

Ben-Gurion University of the Negev

Faculty of Engineering Science

School of Electrical and Computer Engineering

Dept. of Electrical and Computer Engineering

Fourth Year Engineering Project

Preliminary design

P4 based Caching-based Acceleration Mechanisms in Datacenter Networks

|  |  |  |  |
| --- | --- | --- | --- |
|  | **p-2022-035** | | **Project number:** |
| **Nisani Amit**  **205664345** | | **Kedik Michael**  **316645415** | **Students**  **(name & ID):** |
|  | **Prof. Avin Chen, Dr. Scalosub Gabriel** | | **Supervisors:** |
|  | **17/01/23** | | **Submitting date:** |

Contents

[1. Abstract I](#_Toc124881715)

[1.1 English abstract I](#_Toc124881717)

[1.2 תקציר II](#_Toc124881718)

[2. Project Goals 1](#_Toc124881719)

[3. Research Proposal 2](#_Toc124881720)

[4. Literature Review 4](#_Toc124881721)

[4.1 Software-defined Network (SDN) 4](#_Toc124881722)

[4.2 Mininet Emulation Environment 5](#_Toc124881723)

[4.3 P4 Programming Language 5](#_Toc124881724)

[4.4 P4Runtime API 5](#_Toc124881725)

[4.5 BMv2 Programmable Switch 6](#_Toc124881726)

[4.6 SDN controller 6](#_Toc124881727)

[4.7 ONOS SDN Controller 7](#_Toc124881728)

[5. Proposed method 8](#_Toc124881729)

[5.1 Proposed method 8](#_Toc124881730)

[5.2 Constraints 9](#_Toc124881731)

[5.2.1 Time limitation 9](#_Toc124881732)

[5.2.2 Scalability 9](#_Toc124881733)

[5.3 Assumption 9](#_Toc124881734)

[5.4 Initial Risks 9](#_Toc124881735)

[6. Project's Content 10](#_Toc124881736)

[7. Final Testing Proposal 11](#_Toc124881737)

[8. Budget Estimation 12](#_Toc124881738)

[8.1 Equipment: 12](#_Toc124881739)

[8.2 Software and license 12](#_Toc124881740)

[8.3 Manpower 12](#_Toc124881741)

[8.4 Overhead: 25% 12](#_Toc124881742)

[8.5 Unexpected expenses: 10% 12](#_Toc124881743)

[9. Schedule 13](#_Toc124881744)

[10. References 14](#_Toc124881745)

[11. Appendix 15](#_Toc124881746)

[11.1 טופס כיבוד זכויות יוצרים 15](#_Toc124881747)

[11.2 המלצת ציון לדו"ח מכין 16](#_Toc124881748)

# Abstract

**P4 based Caching-based Acceleration Mechanisms in Datacenter Networks**

Students Names: Kedik Michael, Nisani Amit

*kedikm@post.bgu.ac.il*

Advisers’ Names: Prof. Avin Chen, Dr. Scalosub Gabriel



## English abstract

Switches in data center networks are required to store an enormous amount of traffic rules. Usually, those rules are stored in an external device, called a controller, to which the access slows the network’s performance.

The purpose of the project is to find a solution that decreases multiple access to this external device by using Caching-based Acceleration Mechanisms, to speed up the routing process.

Our objective is to minimize the number of accesses to the controller, by using caching technologies and re-placing forwarding rules in distinct switches, which will speed up the overall throughput in the network.

The innovation is using the Cache Mechanisms and the connectivity between switches from different layers in the topology to get the necessary forwarding rule.

The proposed method is to create a basic network, based on a data center topology, with several switches that have cache storage for the most used forwarding rules. In case of missing information, a switch sends a request for forwarding rule to a higher-ranked switch. The highest-rank switch is the controller, which holds the entire forwarding policy.

We expect to get a significant speed-up of our overall throughput in the network.

**Keywords:** Switch, Cache, Routing, Controller, Mininet, P4, Data center, Software-Defined Networking.

**מנגנוני האצה מבוססי מטמון ברשתות מרכזי נתונים מבוססי P4**

שמות הסטודנטים: קדיק מיכאל, ניסני עמית

*kedikm@post.bgu.ac.il*

שם המנחה: פרופ' אבין חן, דר' סקלוסוב גבריאל

## תקציר

מתגים ברשתות מרכזי נתונים נדרשים לאחסן כמות גדולה של חוקי תעבורה. בדרך כלל, חוקים אלה מאוחסנים במכשיר חיצוני, הנקרא בקר, שהגישה אליו מאטה את ביצועי הרשת.

