2.8.** *Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Fusce suscipit libero eget elit. Etiam dui sem, fermentum vitae, sagittis id, malesuada in, quam. Aliquam id dolor. Curabitur bibendum justo non orci.*

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Chapter 3

Lorem ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. In sem justo, commodo ut, suscipit at, pharetra vitae, orci. Pellentesque pretium lectus id turpis. [13]

3.1 Donec odio tempus molestie

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. [14, 15]

3.1.1 Class aptent taciti

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies

et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Kapitola 1 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis.

Kapitola 2 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis.

Kapitola 3 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis.

Kapitola 4 Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. [16]

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3.2 Lorem ipsum dolor sit amet

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Appendix A
Nějaká příloha

Sem přijde to, co nepatří do hlavní části.

Bibliography

1. GALLATIN, Andrew J.; CHASE, Jeffrey S.; YOCUM, Kenneth G. Trapeze/IP: TCP/IP at Near-Gigabit Speeds. In: *Proceedings of the USENIX Annual Technical Conference*. 1999, pp. 109–120. Available also from: https://www.usenix.org/event/usenix99/full_papers/gallatin/gallatin.pdf.
2. COX, Alan L.; SCHAELICKE, Lambert; DAVIS, Al; MCKEE, Sally A. Profiling I/O Interrupts in Modern Architectures. *Proceedings of the Workshop on Performance Analysis and Its Impact on Design*. 2000. Available also from: https://users.cs.utah.edu/~ald/pubs/interrupts.pdf.
3. FD.IO. *What is VPP?* [https://wiki.fd.io/view/VPP/What_is_VPP%3F]. 2025. Accessed: 2025-04-07.
4. LINGUAGLOSSA, Leonardo; ROSSI, Dario; PONTARELLI, Salvatore; BARACH, Dave; MARJON, Damjan; PFISTER, Pierre. High-speed data plane and network functions virtualization by vectorizing packet processing. *Computer Networks*. 2019, vol. 149, pp. 187–199. ISSN 1389-1286. Available from DOI: https://doi.org/10.1016/j.comnet.2018.11.033.
5. BARACH, David; LINGUAGLOSSA, Leonardo; MARION, Damjan; PFISTER, Pierre; PONTARELLI, Salvatore; ROSSI, Dario. High-speed Software Data Plane via Vectorized Packet Processing. *IEEE Communication Magazine* [https://perso.telecom-paristech.fr/drossi/paper/rossi18commag.pdf]. 2018, vol. 56, no. 12, pp. 97–103. ISSN 0163-6804. Available from DOI: 10.1109/MCOM.2018.1800069.
6. FD.IO. *Extensible: VPP and its plugin architecture* [https://fd.io/docs/vpp/v2101/whatisvpp/extensible]. 2021. Accessed: 2025-04-10.
7. DPDK PROJECT. *About DPDK* [https://www.dpdk.org/about/]. 2025. Accessed: 2025-04-13.

8. FREITAS, Eduardo; DE OLIVEIRA FILHO, Assis T.; DO CARMO, Pedro R.X.; SADOK, Djamel; KELNER, Judith. A survey on accelerating technologies for fast network packet processing in Linux environments. *Computer Communications*. 2022, vol. 196, pp. 148–166. ISSN 0140-3664. Available from DOI: https://doi.org/10.1016/j.comcom.2022.10.003.
9. TOPDEVELOPERS.CO. *Light Mode Vs. Dark Mode – Whatś Your Choice For UX? | by TopDevelopers.co | Medium*. 2024. Available also from: https://medium.com/@topdevelopers-co/light-mode-vs-dark-mode-whats-your-choice-for-ux-5bce3ad4a368. [Citováno 27. 2. 2025].
10. CROCHEMORE, Maxime; RYTTER, Wojciech. *Jewels of stringology*. River Edge, NJ: World Scientific, 2002. ISBN 978-9810247829.
11. MOTWANI, Rajeev; ULLMAN, Jeffrey D.; HOPCROFT, John E. *Introduction to automata theory, languages, and computation*. Third. Harlow: Pearson, 2014. ISBN 9781292039053.
12. ŠESTÁKOVÁ, Eliška; JANOUŠEK, Jan. Automata Approach to XML Data Indexing. *Information*. 2018, vol. 9, no. 1. ISSN 2078-2489. Available from DOI: 10.3390/info9010012.
13. KOPKA, Helmut; DALY, Patrick W. *LATEX: podrobný průvodce*. Brno: Computer Press, 2004. ISBN 80-7226-973-9.
14. NEVEN, Frank. Automata, Logic, and XML. In: BRADFIELD, Julian (ed.). *Computer Science Logic*. Springer Berlin Heidelberg, 2002, vol. 2471, pp. 2–26. Lecture Notes in Computer Science. ISBN 978-3-540-44240-0.
15. LIBKIN, Leonid. Logics for Unranked Trees: An Overview. In: CAIRES, Luís; ITALIANO, Giuseppe; MONTEIRO, Luís; PALAMIDESSI, Catuscia; YUNG, Moti (eds.). *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2005, vol. 3580, pp. 35–50. Lecture Notes in Computer Science. ISBN 978-3-540-27580-0.
16. ASTLEY, Rick. *Never Gonna Give You Up (Official Music Video) – YouTube* [online]. 2009. Available also from: https://www.youtube.com/watch?v=dQw4w9WgXcQ&ab_channel=RickAstley. [Citováno 7. 1. 2025].

Obsah příloh