Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

készítette: **Váradi Richárd Tamás**
ricsi19981007@gmail.com
neptun-kód: **XA5OZH**
ágazat: **Médiainformatika**
konzulens: **Dr. Rétvári Gábor**
retvari@tmit.bme.hu
konzulens:

Téma címe: A Kubernetes és a szolgáltatáshálók hálózati kérdései

Feladat:
A Kubernetes és szolgáltatáshálók hálózati megoldásainak feltérképezése volt. Az egyik olyan probléma, amit én és a konzulensem találtunk, hogy az Istio, mint szolgáltatásháló nem képes UDP (User Datagram Protocol) csomagok fogadására. Viszont az Istio-ban használt Envoy proxy egy újabb verziója már képes UDP csomagokat továbbítani így megvalósítható az is, hogy az Istio-ba ilyen forgalmat irányítsunk be. A félév során egy olyan megoldást kellett találnom, amivel az Istio elé ezzel a proxy-val képes legyek egy úgynevezett átjárót létrehozni. Ennek az átjárónak UDP csomagokat kell tudnia fogadni és átalakítani őket olyan formátumúvá, amit az Istio képes lekezelni.

Tanév: 2019/20. tanév, II. félév

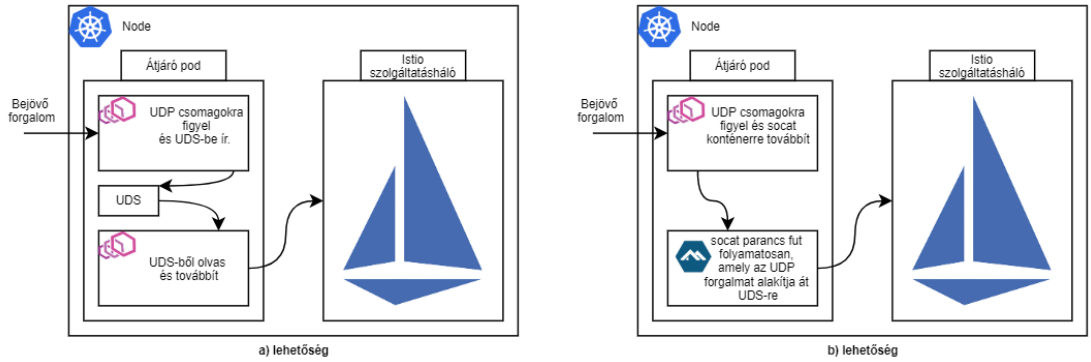
1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1. Bevezető

Az ötlet alapvetően egy GitHub-os hibajegyből született. A GitHub egy webalapú verziókezelő és együttműködésű felület szoftverfejlesztőknek. A hibajegyek úgy születnek, hogy projektekhez lehet őket hozzáírni és erre jó eséllyel a fejlesztők vagy más felhasználók reagálnak. Számunkra most ez a jegy lesz a fontos: https://github.com/istio/istio/issues/1430. Pár példát szeretnék idézni és fordítani, hogy miért kellene ez a támogatottság:

- Publikus felhők, ahol egy titkosított szolgáltatáshálót hoznának létre, amely képes kezelni olyan dolgokat, mint a DNS (Domain Name System) és NTP (Network Time Protocol).
- IoT (Internet of Things) területen.
- Telekommunikáció terén is jó lehet a 3G-től használják az UDP protokollt adattovábbításra.

Ennek megvalósítására a konzulensemmel úgy gondoltuk, hogy érdemes lenne egy átjárót felépíteni az Istio elé, amelyet az alábbi ábrákon be is mutatok.