מטרת הפרויקט היא למצוא פתרון שיגרום להפחתת כמות הגישות למכשיר החיצוני, על ידי שימוש במנגנוני האצה מבוססי מטמון כדי לייעל את תהליך הניתוב.

המטרה שלנו היא למזער את מספר הגישות לבקר, על ידי שימוש בטכנולוגיות מבוססות זיכרון מטמון ועדכון של חוקי תעבורה במתגים, ובכך לשפר את התפוקה הכוללת ברשת.

החדשנות היא שימוש במנגנוני המטמון ובקישוריות בין מתגים משכבות שונות בטופולוגיה, כדי להפחית את התעבורה העודפת ברשת.

השיטה המוצעת היא יצירת רשת בסיסית, המבוססת על טופולוגיה של מרכז נתונים, עם מספר מתגים בעלי זיכרון מטמון עבור חוקי התעבורה הרלוונטיים. במקרה שלמתג אין את חוק התעבורה הרלוונטי, כאשר מגיעה אליו חבילת מידע, הוא שולח בקשת ניתוב למתג בעל דירוג גבוה יותר. המתג בעל הדרגה הגבוהה ביותר הוא הבקר, המחזיק בכל מדיניות ההעברה.

אנו מצפים לקבל שיפור משמעותי של התפוקה הכוללת ברשת.

(שפת תכנות לרשתות תקשורת), מרכז מידע, P4 (אמולציית רשת), Mininet**מילות מפתח :** מתג, מטמון, ניתוב, בקר, רשת מוגדרת תוכנה.

# **Project Goals**

The goal of the project is to improve the management of traffic in data centers by utilizing caching techniques to speed up routing. The efficiency will be measured by the reduction in the number of requests to the controller, with a target of 4% decrease. This is necessary, as internet accessibility has increased in the last decade, leading to a rise in overall traffic and the need for data centers to handle more traffic each year.

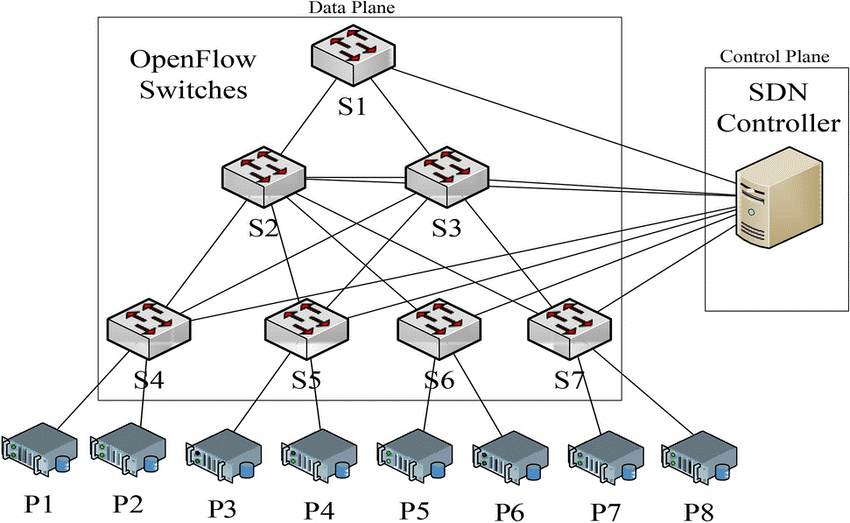
# **Research Proposal**

**Project’s name:** P4 based Caching-based Acceleration Mechanisms in Datacenter Networks.

**Project application:** increasing the throughput of a data center will reduce unnecessary traffic in the network. It will allow services that are more efficient by using the same resources.

**Technology:**

* **Mininet** – Mininet is a network emulator, which creates a network of virtual hosts, switches, controllers, and links. Its support flexible custom routing and Software-Defined Networking.[[1]](#M1)
* **P4 -** Programming Protocol-independent Packet Processors (P4) is a domain-specific language for network devices, specifying how data plane devices (switches, NICs, routers, filters, etc.) process packets.[[2]](#M2)
* **P4runtime -** The P4Runtime API is a control plane specification for controlling the data plane elements of a device defined or described by a P4 program.[[3]](#M3)
* **SDN -** Software Defined Networking (SDN) advocates separation of control and data plane. This paradigm shift in networking architecture.[[4]](#M4)
* **SDN Controller -** An SDN controller has complete view of the network. The controller also has the ability to change the network structure and services at run time.[[4]](#M4)



*Figure 1: block diagram of SDN network using SDN controller*

We will adjust routing table and allocate shared-cache of the switches through the SDN controller, to decrease the number of forwarding requests arriving to the SDN controller.

By using Mininet's emulator we can mimic the behavior and connectivity of a DC network and thus to get more realistic results, as seen in figure 1.

**Work plan:**

* Use Mininet environment that will consist of a small datacenter network that includes switches, hosts, an SDN controller and a traffic generator in multiple topologies.
* The project will use P4 programming language to design the packet pipeline within the switches, and include a cache mechanism to store forwarding rules within the switches themselves.
* Use a given caching algorithm and deploy in into the switches.
* Configure the SDN controller to manage the datacenter network and insert forwarding rules.
* Design a traffic generator.
* Evaluate the performance.

**Project's objectives:** is to enhance the speed of traffic forwarding in large network, thus allowing data centers to improve performance, decrease costs by reducing the need for costly hardware, and easily expand their infrastructure with minimal changes to hardware.

**System Performance Spec:** The caching mechanism should be deployable in all current SDN. By a given SDN, the algorithm will form a methodology, which will improve the DC performance. The equipment needed for this research is a PC equipped with emulation software, and enough memory to emulate a significant number of switches and hosts.

# **Literature Review**

In this section, we will present a review of the relevant literature for understanding the problem and finding ways to resolve it, until the proposed method is achieved.

In the literature review, we will talk about the Software-defined Network approach and its benefits. In addition, we will expand on Mininet, P4 programming language, and P4runtime API, which are tools that we will use in our project.

Moreover, we will elaborate on BMv2, which is a programmable switch that we will use to emulate a hardware switch.

Finally, as part of the proposed solution that we will present below, we will explain the importance of an SDN controller, and we will elaborate on ONOS, the controller that we will use.

## Software-defined Network (SDN)

Software-Defined Networking (SDN) is a new networking technology, which allows centralized, programmable control planes and data plan abstraction, where control and data plane are separated, so that network operators and service providers can manage directly their own virtualized resources and networks without recognizing detailed hardware technologies.[[5]](#M5) This approach differs from the traditional method, which uses dedicated hardware devices to control network traffic. Because the control plane is software-based, SDN is much more flexible than traditional networking.

SDN allows administrators to control the network, change configuration settings, provision resources, and increase network capacity — all from a centralized user interface, without the need for more hardware.

There are three parts to a typical SDN architecture: Applications that are used to communicate resource requests or information about the network, Controllers that use the information from applications to decide how to route a data packet, and Networking devices, which receive information from the controller about where to move the data.

There are many benefits to using the SDN approach. SDN allows data to move easily between distributed locations, which is critical for many of today’s services and applications, especially cloud applications. In addition, because of the speed and flexibility offered by SDN, it can support emerging trends and technologies, which require transferring data quickly and easily between remote sites.

## Mininet Emulation Environment

Mininet is a network emulator, which creates a network of virtual hosts, switches, controllers, and links. Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC. Mininet provides an easy way to get correct system behavior (and, to the extent supported by your hardware, performance) and to experiment with topologies.[[1]](#M1)

Mininet networks run real code that can move to a real system with minimal changes, for real-world testing, performance evaluation, and deployment. Importantly this means that a design that works in Mininet can usually move directly to hardware switches for line-rate packet forwarding.

## P4 Programming Language

Programming Protocol-independent Packet Processors (P4) is a domain-specific language for network devices, specifying how data plane devices such as switches and routers process packets.

Application developers and network engineers can now use P4 to implement specific behavior in the network.

P4 can be compiled in a Mininet emulation environment and program the network control plain behavior pre-run.[[2]](#M2)

## P4Runtime API

P4Runtime API is a control plane specification for controlling the data plane elements of a device defined or described by a P4 program. Through it, a user can redefine switches configured in P4. The protocol is still under development, but many specification sheets have already been published, strictly defining the API between data and control plane devices.[[3]](#M3)

## BMv2 Programmable Switch

BMv2 is a version of a software switch called Behavioral Model version 2. This software is written in C++11 and it is widely used in the academy as well as the industry for research and development purposes, regarding P4 and SDN-related scenarios. It is important to note that the performance of BMv2 - in terms of throughput and latency - is significantly less than that of a production-grade software switch.