Az **a** lehetőség nem teljesen biztos, hogy megvalósítható az Envoy korlátai miatt. Az Envoy egy nyílt forráskódú perem és szolgáltatás proxy, ami natív felhőalkalmazásokhoz lett írva C++-ban. Ez a megvalósítás elméleti szinten működőképes lenne, viszont nagyon kevés dokumentáció található arról, hogy az UDS (Unix Domain Socket) pontosan hogyan működik az Envoy-ban.
A **b** megoldás annyiban egyszerűbb, hogy ott kettő konténert hozunk létre egy pod-ban, ahol az Envoy UDP csomagokat figyel és UDP csomagokat továbbít egy Alpine alapú konténernek, amiben fut egy socket parancs, ami képes UDP forgalmat UDS-re irányítani. A Pod a kubernetes legkisebb egysége, amelyekben konténereket lehet létrehozni. A konténereket érdemes úgy létrehozni, hogy konténerenként egy folyamat fusson benne. Az Alpine egy nagyon kicsi erőforrás igényű unix alapú operációs rendszer.

1.2. Elméleti összefoglaló

1.2.1. Docker

A Docker egy szolgáltatáskészlet Paas (Platform as a Service) termékcsalád, amelyek operációs rendszer szintű virtualizációt végeznek, hogy a szoftvert csomagokban, úgynevezett konténerekben lehessen létrehozni.

A konténerek egymástól elválasztva és saját szoftverüket futtatva léteznek külön könyvtárakkal és konfigurációs fájlokkal. Ezek a konténerek képesek kommunikálni egymással jól definiált csatornákon keresztül. Érdemes azt az elvet szem előtt tartani, hogy minden konténerben csak egyetlen folyamat fusson így sokkal egyszerűbb esetleges hibánál a forrást megtalálni és javítani.

Minden konténer egyetlen operációs rendszer kernelt használ, ezért sokkal kevesebb erőforrást igényelnek, mint a rendes virtuális gépek. Viszont olyan hátulütője van ennek a tulajdonságának, hogy mondjuk a

kiszolgáló egy linux operációs rendszer, akkor csak linux alapú konténerek hozhatóak létre, míg egy windows alapú kiszolgálónál csak windows alapú konténerek születhetnek. Bár az utóbbi állítás a Hyper-V újítása révén már nem probléma érdekességként ajánlom a Microsoft dokumentációját erről [1]

Ezek a konténereket képfájlokból lehet kialakítani, amiket saját magunk is megírhatunk, de böngészhetünk is közülük a DockerHub-on [2], amely hasonlóan működik, mint a GitHub.

Mikor egy ilyen képfájlt létrehozunk, akkor mindig kell egy alap képfájl, amelyet az említett DockerHub oldalról letudunk tölteni a docker alkalmazás segítségével.

További információ gyűjtésére erről a témáról a következő oldalakat ajánlom: [3] [4]

1.2.2. Kubernetes

A kubernetes egy nyílt forráskódú rendszer, amely a fejlesztés automatizálására, skálázásához és konténer alapú alkalmazások kezelésére való. Eredetileg a Google fejlesztette, de jelenleg a CNCF (Cloud Native Computing Foundation) tartja karban.

Azért érdemes használni, mert mikroszolgáltatások lehet benne létrehozni, amelyek tudnak egymással kommunikálni, de a hálózati megvalósítások nem ezekbe a szolgáltatásokban kell létrehozni, mert a kubernetes erről gondoskodik nekünk olyan hálózati technológiákkal, amelyek megtalálhatóak egy átlagos hálózatban is. Ilyen például a DNS, Routing táblák, IP (Internet Protocol) táblák.

A következőkben ismertetem a két legalapvetőbb komponenst, amelyek a Pod és a Node.

A **Pod** a legkisebb létrehozható objektum alkalmazás fejlesztésére. Egyetlen Pod egy futó folyamatot reprezentál a klaszterünkben és egy vagy több Docker konténert tartalmaz, amelyek saját tárhelyez igényel és egyedi IP címet. Ezek a konténerek úgy lettek tervezve, hogy ugyanazon a gépen helyezkedjenek el és legyenek egyszerre ütemezve.

A klaszter úgynevezett dolgozó gépek gyűjteménye, amelyeket Node-oknak nevezünk. Minden klaszternek legalább egy dolgozó node-ja van.

Mint már említettem egy node egy klaszterben dolgozó gépet reprezentál, ezek lehetnek fizikai gépek, virtuális gépek vagy bármi más. Esetünkben ez egy virtuális gép lesz, melyet a Minikube nevezetű alkalmazás fog számunkra biztosítani. További információ a Minikube-ról. [5]

Most, hogy a számunkra fontosabb részeket ismertettem áttérek a kubernetes hálózati modelljének bemutatására. Kezdetben három tulajdonságát szeretném bemutatni.

- Az összes pod képes kommunikálni a hálózatban megtalálható összes pod-al NAT (Network Address Translation) használat nélkül
- Az összes node képes kommunikálni az összes pod-al NAT nélkül
- Amilyen IP címet lát a pod a saját interfészéhez rendelve, ugyan azt a címet fogja látni más pod vagy node is a hálózatban.

Így a következő hálózati kihívások jelentkeznek:

1. Konténer -> Konténer adattovábbítás
2. Pod -> Pod adattovábbítás
3. Pod -> Szolgáltatás adattovábbítás
4. Internet -> Szolgáltatás adattovábbítás

Ami most számunkra fontosabb lesz az az első kettő pont.

Itt kicsit jobban fejtsd ki ez alapján:
https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/

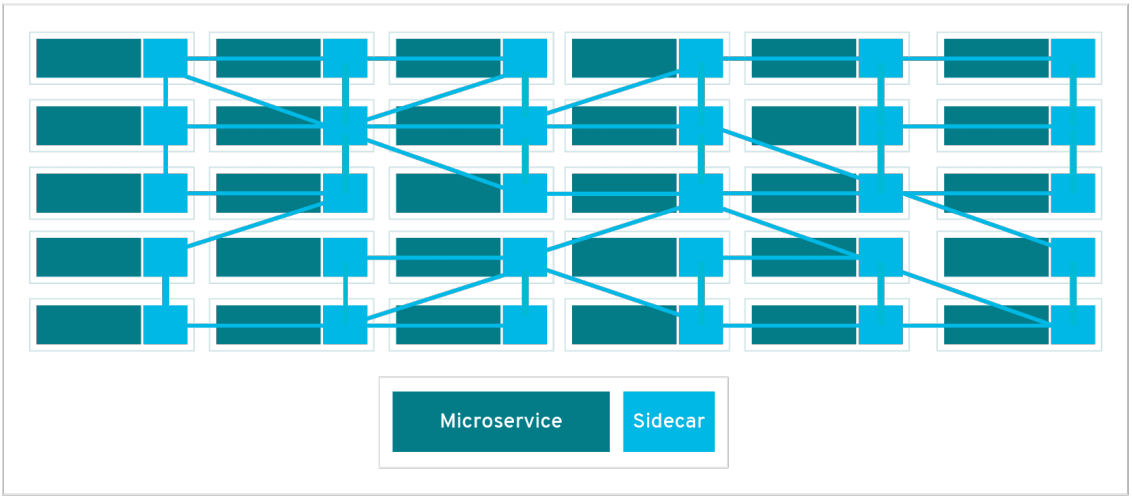
1.2.3. Szolgáltatásháló

A szolgáltatásháló, olyan mint mondjuk az Istio egy módja annak, hogy az alkalmazás különböző részei, hogyan osztanak meg adatot egymás között. Más rendszerekkel ellentétben a kommunikációra a szolgáltatásháló egy dedikált infrastruktúra réteg magában az alkalmazásban. Ez egy "látható" vagyis érzékeljük, hogy ott van, miközben a különböző részek kommunikálnak egymással és így láthatjuk, hogy ezek

a komponensek hogyan működnek vagy nem és így egyszerűbbé válik a kommunikáció optimalizálása és elkerülhető a kiesett idő, amíg esetleg az alkalmazásunk nem üzemel.

Minden részét az alkalmazásnak egy szolgáltatásnak hívunk, amelyek különböző folyamatokért felelősek.

Legjobban úgy lehet megérteni, hogy hogyan működnek a szolgáltatáshálók, ha ezt egy ábrával tesszük.



Minden szolgáltatás egy mikroszolgáltatásból és egy sidecar proxy-ból áll. Mikroszolgáltatás alatt az alkalmazás egy adott funkciójának megvalósítását valamilyen nyelven értjük. Fontos kihangsúlyozni, hogy nem kötött milyen programozási nyelven van az adott mikroszolgáltatás írva, mert ez is nagyon rugalmassá teszi az ilyen fejlesztést, abban az esetben, ha különböző csapatoknak más és más nyelven akarnak megírni egyes szolgáltatásokat. Viszont itt jön képbe a sidecar proxy, ami elvégzi az a hálózati kommunikációt a szolgáltatások között. Szóval nem kell a fejlesztőnek a mikroszolgáltatásban azzal foglalkozni, hogy kiknek címezze a kimenő forgalmat vagy, hogy honnét kapja, mert ez mind a két esetben a sidecar lesz.

A kép forrása és további érdekes információk megtalálhatóak még a Red Hat ezen cikkében [6]

1.2.4. Envoy

Ez egy L7 (alkalmazás) rétegen működő proxy és kommunikációs csatorna arra tervezve, hogy nagy szolgáltatás alapú architektúrákat lehessen létrehozni úgy, hogy a szolgáltatások számára a hálózat átlátszó legyen. A proxy egy olyan szerver, amely hálózati forgalmat irányítja. Az előzőleg említett funkcióját úgy sikerül ellátnia, hogy a szolgáltatáshálók részben lévő szolgáltatásoknál az Envoy lesz maga a sidecar, így a mikroszolgáltatásnak elég mindig csak a localhost-ra küldeni és onnét fogadni adatot.

Képes L3/L4 (hálózati/szállítási) réteg szűrőket is alkalmazni, amivel a különböző protokollokat, mint a TCP (Transmission Control Protocol) és az UDP.

Ezen felül rendelkezik még olyan funkciókkal, amelyek segítségével megvalósítható perem proxy-ként és belső proxy-ként is, felügyelhető, és HTTP (HyperText Transfer Protocol) alapú szűrést is tudunk használni.

További információk ezen az oldalon találhatóak [7]

1.2.5. További technológiák, amelyek szükségesek

UDP - User Datagram Protocol

Az UDP egyszerű kapcsolatmentes összeköttetést hoz létre 2 eszköz között, így nincs kapcsolat ellenőrzés, mint a TCP-nél ezért sokkal gyorsabban felállítható a kapcsolat eszközök között. Viszont nincs garancia arra, hogy a csomagok megérkeznek, sorrendben érkeznek-e vagy, hogy duplikált csomagok jönnek át a csatornán. Ezért olyan szolgáltatásokhoz érdemes használni, ahol nem probléma, ha egy csomag többször jön vagy hiányoznak csomagok. Ilyenek például a telekommunikáció és az élő közvetítések is. További információk találhatóak ezen az oldalon [8]

UDS - Unix Domain Socket

Arra használatos, hogy a folyamatok úgy tudjanak egymással kommunikálni, hogy az minél jobb legyen. A

Unix Domain Socket lehet névtelen, vagy hozzárendelve egy fájlrendszerben létrehozott elérési útvonalhoz. További információkat lehet találni ezen az oldalon [9]

iPerf

Az iPerf egy olyan eszköz, amellyel lehet mérni a maximálisan elérhető sávszélességet IP hálózatokban. Támogatja a különböző protokollokat, így használható TCP és UDP mérésre is, de ezen felül lehet még sok máshoz is használni. De betudunk állítani olyanokat is, hogy adott ideig és időközökkel küldjön általunk megadott csomagmérettel és küldési sebességgel, szóval egy nagyon kis hasznos eszköz.

Minden ilyen teszt végén kapunk egy jelentést arról, hogy milyen sávszélességgel érkeztek meg a csomagokat, ebből mennyi veszett el és még más paramétereket is annak tükrében, hogy milyen részletes jelentések szeretnénk látni.

A használata úgy néz, hogy létre kell hozni egy úgynevezett iPerf szervert, amihez beérkeznek a csomagok. Ez lehet bárhol, amit az hálózaton keresztül elérünk. Ezen felül még létre kell hozni egy iPerf klienst is, aki az általunk specifikált módon fog csomagokat küldeni a szerverre.

További érdekes információk találhatóak az weboldalán [10]

socat

A Socat egy parancssori alapú segédprogram, amely két kétirányú bájt-folyam hoz létre és adatot továbbít közöttük. Mivel az adatfolyamok különféle típusú adatgyűjtőkből és forrásokból állíthatóak össze, és mivel sok címbeállítási lehetőséget lehet alkalmazni az adatfolyamokra, a socat felhasználható többféle célra.

Ezt fogjuk arra használni, hogy UDS csomagokat alakítsunk át UDP csomagokká. További információk találhatóak a következő oldalon [11]

1.3. A munka állapota, készültségi foka a félév elején

A témalaboratórium alkalmával már foglalkoztam a szolgáltatáshálók megismerésével és használatával, ahol az Isio és az SMI (Service Mesh Interface) különböző aspektusait hasonlítottam össze. Ennek során mélyebb rálátást sikerült nyernem a Kubernetes használatára és arra, hogy egy jó dokumentáció milyen nagy segítséget nyújthat azoknak, akik ebben az iparágban dolgoznak.

2. Az elvégzett munka és az eredmények ismertetése

2.1. A munkám ismertetése logikus fejezetekre tagoltan

<Én magam (nem a társam) a félév során következőket olvastam el / programoztam / készítettem el / teszteltem / dokumentáltam / néztem át / tanultam meg, stb. Tételes leírása és felsorolása mindannak, ami a félév során történt, alátámasztandó azon állításom a konzulens/tárgyfelelős felé, hogy összességében mindent beleértve tényleg dolgoztam a TVSZ szerint kreditenként 30 órát, azaz a heti 2 kontakt órás tárgy esetében min. $2,5*30 = 75$ munkaórát, illetve a heti 6 kontakt órás tárgy esetében min. $8*30 = 240$ munkaórát...>

Ebben a részben a hallgató az általa elvégzett munkát mutatja be. Hangsúlyosan a saját munka bemutatása a cél, hiszen a hallgató ezzel igazolja a témavezető és a tárgyfelelős irányába, hogy – folyamatosan fejlődve és egyre több és jobb munkát végezve – a szakdolgozatát/diplomadolgozatát képes lesz megírni. A beszámoló nem munkanapló, nem arra vagyunk kíváncsiak, hogy mit mikor csinált a hallgató és mennyi időt töltött vele, hanem egy eredmény-centrikus beszámolót szeretnénk olvasni. De itt is fontos tudni, hogy megosztott feladat esetén ki-mit csinált, mekkora részt vállalt.

Az egész beszámoló elkészítésénél törekedni kell a magyar nyelv szabályainak követésére és a műszaki dokumentáció/tudományos közlemény írásával kapcsolatosan kialakult közmegegyezés szerinti formai követelmények betartására. (Tehát nem kell többes számként hivatkozni saját magunkra, kerülni kell a furcsa megfogalmazást, passzív és egyéb kifacsart mondatszerkezeteket. Az egy szót határozatlan névelőként történő használatakor ne írjuk ki számként.)

A beszámoló természetesen nem csak szöveget tartalmazhat, hanem képleteket, táblázatokat, ábrákat és még sok minden mást. Ezek kapcsán az alábbi elvek irányadók:

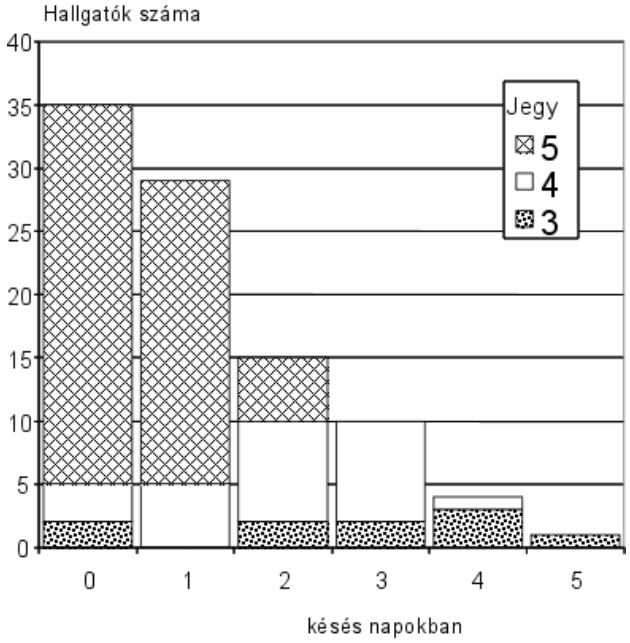
- Az ábráknak, képeknek és táblázatoknak mindig van számuk és címük. (A cím nem ennyi: „1. ábra”, hanem azt írd le, ami látható rajta.)
- Az ábrákra, a képekre és a táblázatokra a szövegben hivatkozni kell, és a szövegben elemezni kell azokat. Például az 1. ábrán látszik, hogy a vizsgált félévben még két napos csúszással is lehetett jeles érdemjegyet szerezni a tárgyból, de a pontosság még nem garancia a jó jegyre: öten nem kaptak jelest, noha nem késtek a leadással.
- Az ábrák, képek és táblázatok mérete a szükségesnek megfelelő legyen: elég nagy ahhoz, hogy kinyomtatva is olvasható és értelmezhető legyen, de nem nagyobb annál, mint amit szerepe indokol.
- A grafikonoknak a tengelyeken legyenek feliratai és ha releváns, a mértékegység is.
- A képletek esetében nem minden képletre történik hivatkozás, de ahol igen, ott a képletet a műszaki irodalomban jellemző módon a sor végére tett kerek zárójelben lévő számmal jelöljük meg. A képleteket ne képként illeszd be a szövegbe.
- Kódrészleteket, ha nem relevánsak, ne illeszd be képként, főleg ne rossz minőségben. Nyugodtan teheted függelékbe és hivatkozd be a szövegben, mint a képeket, például: Az 1. számú függelékben található az adatbeolvasó kód, melyet C++ nyelven készítettem el.

Az írásbeli beszámolót a témavezető és a tárgyfelelős is értékeli. A tárgyfelelősi értékelés szempontjai az alábbiak:

1. Megfelel-e az elvégzett munka a félév elején kiadott feladatnak?
2. Megfele-e a beszámoló a formai követelményeknek? Ezen belül:
    - a. Megfelelő-e az elméleti bevezető és az irodalomjegyzék?
    - b. Egyértelmű-e, hogy mi volt a hallgató saját munkája?
    - c. Megfelelő-e a dokumentum technikai színvonala?

Ezen kívül a tárgyfelelős veszi figyelembe az értékelés során kialakult félévi jegyre vonatkoztatva az ún. „hanyagsági faktor” értékét, amelyet (1) szerint állapítunk meg:

$$F_{hany} = 1 - a - b \tag{1}$$



1. ábra. Hallgatók érdemjegyeinek eloszlása az írásbeli beszámoló késése függvényében

Az írásbeli beszámoló beadásának napja a szóbeli beszámolóhoz képest (munkanapban)	A „b” faktor értéke
-4. munkanap	0.04
-3. munkanap	0.09
-2. munkanap	0.20
-1. munkanap	0.30

1. táblázat. Az írásbeli beszámoló késedelmes beadásával kapcsolatos hanyagsági faktor értéke

Az (1)-ben szereplő a szám a munkaterv beadásában történt késedelemre, míg a b szám az írásbeli beszámoló beadásában történt késedelemre vonatkozik. Utóbbi értékeiről az 1. táblázat tájékoztat.

A beszámoló értékeléséről részletesebben írunk [13]-ban.

A beszámolóban bizonyára szerepelni fognak rövidítések. Ezeket a rövidítéseket, betűszavakat néhány, az infokommunikáció területén nagyon ismert és gyakran használt kifejezéstől (például IP, TCP, GPRS, UMTS) eltekintve ki kell fejteni logikusan az első használat alkalmával (például így: „A GPS (Generalized Processor Sharing) egy ideális folyadékmodellen alapuló csomagütemező eljárás.”).

A beszámoló készítése során előfordulhat, hogy a hallgató úgy érzi, hogy alfejezetekkel tagolva jobban olvasható és érthető lenne a beszámoló. Ennek akadálya nincs, de érdemes arra figyelni, hogy a túlzott tagolás sem tesz jót egy írásműnek, illetve hogy a címsorokban a rövidítések és a hivatkozások használata tilos. Tartalomjegyzéket készíteni nem szükséges a beszámolóhoz, de nem is tilos, kivéve azt az esete, amikor nyilvánvalóan terjedelemnövelési célokat szolgál.

A beszámoló terjedelme tárgyanként változhat. Általános szabály, hogy 1 hüvelyknél nagyobb margókat ne használjunk. A szöveg legyen egyszeres sortávú, sorkizárt és 12 pontos betűméretű. A bekezdések kezdődjenek behúzással a minta szerint.

2.2. Összefoglalás

A félévi munka során elért új eredmények ismételt, vázlatos, tömör Ebben a részben az adott félévre vonatkozó, az *Önálló laboratórium tárgy keretében elvégzett munka során* elért **új** eredmények ismételt, vázlatos, **tömör** összefoglalását várjuk, lehetőleg nem felsorolásként. Itt még egyszer ki lehet térni a leglényegesebb eredményekre, valamint a félév során felmerülő nehézségekre, de meg lehet említeni a továbbfejlesztési irányokat, lehetőségeket is.

Ezt a részt tagolható a következő pontok megválaszolásával:

- Mi volt az **aktuális kérdés**, probléma, amivel a félév során foglalkoztál?
- Mi a dolgozat **célja**, miért érdekes egyáltalán ezzel a problémával foglalkozni?
- Milyen **módszereket** használtál a probléma megoldása érdekében?
- Mik a legfontosabb **eredmények**?
- Milyen **következtetéseket** lehet levonni?

Ha valaki elolvassa ezt a részt, képet kell kapnia az egész dolgozatról. Ne legyen az absztrakt szó szerinti ismétlése.

Fontos, hogy az itt megadott sablontól el lehet térni, használata nem kötelező, csak segítséget jelenthet, viszont a fedőlap lehetőleg maradjon ugyanez és tartalmilag egyezzen meg a sablon irányelveivel. A beszámoló felépítésében nem érdemes eltérni a *Bevezető – Féléves munka és eredmények bemutatása – Összefoglaló* hármastól.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

3.1. A tanulmányozott irodalom jegyzéke

[1] Microsoft Corporation, *Linux containers on Windows 10* https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers

[2] Docker Inc, *DockerHub* https://hub.docker.com/

[3] Docker Inc, *Docker Documentation* https://docs.docker.com/

[4] Wikipedia contributors, *Wikipedia:Academic use*, Wikipedia, The Free Encyclopedia, 2011 Nov 11. Available from: https://en.wikipedia.org/wiki/Docker_(software)

[5] The Kubernetes Authors, *Minikube documentation* https://minikube.sigs.k8s.io/docs/

[6] Red Hat Inc, *What's a service mesh?* https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

[7] Envoy Project Authors, *What is Envoy* https://www.envoyproxy.io/docs/envoy/latest/intro/what_is_envoy

[8] Wikipedia contributors, *User Datagram Protocol* https://en.wikipedia.org/wiki/User_Datagram_Protocol

[9] Linux man-pages project, *Unix Domain Socket* http://man7.org/linux/man-pages/man7/unix.7.html

[10] iPerf contributors, *iPerf* https://iperf.fr/

[11] Gerhard Rieger, *socat - Linux man page* https://linux.die.net/man/1/socat

[12] Esterházy Péter, *Termelési-regény (Kisssregény),* Magvető Könyvkiadó, 2004, ISBN: 9631423948.

[13] *Tájékoztató a Műszaki Informatika Szak önálló laboratórium tantárgyainak 2008/9. tanév I. félévi lezárásáról a BME TMIT-en (VITMA367, VITMA380, VITT4353, VITT4330),* http://inflab.tmit.bme.hu/08o/lezar.shtml, szerk.: Németh Felicián, 2008. november 5.

[14] Wikipedia contributors, *Wikipedia:Academic use*, Wikipedia, The Free Encyclopedia, 2011 Nov 11. Available from: http://en.wikipedia.org/w/index.php?title=Wikipedia:Academic_use&oldid=460041928

Itt jegyezném meg, hogy a tanulmányozott irodalmat hivatkozni kell a szövegben. Szükség esetén többször is. Az irodalomjegyzék célja (lásd a 3.1 fejezetet) ugyanis kettős[1]:

1. Az olvasó tájékoztatása, hogy a dokumentumban ki nem fejtett dolgoknak, a tudottnak vélt ismereteknek hol lehet bővebben utánanézni, így ott kell meghivatkozni az irodalmat [**?**, 12], ahová az irodalom kapcsolódik.

2. Megmutatni a tárgyfelelosnek/konzulesnek az elolvasott irodalom mennyiségét

Javasoljuk, hogy a hallgatók tanulmányozzák, hogyan néznek ki a hivatkozások a villamosmérnöki/informatikai szakma vezető szakmai folyóirataiban megjelenő cikkekben. Ebben a témavezető is biztosan tud segíteni. A hivatkozás teljességére és egyértelműségére tessék ügyelni. Például, ha egy könyvnek több, eltérő kiadása is van, akkor azt is meg kell jelölni, hogy melyik kiadásra hivatkozunk. A webes hivatkozások problémásak szoktak lenni, de manapság egyre több az olyan dokumentum, ami csak weben lelhető

[1] Akárcsak ennek a fejezet hivatkozásnak, ami a `\aref babel` parancsot demonstrálja

fel, ezért használatuk nem zárható ki. Itt is törekedni kell azonban a pontosságra és a visszakereshetőségre. A weben található dokumentumoknak is van címe, szerzője, illetve érdemes megadni a letöltés/olvasás időpontját is, hiszen ezek a dokumentumok idővel megváltozhatnak.

A wikipédiás hivatkozások használata nem javasolt, mert a wikipedia másodlagos forrás. Tájékozodjuk a wikipédián, de aztán olvassuk el az adott oldalhoz megadott hivatkozásokat is. A wikipedián külön szócikk foglalkozik azzal, hogy miért nem szerencsés tudományos munkákban a wikipédiára hivatkozni [14].

Nem publikus dokumentumok hivatkozása nem javasolt és csak kivételes helyzetben elfogadható!

3.2. A csatlakozó dokumentumok jegyzéke

<A munka ezen beszámolóba be nem fért eredményeinek (például a forrás fájlok, mindenképpen csatolni akart forráskód részlet, felhasználói leírások, programozói leírások (API), stb.) megnevezése, fellelhetőségi helyének pontos definíciója, mely alapján a az erőforrás előkereshető – értelemszerűen nem nyilvános dokumentumok hivatkozása nem elfogadható.>