It is possible to use BMv2 on top of the Mininet Switches and by doing so; we can manage the control plane through the data plain by using p4runtime.[[6]](#M6)

## SDN controller

An SDN controller is an application in a software-defined networking architecture that manages flow control for improved network management and application performance. The SDN controller platform typically runs on a server and uses protocols to tell [switches](https://www.techtarget.com/searchnetworking/definition/switch) where to send [packets](https://www.techtarget.com/searchnetworking/definition/packet).

The controller provides a high abstraction level of the forwarding elements management which is absent in today’s networks. Therefore, the controller is a fundamental component of the SDN architecture that will contribute to the success or failure of SDN.[[7]](#M7)

One major benefit of SDN controllers is that the centralized controller is aware of all the available network paths and can direct packets based on traffic requirements. Because of the controller's visibility into the network, it can automatically modify traffic flows and notify network operators about congested links.

## ONOS SDN Controller

ONOS (Open Network Operating System) is the leading open source SDN controller for building next-generation SDN solutions. It is an experimental distributed SDN control platform motivated by the performance, scalability, and availability requirements of large operator networks.[[8]](#M8)

ONOS supports both configuration and real-time control of the network, eliminating the need to run routing and switching control protocols inside the network fabric. By moving intelligence into the ONOS cloud controller, end-users can easily create new network applications without the need to alter the data plane systems.

The ONOS platform includes a set of applications that act as an extensible, modular, distributed SDN controller. It allows Simplified management, configuration and deployment of new software, hardware and a scale-out architecture to provide the scalability required to meet the rigors of production carrier environments.

# **Proposed method**

## Proposed method

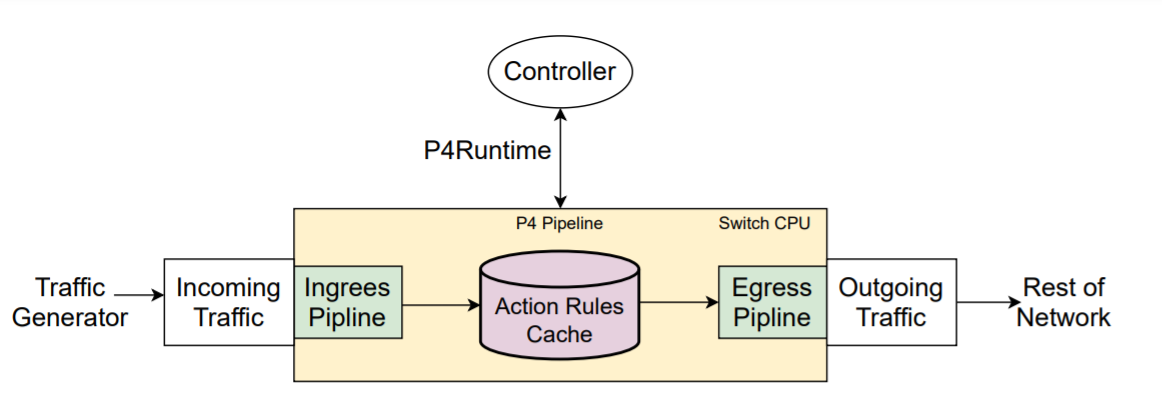
In this project, we will create a Data-Center network layout as seen in figure 1. We will do so by using Mininet.

We will implement a cache mechanism on every switch in the network that will contain forwarding rules needed by the switch. If the switch knows where to forward a certain packet, it will be considered a cache hit; otherwise, it will be considered a cache miss.

The goal of this proposed method is that the cache will hold the rules needed to handle as many packets as possible to maximize the cache-hit rate and reduce the number of hops to the higher layers in the network, thus reducing the overall unnecessary traffic in the network.

As mentioned above, a generated traffic is reaching the switch, as seen in figure 2 at the first block.

When a cache miss happens, the unknown destination packet will be sent to a higher-layer switch, hopefully, this switch will contain the destination in its cache table. This action will continue until a cache hit will occur or the packet will reach the final layer, the controller, which holds the information about all the destinations in the network. As seen in the second block, the ONOS controller that manage distributed algorithm, inserting rules to caches in switches belonging to different layers in the topology.

****

*Figure 2: general flow of the system.*

## Constraints

### 5.2.1 Time limitation

Our project’s duration is limited due to the academic year. The project’s due date is 25.6.23.

### 5.2.2 Scalability

As we mentioned, we will use Mininet and BMv2 switches, by doing so we will be able to conduct experiments and tests according to the limitations of the computer resources on which we will run the emulation. Therefore, although we estimate that the results of the project will indeed be compatible with the results in a real-life DC network, we will not be able to verify it.

## Assumption

We will assume that the test's results are compatible with a large-scale datacenter.

## Initial Risks

The main risk in our project is the assumption that the results are compatible with a real-life datacenter.

# Project's Content

The primary objective of the first part of the project is to establish an environment that integrates the use of an SDN controller (ONOS), and bmv2 switches that can be configured using the P4 programming language. To achieve this, we will research and identify the compatible versions of each component that can work together seamlessly. By doing so we will be able to create a topology that resembles data center in which we can conduct our experiments in the second part of the project.

In the second phase of the project, we will construct a traffic generator to simulate network traffic. Then, we will incorporate our caching strategies and run experiments with various workloads to evaluate the network's performance with those mechanisms. We will implement both push and pull caching algorithms. The pull algorithm allows each switch to identify frequently used rules and request them from the controller, which is then inserted into its caching table. The push algorithm allows the controller to identify frequently used rules and insert them directly into the relevant switch's caching table. Our aim is to merge these two methods and evaluate the results.

# Final Testing Proposal

Once we have set up our emulation environment and connected the necessary components, we plan to begin testing our system. These tests will determine whether it is more effective to use the distributed cache mechanism as previously demonstrated or to bypass it and go directly to the controller. It is important to note that the switches have limited memory cache, so the order of actions will vary depending on the traffic generator.

The goal is to optimize the usage of the cache in the switches. Each switch will check if the package's transfer rule is already in its cache and if so, it will be transferred to its destination, otherwise, it will be sent to the next switch of higher hierarchy to check its cache. If the rule is not found in the second switch, it will be sent to the controller for handling. In some cases, the controller may also update the cache in the switches as necessary.

To assess the effectiveness of our proposed method, we will evaluate the network's performance in terms of throughput and latency on a data center topology. To measure throughput and latency, we will use measurement tools provided by the operating system or the emulation environment.

We will first measure these parameters without using the distributed caching mechanism as a reference point. Then, we will measure them again after implementing the caching mechanism to determine the improvement in the network's topology.

# Budget Estimation

## Equipment:

|  |  |  |
| --- | --- | --- |
| **Name** | **Cost** | **Purchase/ lab equipment** |
| Linux computer x2 |  | Existing lab equipment |

## Software and license

|  |  |  |
| --- | --- | --- |
| **Name** | **Cost** | **Purchase/ lab equipment** |
| Mininet | Free |  |
| ONOS SDN Controller | Free |  |

## Manpower

|  |  |  |
| --- | --- | --- |
| **Name** | **Hours approximation** | **Cost** |
| Student | 700 Hours |  |
| Consultants | 35 Hours |  |

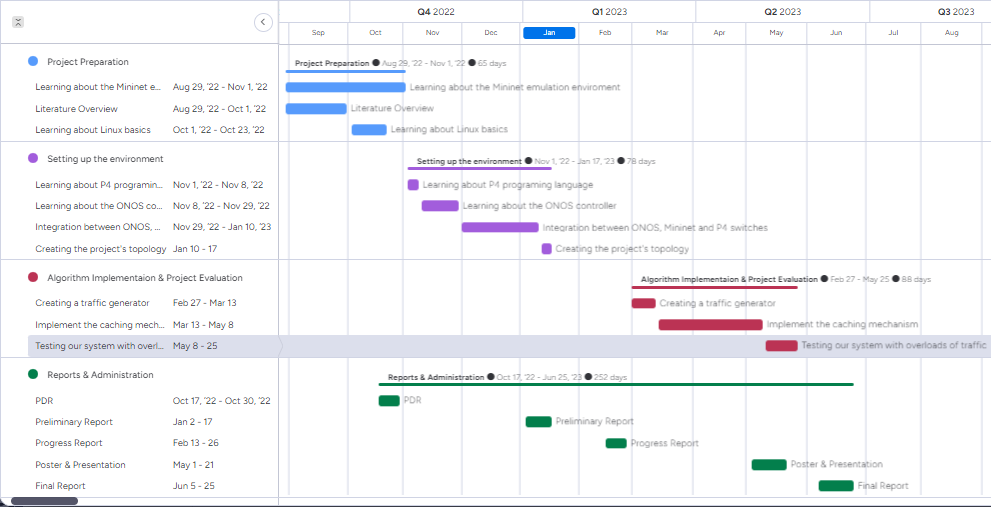
## Overhead: 25%

## Unexpected expenses: 10%

**Total:** in practice, we are not planned to have any expenses, as all necessary equipment is available at the lab.

# Schedule

We scheduled our tasks in a Gantt form as shown below. We will work on the tasks in a parallel manner.



*Figure 3: Gantt chart.*

# References

[1] “Mininet Overview” mininet.org. <http://mininet.org/overview> (accessed: Oct. 26, 2022).

[2] **"P4 Open Source"** P4.org. <https://p4.org/> (accessed: Oct. 26, 2022).

[3] **"P4Runtime Specification"** P4.org. <https://p4.org/p4-spec/p4runtime/main/P4Runtime-Spec.html#tab-p4-type-usage> (accessed: Oct. 26, 2022).

[4] Zuhran Khan Khattak, Muhammad Awais and Adnan Iqbal, **"Performance Evaluation of OpenDaylight SDN Controller"**, [IEEE International Conference on Parallel and Distributed Systems (ICPADS)](https://ieeexplore.ieee.org/xpl/conhome/7092978/proceeding), 2014.

[5] Myung-ki shin, Ki-Hyunk Nam, Hyoung-Jun Kim, "**Software-Defined Networking (SDN) : A Reference Architecture and Open APIs"** , International Conference on ICT Convergence (ICTC), 2012.

[6] “**GitHub: p4lang/behavioral-model**” github.com. <https://github.com/p4lang/behavioral-model> (accessed: June. 26, 2022).

[7] Ola Salman, Imad H. Elhajj, Ayman Kayssi, Ali Chehab, **"SDN Controllers: A Comparative study"** , [18th Mediterranean Electrotechnical Conference (MELECON)](https://ieeexplore.ieee.org/xpl/conhome/7489933/proceeding), 2016.

[8] “**ONOS**” opennetworking.org. <https://opennetworking.org/onos/> (accessed: June. 26, 2022).

# **Appendix**

## טופס כיבוד זכויות יוצרים

**טופס כיבוד זכויות יוצרים וסודיות**

אני מצהיר שלא אעשה שימוש בפרויקט ההנדסי שלי בכל חומר בעל זכויות יוצרים כגון:

טקסט,

תמונה,

אודיו,

וידיאו ,

מוזיקה,

סרט,

אנימציה,

תוכנה

חומרה

תיכנון מעגל

ללא קבלת אישור מראש מבעל הזכויות

אני מצהיר שאשלב בפרויקט ההנדסי שלי בדוחות, סרטונים, והרצאות אינפורמציה שאינה נחלת הכלל

רק בתנאי שאושרה מראש ע"י בעל הזכויות.

הרישום לפרויקט ההנדסי משמש ההתחייבות שלי לקיים ולכבד זכויות יוצרים וסודיות



חתימה:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך 17/01/2023

חתימה:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ תאריך 17/01/2023

## המלצת ציון לדו"ח מכין

**המלצת ציון (ע"י מנחה אקדמי) לדו"ח מכין**

אם יש צורך, לכל סטודנט/ית בנפרד

מספר הפרויקט: \_\_\_\_\_-\_\_\_\_20-P

שם הפרויקט:

שם המנחה החיצוני:

שם המנחה מהמחלקה:

שם הסטודנט/ית: ת.ז.:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| % |  | חלש  55-64 | בינוני  65-74 | טוב  75-84 | ט"מ  85-94 | מצוין  95-100 |  |
| 15 | הבנת הנושא הצורך וסביבת היישום |  |  |  |  |  |  |
| 15 | חיפוש מקורות והבנת עבודות דומות |  |  |  |  |  |  |
| 15 | שלמות דף מפרט (הצעת מחקר) |  |  |  |  |  |  |
| 15 | הצעת תכנון ותכנון הבדיקות הסופיות |  |  |  |  |  |  |
| 10 | גילוי יוזמה וחריצות |  |  |  |  |  |  |
| 20 | פתרון בעיות, מקוריות ותרומה אישית  (מעבר למילוי ההנחיות) |  |  |  |  |  |  |
| 10 | הערכת תקציב, לו"ז וחלוקת עבודה,  ציון מקורות ושלמות כללית |  |  |  |  |  |  |

הערכת רמת הקושי של הפרויקט: קל מאוד / קל / בינוני / קשה / קשה מאוד

הערות: