Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Integrált tömegfelügyeleti rendszer okos városokban

Készítette
Nagy Attila Mátyás

Konzulens
dr. Simon Vilmos

2015. október 25.

Tartalomjegyzék

1. fejezet

Tömegfelügyelet

Manapság a tömegrendezvények (fesztiválok, felvonulások, vallási, politikai vagy sportművelődési rendezvények, stb.) óriási látogatottságnak örvendenek, ugyanakkor számtalan veszélyt is hordoznak magukban, hiszen nagy tömegekben már egy kisebb pánik - amelyek megjóslása, megelőzése komoly feladat - is beláthatatlan katasztrófát okozhat, annak ellenére, hogy a szervezők mindent megtesznek a résztvevők biztonságáért. Ezek a váratlan helyzetek bárhol és bármikor kialakulhatnak, melyek a rendfenntartó szervektől a lehető leggyorsabb reakciót kívánják. Minden egyes alkalommal más-más biztonsági kérdésekkel kell megbirkózniuk, ami egy több tízezres, vagy akár százezres tömeg esetén rendkívül nehéz feladat és nagy felelősség. Hogyan lehetne megelőzni egy tömegpánik kialakulását? Hogyan tudnánk hatékonyan detektálni, hogy a tömeg kritikus méretűre duzzadt? Esetleg, hogyan lehetne megelőzni a korábbiakhoz hasonló esetek kialakulását például tömegdinamikai predikció segítségével?

1.1. ábra. *2010-ben a Németországban megrendezett Love Parade fesztiválon 21-en vesztették életüket az alagút bejáratánál.*

Sajnos minden évben előfordulnak tömegkatasztrófák, melyeknek egy része megelőzhető lenne, ha a szervezők számára több információ állna rendelkezésre, bár vannak olyan előre nem megjósolható események is, amelyekre nagyon nehéz és körülményes felkészülni. Talán a leghírhedtebb katasztrófák Mekkában, a Hajj idején történtek. 1990-ben 1492-en, 2006-ban 432-en és idén több mint 1636-an vesztették életüket, amikor a túl nagy tömegben az emberek összenyomták és eltaposták egymást. Indiában az elmúlt tíz évben közel két évente fordulnak elő hasonló katasztrófák, ahol több százan halnak meg. Sajnos Európában is történt már ilyen tragédia, 2010-ben Németországban az évente megrendezett Love Parade-on 21 ember halt meg és 500-an megsérültek, amikor egy alagútnál összezsúfolódott a tömeg. Valószínűleg ahogy folyamatosan nő a különböző típusú rendezvények száma és mérete, egyre gyakoribbá fognak válni a hasonló szerencsétlenségek, ha nem teszünk ellene valamit. A szervezőknek szükségük van olyan támogató rendszerekre, szoftverekre, amelyek segítségével jobban megfigyelhető, megérthető a tömegek mozgása. Az új információk alapján gyorsan és célzottan lehet beavatkozni, így akár életeket is mentve.

A mai tömegrendezvények látogatóinak túlnyomó többsége már rendelkezik olyan mobil készülékkel (okostelefonnal, tablettel), amelyekben különféle szenzorok (GPS, giroszkóp) találhatóak meg. A szenzorokból származó adatok összegyűjthetőek és ezekből rengeteg információ kinyerhető, így lehetőségünk van monitorozni a tömeg dinamikáját, mozgását, akár becsléseket készíteni a jövőbeli állapotokról. Ez több szempontból is nagyon hasznos lehet. Egy nagy rendezvény esetén, bár vannak elképzeléseink és pontatlan becsléseink a tömeg eloszlásáról, nem tudjuk pontosan, hogy egy területen mennyi ember tartózkodik egy adott pillanatban. Ha ezt mérni tudnánk egy mobil alkalmazáson keresztül, a tömeg eloszlásától függően képesek lehetnénk a rendfenntartó egységek helyesebb átcsoportosítására és dinamikusan tudnánk módosítani az evakuációs terveket, ha a szükség úgy kívánja. A mobil alkalmazás továbbá lehetővé teszi, hogy a résztvevők számára, pozíciójuk alapján különböző üzeneteket küldjünk. Ezek az üzenetek lehetnek közérdekű felhívások, de akár promocionális üzenetek is.

Abban az esetben, ha az applikáció népszerűvé válik fel kell készülnünk nagy adatmennyiség beérkezésére. Ez egyrészt azt is jelenti, hogy nagyszámú párhuzamos kapcsolat kezelésére kell képesnek lennie a rendszernek, hiszen könnyen előfordulhat, hogy egy százezres tömegből egy időpillanatban több ezren vagy akár tízezren is küldenek be adatokat, másrészt pedig olyan rendszerre van szükség, ami képes hatékonyan feldolgozni ezt a nagy mennyiségű mérést, akár szerverfürtökön elosztva, mivel az eredményekre valós időben van szükség. Az előzőeket megfontolva nem kifizetődő folyamatosan adatokat gyűjteni minden egyes felhasználótól, mert ez kezelhetetlen méretű adathalmazt hozna létre, rengeteg szükségtelen állapotot is tárolna a rendszer, a készülékek akkumulátora nagyon hamar lemerülne és elképesztő méretű hálózati forgalmat is generálna. Jobb ötletnek tűnik, ha kevesebb adatot gyűjtünk, viszont azok releváns információkat tartalmaznak.

A megszerzett ismeretek mit sem érnek, ha azokat nem vagyunk képesek az ember által

könnyen értelmezhető formában bemutatni. Emiatt nagyon fontos, hogy a rendszer ergonomikus és átlátható felületeken logikusan jelenítse meg a kinyert információkat, hogy a szervezők, rendfenntartók minél gyorsabban átlássák a kialakult helyzetet, jó rálátásuk legyen az aktuális állapotra. Így képesek időben reagálni például a tömeg növekedésére, értesíteni tudják a közelben lévő kollégáikat akár GPS koordinátákat is mellékelve, akik a helyszínen azonnal be tudnak avatkozni, megakadályozva a tömeg kritikus érték fölé emelkedését.

Hasznos funkció lehet az is, ha vissza tudjuk játszani a tömegdinamikai állapotokat, hasonlóan egy videófelvételhez. Így a beérkezett adatokat újra és újra ki tudjuk értékelni és következtetéseket vonhatunk le például abból, hogy mik voltak az adott alkalommal a leginkább látogatott helyek, hol voltak azok a pontok, ahol a legtöbb látogató haladt át, hol milyen irányba mozogtak az emberek. Egyrészt biztonsági szempontból ezeket a területeket kiemelt fontossággal kell kezelni a következő alkalommal, másrészt a kinyert információk segítségével a résztvevők kényelmét, komfortérzetét is növelhetem. A funkció az új tudományos modellek kiértékelésénél is felhasználható, így empirikus teszteket is lehet végezni egy modell helyességének vizsgálatánál.

1.1. Tömegfelügyeleti rendszerek

Természetesen léteznek már olyan rendszerek, amelyek különböző felügyeleti feladatokat látnak el, viszont ezek nem feltétlenül a bevezetésben megfogalmazott feladatokkal foglalkoznak, vagy más módon oldják meg a problémát.

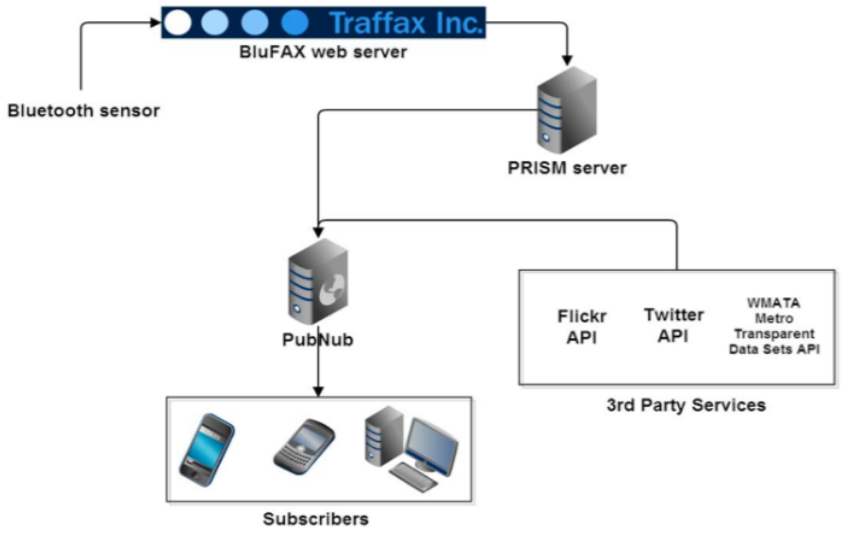
Manapság a kamera alapú felügyeleti rendszerek a legelterjedtebbek a tömegfelügyeleti rendszerek esetében. Egy 2011 márciusában született cikk [1] szerint az Egyesült Királyság területén eddig 1,85 millió kamerát telepítettek, bár ezek közül 1,7 millió privát rendszerekben teljesít szolgálatot. Rendkívül sok helyzetben alkalmazzák őket, természetesen nem csak rendezvények megfigyelésére. A városi gyalogos forgalom vagy járműforgalom vizsgálatával, tanulmányozásával hatékonyabbá tehetjük a létező rendszereket. Autópályák esetén a kamerarendszer segítségével egyből észrevehetőek a balesetek, hamarabb lehet értesíteni a mentőket, rendőröket, a baleset mögött haladó forgalom egy részét el lehet terelni, így csökkenthető a dugók mérete, kialakulásának valószínűsége, de egy átlagos napon is a kamerák adatainak feldolgozásával hasznos információkat lehet közölni a vezetőkkel. Meghatározó szerepe lehet még a bűnmegelőzésben és bűnüldözésben is. A kamerák segítségével a megfigyelt tömegben felfedezhetőek a körözött személyek, vagy egy bűntényről készült videó felvétel kulcsfontosságú információkat tartalmazhat az ügy megoldásához vagy a bíróságon terhelő bizonyíték lehet a tettessel szemben. A kamerás rendszerek jelentős hiányossága, hogy nem képesek egy átfogó képet adni az aktuális állapotról, tehát hiába helyezünk el sok kamerát egy esemény területén, azok képeiből nem tudunk egy aggregált képernyőképet mutatni, illetve csak durva becsléseket tudunk adni vele a tömeg méretére. Ennek ellenére érdemes továbbra is használni, akár drónokra szerelve őket, így a kamerákat dinamikusan

mozgatva, elosztva a rendezvény területén, annak függvényében, hogy hol van szükség rájuk.

A szakirodalomban felfedezhető Bluetooth alapú megoldás is. A rendszer [2] fejlesztésénél az volt a cél, hogy egy olyan mobil alkalmazást készítsenek, ami segítséget nyújt a menedzsmentben és a kommunikációban a szervezőknek egy nagyméretű szabadtéri eseményen. Ezeken a rendezvényeken a kritikus információk az elsősegély pontokhoz, az emberek irányításához vagy a mobilitáshoz (parkolók, ajánlott útvonalak) kapcsolódnak. Fő céljaik a következőek voltak:

- Tömeg méretének, sűrűségének, mozgásának mérése automatizmusok segítségével;
- A mért adatok transzformálása a szervezők számára hasznos információvá;
- Biztosítani egy dedikált kommunikációs csatornát a hatóságoknak és a szervezőknek, hogy értesíteni tudják az eseményen résztvevőket;
- Megbizonyosodni, hogy a kommunikációs kapcsolat robosztus és hibavédett, hogy elérhető legyen vészhelyzet esetén is amikor a celluláris kommunikáció esetleg csődöt mond.

A tömeg méretének, eloszlásának érzékeléséhez a fentiekben bemutatott rendszer a Traffax által fejlesztett szenzorokat használja [3]. A berendezéseket eredetileg jármű forgalom méréséhez tervezték, de alkalmas gyalogos forgalom vizsgálatára is. A rendszer korlátja, hogy Bluetooth-os érzékelő kapukat csak a ki és bejáratok mellé helyezték el, így a fesztivál belső területeinek állapotát nem tudták megfigyelni, de ennek az oka az lehet, hogy sok ilyen kapu lehelyezése módfelett költséges lenne.



1.2. ábra. *A Bluetooth alapú megoldás rendszer komponensei. [2]*

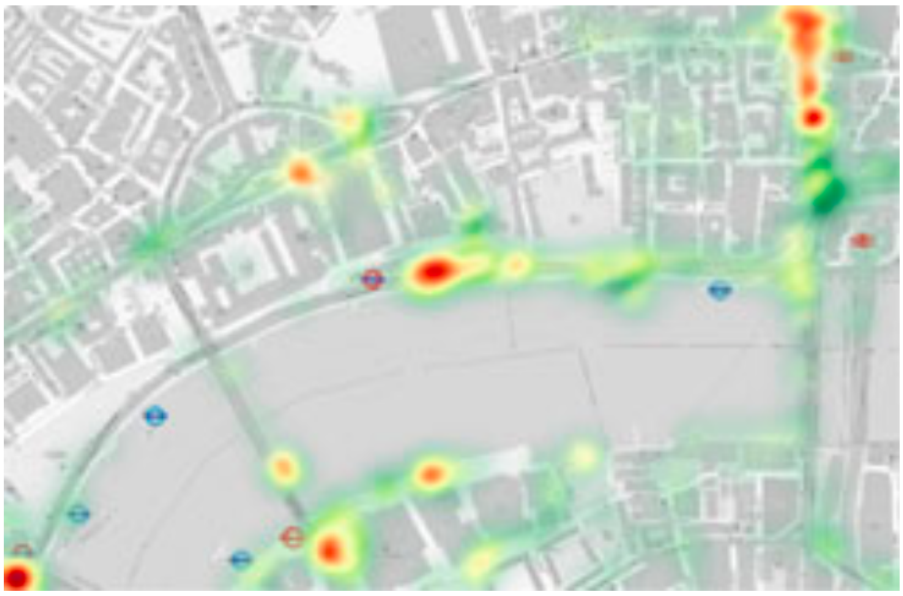
A következő rendszer [4] teljesen más értelmezésben és méretekben közelíti meg a felügyeleti rendszer fogalmát. Míg az előző egyértelműen egy tömegrendezvény biztonságosabbá tételével foglalkozott, addig ez a rendszer nem egy ekkora méretű eseményre koncentrál, hanem arra, hogy megoldható-e, hogy egy nagyobb területet, például egész megyéket, országokat figyeljünk meg adott pillanatban és detektálhatunk-e különböző vészhelyzeteket. Az elmúlt években a közösségi oldalak igen népszerű médiummá váltak a világ majdnem minden táján, emiatt rendkívül jó információforrásnak bizonyulnak és gyors kommunikációs csatornaként is képesek funkcionálni, különösen természeti katasztrófák esetén. Az egyik ilyen szolgáltatás a Twitter, melynek segítségével a felhasználók rövid szöveges üzeneteket (maximum 140 karakteres tweeteket) oszthatnak meg követőikkel webes és mobil alapú platformok használatával. A Twitter egyik legfontosabb jellemzője a valósidejűsége. A felhasználók gyakran osztanak meg információkat arról, hogy mi jár a fejükben, mit látnak, mit tapasztalnak. A fejlett országokban a legtöbb város területén redundáns a hálózat, elérhető egyszerre vezetékes, WiMax, Wi-Fi, vagy celluláris hálózat, amelyek közül mindegyiken lehet a szolgáltatás segítségével kommunikálni. Ha valahol baj történne, és erről valaki írna egy tweetet (például hogy mit lát, mi történt), az a biztonsági szervezetek számára igen releváns információkat tartalmazhatna. Természetesen nem lenne megoldás, ha innentől kezdve egy csapat éjjel-nappal figyelné az adott területről beérkezett Twitter üzeneteket és keresné a biztonsági szempontból hasznos tweeteket, hanem szükség van egy olyan rendszerre, ami megbirkózik a közösségi média által szolgáltatott óriási adatfolyammal, és ebből különböző adatbányászati technikák segítségével kiválasztja a lényeges üzeneteket. Ezek a technikák a börszt detekció, szöveg klasszifikáció, online klaszterezés, vagy a geotaggelés. Bár ez a megoldás sokkal nagyobb területekre koncentrál, mint egy szimpla tömegrendezvény, a közösségi média által szolgáltatott adatfolyamot kisebb események esetén is nagyon jól lehetne hasznosítani, de felelőtlenség lenne csak ezekre az információkra alapozva döntéseket hozni.

A gyakorlatban talán legjobban alkalmazható rendszer egy 2013-ban publikált cikkben [5] található. A rendszer egy általános keretrendszert - amit egy adott eseményre testre lehet szabni - alkalmaz arra, hogy a résztvevőktől másodpercenként GPS pozíciókat gyűjtsenek, amelyeket később a mobilszervereknek továbbít, ahol azokat ki is értékelik. Az adatok feldolgozása után hőtérképet állítanak elő a különböző tömegjellemzők szemléltetésére:

- sűrűség
- sebesség
- turbulencia
- tömegnyomás [6]

A hőtérképen (1.3. ábra) a meleg színek jelölik az egyes jellemzők magas értékét, a színezés átlátszósága pedig az ottani sűrűséget mutatja. Ezáltal jól felismerhetők a sűrű és nagy sebességű és tömegnyomású területek, amelyek kritikusak lehetnek. A cikkben röviden kitérnek a minták reprezentativitására is. Azt feltételezik, hogy az összes felhasználó

statisztikailag a tömeggel arányosan oszlik el a területen, így helyes következtetések vonhatók le a tömeg viselkedéséről. Persze kívánatos minél több aktív felhasználó jelenléte és a mérések a CCTV felvételekkel pontosíthatók.



1.3. ábra. *A sűrűség és tömegnyomás ábrázolása hőtérképen. [5]*

Megemlíti még, hogy az alkalmazást 2011-ben a Londoni Lord Mayor's Show-n is tesztelték, ahol a teszt előtt csak CCTV-vel monitorozták a tömeget. A rendfenntartók úgy találták, hogy a hőtérképes ábrázolás sokkal könnyebben adott globális képet a tömegről, az általános jellemzők sokkal könnyebben leolvashatók voltak. A résztvevőket is megkérdezték arról, hogy vészhelyzet esetén mennyire hagyatkoznának az alkalmazás által küldött tippekre. A válaszok alapján ez függne a vészhelyzet típusától, attól, hogy lenne-e a helyszínen hivatalos személy, a mobilon érkező információ hivatalos szervektől jön-e, az információ koherens-e a helyszínen tapasztaltakkal. A cikk két fontos következtetést is levon. Első, hogy minél nagyobb felhasználószámra van szükség, hogy pontosabb becsléseket tudjon adni, tehát szükség van arra, hogy ösztönözzék a felhasználókat arra, hogy feltelepítsék az alkalmazást a készülékeikre. Második fontos következtetés, hogy a felhasználókra szondaként kell tekinteni és akkor is lehetséges a pontos tömegjellemzők meghatározása, ha nem tudunk mindenkit követni. A rendszer nagy hibája, hogy a másodpercenkénti GPS lekérdezések nagyon hamar lemerítik az eszközök akkumulátorát, emiatt majdnem biztos, hogy a résztvevők nem használnák hosszútávon.

2. fejezet

Motiváció

Miután kellően elmerültem a témában és megismertem a már létező felügyeleti rendszereket azt láttam, hogy bár léteznek olyan megoldások, amik nagymértékben segítik a szervezők munkáját, mégis szükség van egy olyan új megközelítésre, amely egyesíti a különböző megközelítések előnyeit illetve a résztvevők számára is vonzóbbá teszi a használatot.

A korábban bemutatott felügyeleti rendszerek nem szolgáltatnak pontos mért adatokat arról, hogy egy bizonyos területen mennyien helyezkednek el, hol milyen irányba és hányan mozognak a rendezvény területén vagy ha részben mégis, akkor is nagyon hamar lemerítik a résztvevők készülékeit. A fejlesztendő rendszer nagy előnye lenne, hogy képes lenne jóval pontosabb információkat szolgáltatni a rendezvények házigazdáinak sokkal kisebb hálózati forgalom generálásával, alacsonyabb energia fogyasztás mellett.

Ahhoz, hogy ez megvalósítható legyen szükséges egy részben új tömegfelügyeleti modell megalkotása is, mert a szakirodalomban ismertetett absztrakciók evakuációk vagy menekülési pánik szimulációjára alkalmasak, azonban egy valós környezetben, ahol közel valós időben kell megfigyelni a tömeg eloszlását, mozgását nem igazán használhatóak, de ez érthető is, hiszen nem erre a célra lettek kifejlesztve. A modellnek alkalmasnak kell lennie arra, hogy ábrázolja a tömeg aktuális állapotát, valamint megfelelő absztrakciót biztosítson, melynek segítségével jelentősen lecsökkenthető az irreleváns adatok száma és így a hálózati forgalom, illetve az eszközök energia fogyasztása is.

A cél az, hogy egy olyan rendszert alkossak meg, amely éles környezetben jelentős terhelés alatt is derekasan helyt áll, képes akár tízezres nagyságrendű párhuzamosan beérkező forgalom kezelésére is és szerverfürtökön elosztottan hatékony adatfeldolgozásra. A tervezés és implementáció során fontos szempont volt, hogy új, megbízható és innovatív technológiákat használjak, amelyeket az iparban is előszeretettel alkalmaznak.

3. fejezet

Tömegfelügyeleti modell

3.1. Tömegdinamikai mozgás minták

Első lépésben tanulmányoztam a szakirodalmat, hogy olyan tömegdinamikai modelleket találjak, melyek módosított változata alkalmas lehet a bevezetésben említett probléma megoldására. A megismert modellek mindig valamilyen fizikai modellre épülnek, mivel az emberi tömegek mozgása nagy hasonlóságot mutat a folyadékok vagy szemcsés anyagok áramlásának dinamikájával, így egy ember a fizikai modellekben egy olyan részecskeként modellezhető, ami interakcióba kerül a szomszédjaival. Ha jobban megvizsgáljuk a tömeg mozgását különböző komplex dinamikai mintázatok is azonosíthatóak. A szakirodalom ezekre a tér-időbeli mozgásmintákra önszerveződő jelenségekként hivatkozik [7].

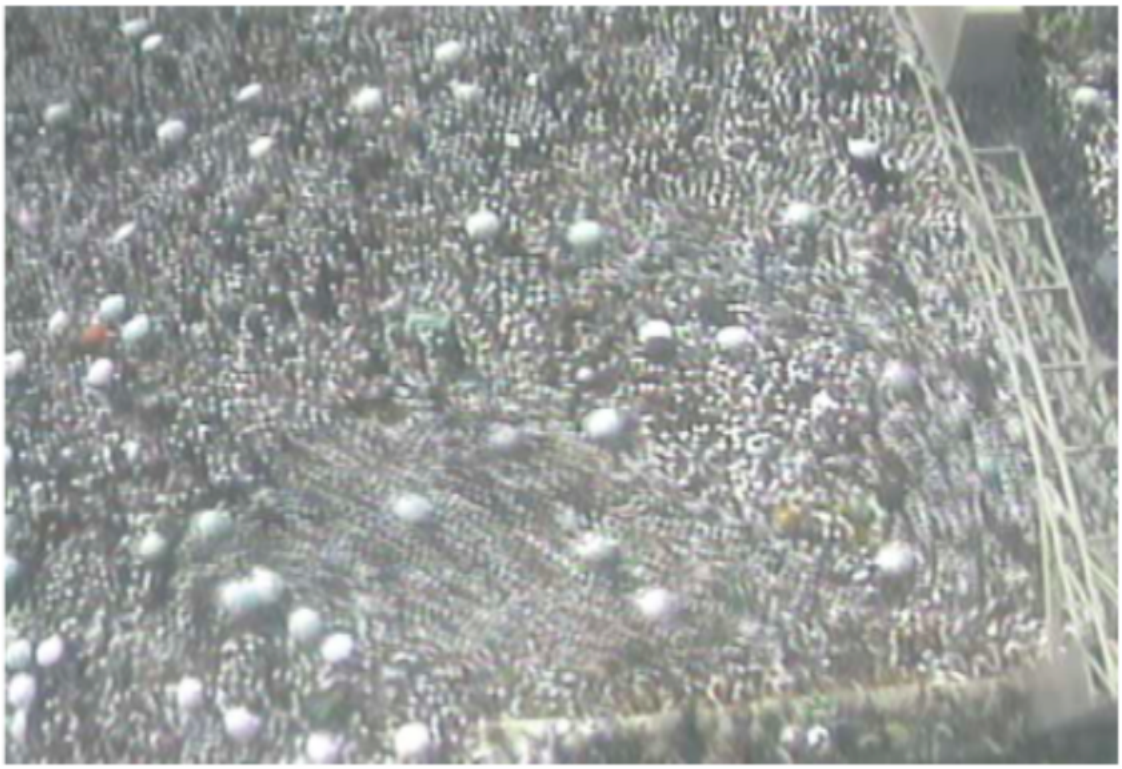
Kis tömegsűrűség esetén az egymással szemben haladó gyalogosok nagyon könnyen, kis sebesség - vagy irányváltással ki tudják egymást kerülni, viszont ahogy növekszik a sűrűség, egyre több mozgási irány jelenik meg, amik igen komplex helyzetet hoznának létre, ha mindenki továbbra is ugyanabba az irányba szeretne haladni. Megfigyelések bizonyítják [8], hogy ilyen esetekben a járókelők sávokba rendeződnek. Ha általánosítjuk a sávokba rendeződés elvét, úgy, hogy csak két sávban, ellentétes irányban mozognak az emberek, láthatjuk, hogy ennek az önszerveződésnek a lényege, hogy minden gyalogos hatékonyabban valósíthassa meg a kívánt mozgását. Azonban ennél jóval komplexebb esetek is előfordulhatnak, például sávok egymáson keresztül haladnak, vagy sávoknak kell összefésülődnie, de új sávok is keletkezhetnek. Ilyenkor cipzár stratégia figyelhető meg, például az autópályákon, ahol a cipzár működéséhez hasonlóan olvadnak egymásba az autófolyamok.

Érdekes viselkedési minta még a nyomvonalak követése is. A jelenséget [9, 10] legjobban füves vagy havas területeken lehet megfigyelni a természetben, ahol az emberek (de egyes állatok is) a már előre kitaposott ösvényeket használják ahelyett, hogy egy újat hoznának létre, mivel azon általában hamarabb, kisebb erőfeszítéssel képesek elérni a céljukat. A hangyák mozgásánál is megfigyelhetjük a jelenséget, bár ők a kemotaxis kihasználásával maradnak ugyanazon az útvonalon. Természetesen, ha egy ösvény nagymértékben eltér a kívánt haladási iránytól, a gyalogos új ösvényt fog létrehozni. Fontos megfigyelés, hogy

természetes környezetben nem fogunk találni olyan nyomvonalat, ami 90 fokos szögben válna ketté (a városi kereszteződésekkel ellentétben), hanem az Y alak a jellemző.

3.1. ábra. *A képen megfigyelhető, hogy míg a homályos részen mozognak az emberek, addig az élesebb részeken nem mozdulnak. [11]*

Ha a sűrűség elér egy kritikus értéket, függetlenül attól, hogy korábban milyen dinamikai mintázatot követtek az abban résztvevők, onnantól fogva nehéz az egyénnek érvényesítenie saját céljait, akaratát, az emberek összetömörülnek, és nem tudnak úgy haladni, ahogy azt ők szeretnék. Ilyenkor alakulnak ki stop-and-go hullámok [6] (3.1. ábra), amelyek esetén váltakozó mozgás figyelhető meg, egyszer mozognak aztán megállnak a járókelők, majd újra elindulnak, és újra megállnak. A gyalogos kénytelen megállni, mivel elfogyott előtte a szabad terület, viszont, ha újra felszabadul előtte egy kis tér, megpróbál odalépni. Ez a jelenség figyelhető meg az autóutakon, például dugók esetén, melynek kiváltó oka lehet, hogy az utak találkozásánál a forgalmak összefésülődése nem megy zökkenőmentesen, de akkor is előfordulhat, hogyha valaki hirtelen fékez, így a mögötte lévő járművet is fékezésre készteti, akinek már nagyobbat kell fékeznie, majd az emögött haladó is fékezni fog, végül a sok fékezés a forgalom lassulását, majd beállását eredményezheti. Miután a tömeg mozgására a stop-and-go hullám válik jellemzővé, még mindig tovább növekedhet a sűrűség, ami egy bizonyos idő után egy másik áramlási formába fog váltani, amit a szakirodalom tömegturbulenciának [6] (3.3. ábra) nevez. Ebben az esetben az emberek csoportjai, blokkjai az összes lehetséges irányba mozognak, teljesen szabálytalanul, és a nagy sűrűség miatt az emberek szó szerint összenyomják egymást és eltaposssák elesett társaikat. Rendkívül veszélyes jelenség, ami gyakorlatilag minden eddigi megfigyelt esetben halálos áldozatokkal járt. A nagy sűrűség ellenére, a mechanikai nyomás nem mindenhol hat ugyanakkora erővel a tömegre!



3.2. ábra. *A képen jól látható az ember csoportok rendezetlen mozgása. [11]*

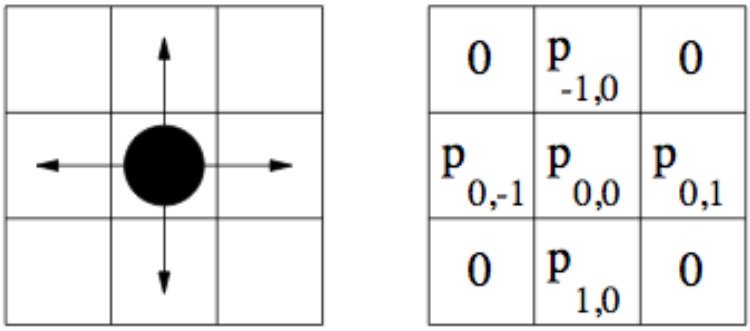
3.2. Tömegdinamikai modellek

Miután megismerkedtem a korábbi komplex mintázatokkal, hozzáláttam a modellek feltérképezéséhez is. A tömegdinamikai modelleket több módon is osztályozhatjuk, ezek közül a számomra két legfontosabb módszert emeltem ki. Egyrészt osztályozhatjuk a modelleket az alapján, hogy azok mikro- vagy makroszkopikusak-e. A mikroszkopikus modellben minden egyén külön kezelendő, tehát bevezethetünk különböző tulajdonságú gyalogosokat, akik más-más irányba haladnak, más sebességgel. Ezzel ellentétben egy makroszkopikus modell esetén nem különböztetünk meg egyéneket, hanem itt a rendszert sűrűségekkel jellemezhetjük. A tömeg sűrűsége az emberek pozíciója alapján számolható, illetve sebességük egy területen kiátlagolható. Egy másik csoportosítási lehetőség, ha a modelleket aszerint vizsgáljuk, hogy azok diszkrétek vagy folytonosak-e. Egy modellt 3 különböző változóval tudunk leírni: tér, idő és állapot (az állapot például lehet a sebesség). Ha ezek az értékek folytonosak, például tetszőleges időpillanatra tudunk állapotot számolni a tér egy adott koordinátájára, akkor értelemszerűen egy folytonos modellről beszélünk (hidrodinamikai modellek), viszont ha ezek az értékek diszkrétek, például 10 másodpercenként vannak állapot adataink adott területekre aggregálva, akkor természetesen diszkrét modellről beszélhetünk.

Még a fejezet elején utaltam arra, hogy a tömegmozgások dinamikája nagymértékben hasonlít a folyadékrészecskék mozgásának dinamikájára. Megfigyelhető például, hogy akadályok esetén a mozgás egyértelműen áramlásszerű. Emiatt az első tömegmozgási modellek a folyadék- és gázmodellekből merítettek inspirációt. Ezek tipikusan makroszkopikus, foly-

tonos erőalapú modellek voltak, amik nem igazán voltak valósághűek.

Henderson [12] megpróbált analógiát találni a klasszikus gázmodellek és a nagy tömegek között. A különböző kis sűrűségű tömegeken végzett mozgásmérések alapján jó megfeleltethetőséget talált a sebességek eloszlása és a Maxwell-Boltzmann eloszlás között [12]. Később, ezen felbuzdulva létrehozott egy folyadékdinamikán alapuló modellt, bár itt vannak olyan megfeleltetések, amelyek nem teljesen tisztázottak. Helbing cikkjében [13] egy fejlettebb folyadékdinamikai alakon nyugvó modellt írt le. Itt a rendszer egy f(r,v,t) sűrűségfüggvényen alapul, ami Boltzmann transzport egyenlétenek segítségével működik. Lényeges, hogy először itt említették meg a kívánt mozgási irány fogalmát, mely lehetővé tette, hogy megkülönböztethetőek legyenek a részecskék μ csoportjai. A hasonló fμ sűrűségek különböző okok miatt változnak meg egy adott időben. A témában két magyar kutató is komoly eredményeket ért el, Farkas Illés és Vicsek Tamás a Fizikai Szemlében megjelent cikkjükben [14] önjáró sokrészecskés rendszerek mintájára modellezték a tömegek mozgását, ami tartalmazza fizikai és szociopszichológiai hatások keverékét is. Sajnos ezek a folytonos modellek nem alkalmasak arra, hogy valós időben több tízezer forrásból érkező adatokat kezeljenek.



3.3. ábra. *Általános celluláris felbontás. A bal oldali képen a lehetséges mozgási irányok, a jobb oldalin pedig az átmeneti valószínűségek vannak feltűntetve.*

A cellás automaták [15] olyan szabályalapú (a személy önmaga dönt arról, hogy mit fog csinálni, tehát az aktuális állapota, a szomszédok mozgása, vagy célpontja befolyásolja döntésében.) dinamikus modellek, ahol minden változó diszkrét (az osztályozásnál említett három változó). Az idő diszkrétsége jelen esetben azt jelenti, hogy egy gyalogos pozíciója mindig bizonyos Δt időnként frissül, ami egy számítógépes szimuláció esetén párhuzamosan valósulhat meg ugyanabban az időpillanatban (például GPGPU használatával). Ez a megközelítés alkalmassá teszi predikciók végzésére. Természetesen a teret is fel kell osztani diszkrét területekre, úgynevezett cellákra, amelyek mérete előre meghatározott a maximális sűrűségek alapján. Ha azt feltételezzük, hogy egy részecske minimum egy cellát foglal el és a maximális sűrűség 6,25 fő/m2 [16], akkor egy cella mérete 40 x 40 cm2 lesz. Fontos megjegyezni, hogy egy cellában egyszerre csak egy részecske tartózkodhat (kizárási elv), de egy részecske több cellát is elfoglalhat, és a részecskék nem összenyomhatók és közöttük taszító erők lépnek fel (privát szféra). Egy rendszer dinamikája a cellák közötti

átmeneti valószínűségekkel írható le, egy gyalogos a 4 fő irányba képes mozogni. Minden cellának minden szomszédjához meg van határozva egy átmeneti valószínűség, ami alapján majd mozognak a részecskék. Egy determinisztikus rendszerben egy cellához tartozó összes átmenet 0, kivéve egyet, ahol 1.

A cellás automatáknak sokféle változata létezik. Fukui-Ishibashi modellje [15] hosszú folyosókon lévő tömegáramlások vizsgálatára lett kifejlesztve, ahol a részecskék egymással ellentétes irányba haladnak (csak kétirányú mozgást vizsgáltak), és pozíciójuk felváltva frissül. Blue-Addler modellje [15] Nagel-Schreckenberg autóforgalmi modelljének egy változata, ahol a gyalogosok ahhoz hasonlóan mozognak, mint egy több sávos autópályán a járművek. Érdekes változata még a cellás automatáknak a Floor field CA, ahol az átmeneti valószínűségek a cellák között már dinamikusan változhatnak. Itt a gyalogosok egy csíkot húznak maguk után, virtuális útvonalat jelölnek ki maguk mögött, amit a többiek lehet, hogy követnek, de lehet, hogy nem (sávokba rendeződés). Bár a cellás automaták diszkrétségük miatt alkalmasabbnak tűnnek tömegfelügyeleti feladatok ellátására, mint egy folytonos modell, a mikroszkopikusságuk miatt sajnos ezeket sem lehet használni egy valós idejű rendszerben.

3.3. Saját tömegdinamikai modell

Természetesen az előzőekben ismertetett tömegdinamikai modelleken kívül még sok másik fellelhető a szakirodalomban, azonban ezeket, a bemutatottakhoz hasonlóan szimulációra használják, például evakuációs tervek kidolgozásánál vagy épületek alkalmasságának vizsgálatnál. Emiatt közvetlenül nem alkalmazhatóak tömegfelügyeleti feladatokra, viszont számtalan hasznos ötletet és megközelítést tartalmaznak, melyeket mindenképpen célszerű felhasználni. Azonban szűkítik a használható modellek körét a bevezetésben megfogalmazott követelmények. Olyan modellre van szükség, amely megfelelő absztrakciót biztosítva lehetővé teszi az összes releváns állapot megjelenítését, viszont kíméli a résztvevők eszközeinek akkumulátorát. Ez úgy érhető el, ha nem periodikusan küldenek adatot a szerver felé, hanem különböző események bekövetkeztekor, például belépett a látogató egy bizonyos részre a rendezvény területén. Ez a megközelítés magában hordozza azt az előnyt is, hogy sokkal alacsonyabb lesz a hálózati forgalom a szerver irányába, ami a csökkent terhelés miatt gyorsabban képes kiértékelni a beérkező adatokat. A korábbiakat megfontolva úgy gondoltam, hogy a legjobb megoldás, ha egy diszkrét makroszkopikus modellt alkotok meg.

3.3.1. Tér reprezentáció

A teret diszkrét területekre (3.4. ábra) bontottam (cellás felbontás [15]). Fontos, hogy itt a cellák nem négyzetek, hanem tetszőleges konvex négyszögek. Ez azért lényeges, mert a cellákat érdemesebb úgy elhelyezni, hogy a valamilyen szempontból összetartozó területeket ne bontsuk szét. Például nem célszerű úgy elhelyezni egy cellát, hogy a területének a fele egy forgalmas úton van, a másik pedig egy épületre lóg rá, ahol nyilván nem lesz senki. Ez pontatlanságot okozna a sűrűségek kiszámításánál, amely egyenesen arányos a nem

kihasznált területtel. A sűrűség számításhoz még szükség van arra, hogy minden cellához megadjuk, hogy mekkora a területe.

3.4. ábra. *Egy példa a cellás felbontásra.*

3.3.2. Gyalogos reprezentáció

Második lépésben a résztvevőket kellett elhelyezni a modellben. A gyalogos (vagy eszköz, hiszen jelen esetben ugyanazzal a mozgás állapottal rendelkeznek) aktuális állapotát két változó segítségével lehet jellemezni: a személy pozíciójával, illetve a sebességének ingadozásával, ugyanis ezeket tudjuk folyamatosan mérni. A pozíció esetén felesleges pontos GPS koordinátákat eltárolni, hiszen a sűrűségeket a kiértékelésénél úgyis cellákra határozzuk meg, emiatt elegendő, hogyha az eszköz csak cella azonosítókat küld el, melyik cellából melyikbe lépett. A sebesség ingadozás ahhoz szükséges, hogy a Lord Mayor's Show-n használt rendszerhez [5] hasonlóan itt is lehessen tömegnyomást számolni, amely egy másik megfelelő metrika a tömeg állapotának vizsgálatára. Egy eszköz csak akkor küld el a szervernek egy mérési üzenetet, ha cellát váltott, illetve ha a készüléken futó alkalmazást, ami az adatokat gyűjti, kikapcsolják, így tényleg akkor történik üzenetváltás, ha az eszköz releváns adatokkal rendelkezik. Természetesen kritikus helyzetekben lehetőség van arra is, hogy egy eszköz ne csak cellaváltásnál küldjön üzenetet.

4. fejezet

A tömegfelügyeleti rendszer architektúrája

A tömegfelügyeleti modellem megalkotása után a következő lépés a tömegfelügyeleti rendszerem architektúrájának megtervezése és implementációja volt. Először utánanéztem annak, hogy milyen új, innovatív technológiák érhetőek el, melyek segítségével gyorsabban és megbízhatóbban tudom elkészíteni a rendszert. A megfelelő technológia megválasztása azért is nagyon fontos volt, mert a követelményekben megfogalmazott kritériumok, a nagyszámú párhuzamos kapcsolat kezelése és az elosztott számítások végzése egyáltalán nem triviális feladat.

Elosztott számítások végrehajtásához napjainkban már több Big Data keretrendszer is elérhető. A korábbi években egyeduralkodó Hadoop [17] mellett megjelent egy új vetélytárs is a Spark [18], amely bizonyos körülmények között akár százszor gyorsabban végez adott feladatokkal, mint a Hadoop. Gyorsaságát annak köszönheti, hogy a műveleteket in memory végzi, tehát a szükséges adatokat a fizikai háttértároló helyett a memóriában tárolja. Viszont nem szabad elfelejteni, hogy a Spark nem rendelkezik saját elosztott fájlrendszer megoldással, emiatt kompatibilis több megoldással is, például a Hadoop HDFS-el is. A Spark egyik legfontosabb funkciója, ami miatt végülis rá esett a választásom, hogy képes TCP folyamon keresztül beérkező adatokat feldolgozni, ami teljesen egybevág a tervezett rendszer működési elvével.

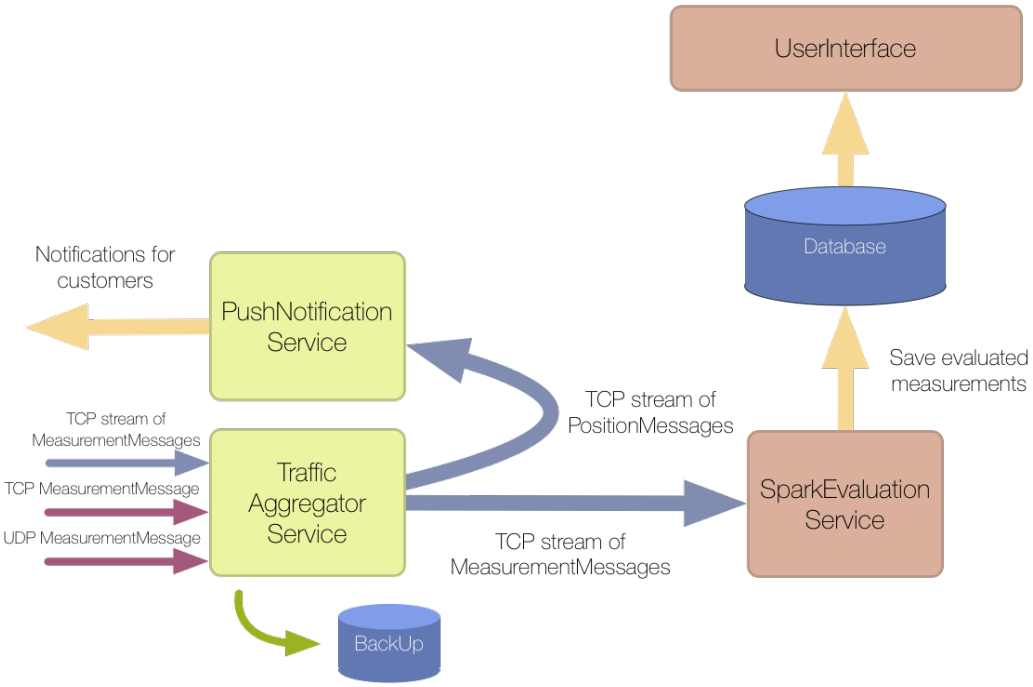
Természetesen önmagában a Spark nem képes megoldani az összes megfogalmazott elvárást a rendszerrel szemben. Bár képes TCP folyam feldolgozására, ezt csak úgy tudja megtenni, hogy kliensként csatlakozva tölti le az adatokat egy szerverről. Nyilváló tehát, hogy egyedül nem képes több tíz vagy akár száz ezer forrásból érkező egyedi mérés fogadására, emiatt szükség van egy olyan komponensre is, ami aggregálja az eszközöktől érkező üzeneteket egy vagy több folyamba. A Spark ezekre a folyamokra csatlakozva már tökéletesen illeszkedik a rendszer architektúrába, viszont a nagyszámú párhuzamos kapcsolat kezelésének problémája még mindig nincs megoldva, hiszen ezzel most akkor a forgalom aggregátornak kell megküzdenie.

Nagyszámú párhuzamos kapcsolat kezelésére nem alkalmas a hagyományos szinkron

megközelítés, ahol minden felépült kapcsolat számára egy külön szálat adunk. Ez a megközelítés nagyon hamar kimerítené a szerver erőforrásait, a sok kontextus váltásról nem is beszélve, megoldás emiatt csak aszinkron módon képzelhető el. Sok különböző keretrendszert tanulmányoztam át, a választásom végül a Netty-re [19] esett, mert amellett, hogy API-ja letisztult és jól használható, gyakran jönnek ki hozzá frissítések és sok nagy cég, mint a Facebook vagy a Twitter is használja a szolgáltatásaihoz.

Ahhoz, hogy szervezők a résztvevőknek üzeneteket küldhessenek, szükség volt arra, hogy valamilyen üzenetküldő szolgáltatást is beépítsek a rendszerbe, mely pozíció alapján küld a látogatóknak értesítéseket. Ehhez felhasználhattam volna az Apple Push Notification szolgáltatását [21] vagy a Google Cloud Messaging szolgáltatását [22], viszont nem szerettem volna, ha egy külső szolgáltatástól függővé válna a rendszer, ezért itt az előzőekhez hasonló saját szolgáltatást készítettem.

Természetesen a kiértékelt adatokat valamilyen módon meg is kell jeleníteni a szervezők számára. Úgy döntöttem, hogy ezt egy webes felületen fogom megvalósítani, mert amellett, hogy ugyanazt a felületet asztali számítógépen, laptopon, tableten és telefonon is el lehet érni, nagyon sok ingyenesen elérhető plugint lehet használni ahhoz, hogy minél ergonomikusabb és átláthatóbb felületeket alakítsak ki. Így a szervezők gyorsan átláthatnak egy kialakult kritikus helyzetet és azonnal tudnak reagálni.



4.1. ábra. *Rendszerarchitektúra.*

A rendszeren belül tehát ezek a különálló komponensek működnek együtt. Az egységek között a feladatok úgy lettek szétosztva, hogy egy komponens meghibásodása ne okozza a rendszer teljes leállását.

A résztvevők készülékeinek mérési üzenetei a forgalom aggregációs modulba (4.1. ábra, TrafficAggregatorService) érkeznek meg, ahol ezeket különböző folyamokba aggregálom. Ezeken a folyamokon keresztül továbbítom őket az üzenet küldő, illetve a kiértékelő szol-

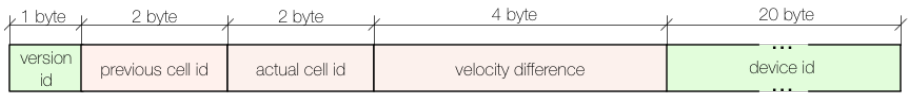
gáltatások felé. A Push modul (4.1. ábra, PushNotificationService) a rendezvény látogatóit informálja a pozíciójuk alapján a szervezők által definiált üzenetekkel, míg a Spark kiértékelő szolgáltatás (4.1. ábra, SparkEvaluationService) elemzi az összes beérkezett üzenetet és az eredményeket az adatbázisba tölti. Az adatbázis (4.1. ábra, Database) tartalmát egy letisztult felhasználó felületen (4.1. ábra, UserInterface)keresztül érhetik el a szervezők. Magát az applikációt nem mi tervezzük, hiszen egy rendezvény manapság már rendelkezik sajáttal, mi csak egy plugint biztosítunk, amit csak egyszerűen hozzá kell csatolni a már létező alkalmazáshoz. A plugin egy API-n keresztül hívható és amellett, hogy ki-be kapcsolható fel lehet iratkozni a szervertől érkező üzenetekre, amiket aztán már megjeleníthetnek a felhasználóknak.

4.1. Forgalom Aggregációs Szolgáltatás

A rendszer egyik központi eleme a TrafficAggregatorService (továbbiakban TAS). Feladata, hogy felületet biztosítson a mobil eszközök számára, ahová beküldhetik méréseiket (MeasurementMessage), majd a beérkezett adatokat egy TCP folyamban továbbítja a kiértékelő komponens felé (SparkEvaluationService) és a PushNotificationService felé ha az be van kapcsolva. Egy időpillanatban több kiértékelő szerver is csatlakozhat, így lehetővé téve, hogy akár egy egész szerver klaszter képes legyen résztvenni az adatok feldolgozásában.

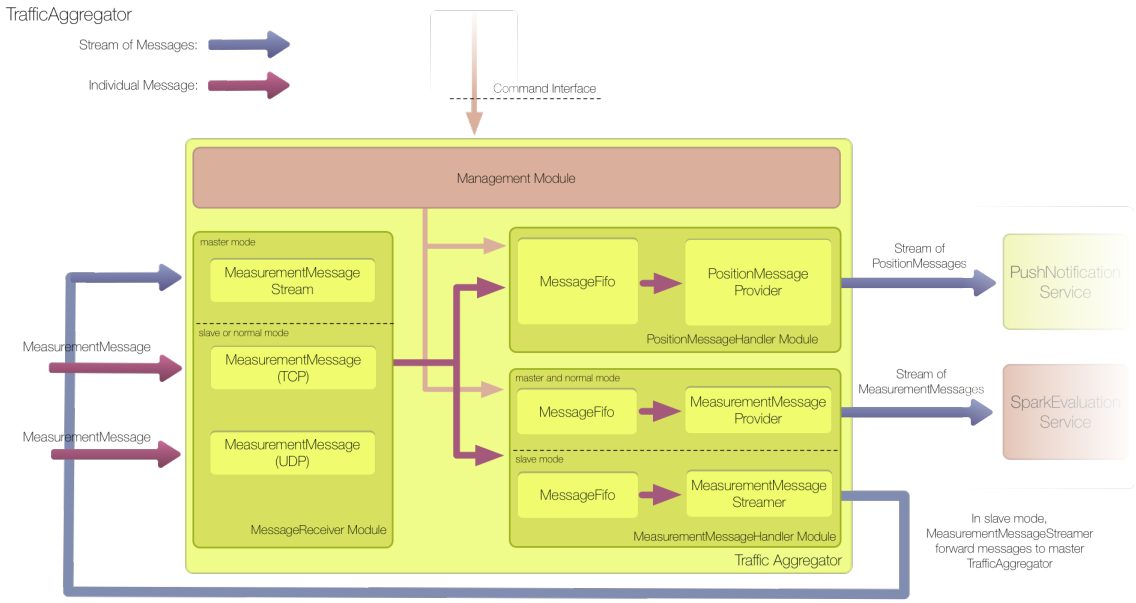
A saját modellemet figyelembe véve készítettem el egy mérési üzenet struktúráját, melynek végleges mérete összesen 29 bájt lett, ami az alábbi értékeket tartalmazza:

- térkép verzió száma (1 bájt)
- előző cella azonosítója (2 bájt)
- következő cella azonosítója (2 bájt)
- sebesség ingadozás (4 bájt)
- eszköz azonosítója (20 bájt)



4.2. ábra. *Mérési üzenet felépítése.*

A térkép verzió száma azért szükséges, mert előfordulhat olyan eset, amikor az eszköz által ismert felbontás nem egyezik meg a szerveren lévővel (módosítani kellett valamilyen okból kifolyólag a cellák elrendezését), ilyenkor az eszköznek frissítenie kell a saját térképet (ennek módját egy későbbi fejezetben ismertetem). Terveim szerint a változtatás maximum naponta szükséges, emiatt egy rendezvény a 256 különböző lehetőséget nem valószínű, hogy kimerítené. A cella azonosítók összesen 65535 különböző értéket vehetnek fel (a 0-s cella kivételt képez), ami biztosan elegendő lesz, hogy lefedjék vele egy rendezvény területét. A sebesség ingadozás egy lebegőpontos szám érték, az eszköz azonosítója pedig a SHA-1 egyirányú függvény kimenete, amely bemenetére az eszköz MAC címét kapja.



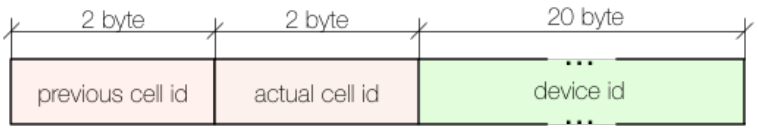
4.3. ábra. *A forgalom aggregátor belső architektúrája.*

Hogy elbírja a nagy terhelést a mobil eszközöknek biztosított interfészén, a Netty aszinkron esemény vezérelt keretrendszerét [19] használom, melynek segítségével akár több tízezer párhuzamos kapcsolatot is lehet szimultán kezelni. Bár a Netty nagy segítség, nem oldja meg önmagában a problémát. Implementálnom kellett hozzá még egy MeasurementMessage kodeket, melynek segítségével a beérkező bájt tömböket objektummá lehet alakítani, illetve meg kellett írnom az üzenet kezelő logikát is, ami fogadja a beérkezett méréseket. Fontos megjegyezni, hogy az eszközök a minél hatékonyabb erőforrás kihasználás céljából nincsenek folytonos kapcsolatban a TAS-sal, hanem csak akkor küldenek adatot (mindössze 29 bájtban), hogyha az szükséges a cella váltáskor. Mivel kicsik az elküldött adatcsomagok, az esetek túlnyomó többségében egy TCP kapcsolat felépítése 3 utas kézfogással több idő lenne, mint magát az adatokat elküldeni, így ez csak felesleges kommunikációt jelentene. Emiatt a TAS egyszerre biztosít TCP és UDP portokat is a kommunikációhoz. A mobil eszközökön lévő alkalmazások általános esetben az üzeneteket az UDP portra, vészhelyzetek esetén a TCP portra küldik, így biztosítva, hogy a fontos üzenetek biztosan ne vesszenek el.

A beérkezett üzenetek egy FIFO-ba kerülnek, ahonnan később továbbítódnak egy kiértékelő szerver felé. A TAS és a kiértékelő szerver közötti kapcsolat megvalósításához nem a Netty-t, hanem egy saját Java szerver megoldást használok. A FIFO-t magam implementáltam úgy, hogy szálbiztos legyen és minél kevesebb időt vegyen igénybe az üzenetek ki és behelyezése. A szálbiztonság azért volt nagyon fontos, mert a Netty egyszerre több szálon keresztül használja, míg a másik oldalról is párhuzamosan több szál szolgálja ki a kiértékelő szervereket, hogy minél gyorsabb legyen a feldolgozás.

Tesztelés közben feltűnt, hogy az ArrayList gyári implementációjánál nagyon időigényesek az add és remove műveletek (kb. 8×10^{-5} másodperc), úgy döntöttem, hogy egy gyorsabb megoldást fogok használni, a Javolution [20] FastTable megvalósítását, ami az előző műveletekkel kevesebb mint 1×10^{-5} másodperc alatt végez.

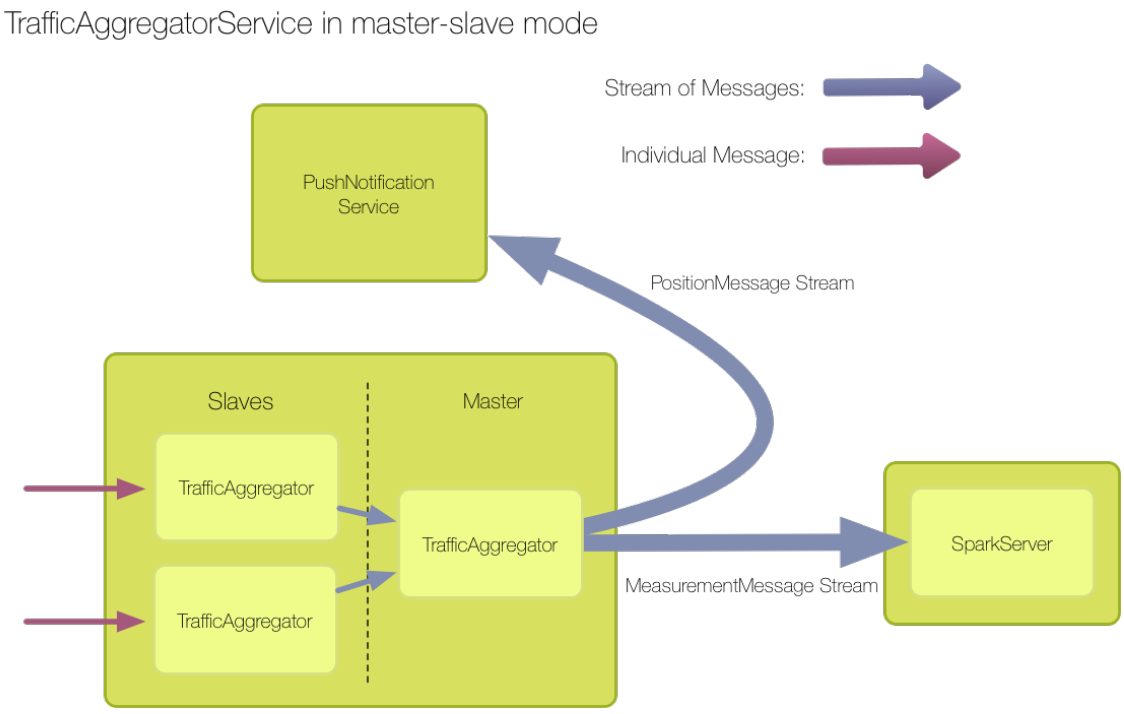
Ha a PushNotificationService szolgáltatás be van kapcsolva, akkor minden üzenetből egy úgynevezett PositionMessage (4.4. ábra) is készül, ami a PNS-t kiszolgáló FIFO-ba kerül. Egy Position Message nagyon nagy hasonló egy MeasurementMessage-hez, viszont nem tartalmazza a térkép verzió számát, illetve a mért sebesség különbséget, mérete így 24 bájtos.



4.4. ábra. *Egy PositionMessage üzenet struktúrája.*

Abban az esetben, ha nem lenne elegendő egy darab TAS a beérkező kérések kiszolgálásához, megoldottam, hogy egyszerre több TAS példányt egymáshoz láncoljunk master-slave architektúrában (4.5. ábra), így növelve még jobban a teherbírást. Hogy ne kelljen új komponenseket elhelyeznem a TAS-on belül felvérteztem a Nettyt üzenet folyamok kezelésének képességével is. Ilyenkor a slave példányok tartják a közvetlen kapcsolatot a résztvevők készülékeivel, majd a beérkezett üzeneteket már folyamként továbbítják a mester példány felé, ahonnan a SparkEvaluationService, illetve a PushNotificationService letölti az üzeneteket.

Hogy a TAS-t dinamikusan lehessen konfigurálni, készítettem egy konfigurációs fájlt, ami tartalmaz minden lehetőséget egy példány részletes beállításához például hogy melyik portokon figyeljen a Netty vagy hogy normál, master vagy slave üzemmódban működjön.



4.5. ábra. *Forgalom aggregátor példányok master-slave architektúrába rendezve.*

4.2. Spark Kiértékelési Szolgáltatás

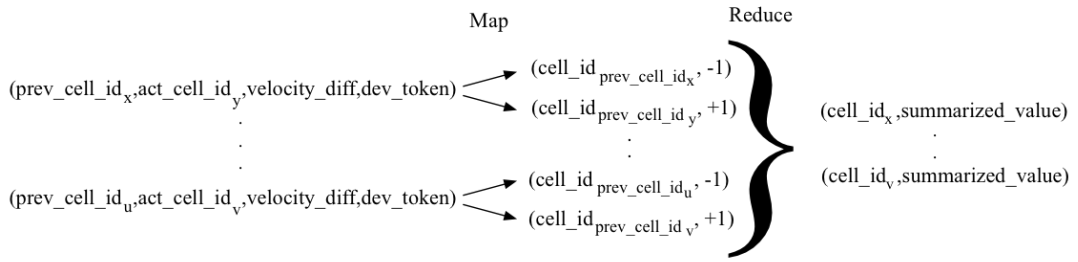
A SparkEvaluationService komponens felel a beérkező mérési adatok hatékony kiértékeléséért. Ehhez a Spark általános célú adatfeldolgozó motort [18] használja, amely képes szerver klaszteren elosztottan nagymennyiségű adat feldolgozására. Ennek köszönhetően nekem azzal kellett foglalkoznom, hogy implementáljam a megfelelő kiértékelő algoritmust, illetve felépítsem az adatbázis stuktúrát és összekapcsoljam azt a rendszerrel. A Spark nagy előnye, hogy képes TCP folyamot is adatforrásként kezelni (SparkStreaming), viszont az itt implementált algoritmus, amely a kiértékelésért felel, előre beállított időközönként fut le, tehát egyszerre csak egy adott időablakban beérkezett üzeneteket összegzi. Mivel az eszközök nem periodikusan küldenek adatokat magukról, hanem csak akkor amikor az fontos, egy időablakból származó eredmény csak a változást adja meg az előző időablakhoz képest. Az aktuális állapotot a következő képlet eredményeként kaphatjuk meg:

$$S_{act} = C_{act} + S_{act-1} \tag{4.1}$$

$$S_{act} = \sum_{i=1}^{act} C_i + S_{init} \tag{4.2}$$

, ahol C_i az i. időablakból származó eredmény (állapotváltozás) S_{init} pedig a kiindulási állapot. S_{init}-et a komponens indulásánál határozom meg az adatbázisban szereplő értékek alapján. Erre azért van szükség, mert egy újraindulás esetén is onnan tudja folytatni a kiértékelést, ahol azt korábban abbahagyta.

A C_i állapotváltozás a következő módon számolható ki. Legyen az adott időablakban i.-ként beérkezett üzenet $mess_i$, mely a $(prev\_cell\_id, act\_cell\_id, velocity\_diff, dev\_token)$ négyesből áll, ahol $prev\_cell\_id$ az előző cella azonosítója, act_cell_id az aktuális cella azonosítója, $velocity\_diff$ az eszköz által mért sebesség ingadozás az előző mérési pontjához képest, dev_token pedig az eszköz egyedi 20 bájtos azonosítója. Először a létszám változást kell kiszámolni, melyet egyszerű megkapni, ha a $prev\_cell\_id$-t -1-ként az act_cell_id-t pedig $+1$-ként képezzük le egy cella azonosítóhoz, majd a -1 és $+1$-et összegezzük (4.6. ábra).



4.6. ábra. *A kiértékelésnél használt Map Reduce ábrázolása.*

Ezután az act_cell_id alapján csoportosítom a sebesség ingadozás értékeket, majd csoportosítás után cella azonosítók alapján kiátlagolom őket, az így kapott eredményt hozzácsatolom a létszám változásnál kiszámolt eredményekhez.

Az előzőeken kívül meg kellett határoznom még, hogy milyen információk szükségesek mindenképpen egy S_i állapotban cellákra vonatkoztatva a szervezők számára, majd ki kellett találnom, hogy ezeket, hogyan tudom kinyerni az állapotváltozásokból. A szakirodalmat és a korábbi rendszereket tanulmányozva az alábbiak mellett döntöttem:

- létszám
- átlagos sebesség variancia
- sűrűség
- tömegnyomás

A létszám számítása nem túl bonyolult a jelen környezetben. Ha ismerem S_{act-1} állapot létszámait adott cellákra, akkor ezekhez csak hozzá kell adni a C_act állapotváltozás létszámváltozásait, amelyből meg is kapom az aktuális létszámokat. Az átlagos sebesség ingazodásnál nem vehetőek figyelembe az S_{act-1} állapotban tárolt értékek, így az S_{act} sebesség variancia értékek meg fognak egyezni a C_{act}-ban kiszámoltakkal. A sűrűség számításához szükség van az aktuális létszámra, illetve a cella területekre, viszont ezeket az értékeket már ismerem, így könnyen kiszámolható. A tömegnyomás [6] eredeti képletén nagy mértékben tudtam egyszerűsíteni a cellás felbontásnak köszönhetően, melyet végül a 4.4-s képlet alapján számolok.

$$Cell_u.density = \frac{S_{act}.Cell_u.headcount}{Cell_u.area} \tag{4.3}$$

$$Cell_u.crowd\_pressure = S_{act}.Cell_u.density * S_{act}.velocity\_difference \tag{4.4}$$

Az előző értékek kiszámítását természetesen a SparkEvaluationService végzi. Indulásnál, ahogy azt már korábban említettem felépíti az adatbázis alapján az aktuális állapotot, amit a későbbiekben mindig a memóriában tart. Ha beérkezik egy új állapotváltozás módosítja a belső állapotát a kiszámolt értékekkel és ezt elmenti az adatbázisba is.

4.3. Push Szolgáltatás

A PushNotificationService komponens feladata, hogy egy egyirányú kommunikációs csatornát biztosítson a rendfenntartók és a résztvevők között push jelleggel a pozíció információk alapján. Ahhoz, hogy ezt meglehessen valósítani, szükség van egy állandó TCP kapcsolat fenntartására a kommunikációs felek között, cserébe a szervezők által beállított értesítés azonnal megjelenik egy adott cellában elhelyezkedő összes látogató készülékén.

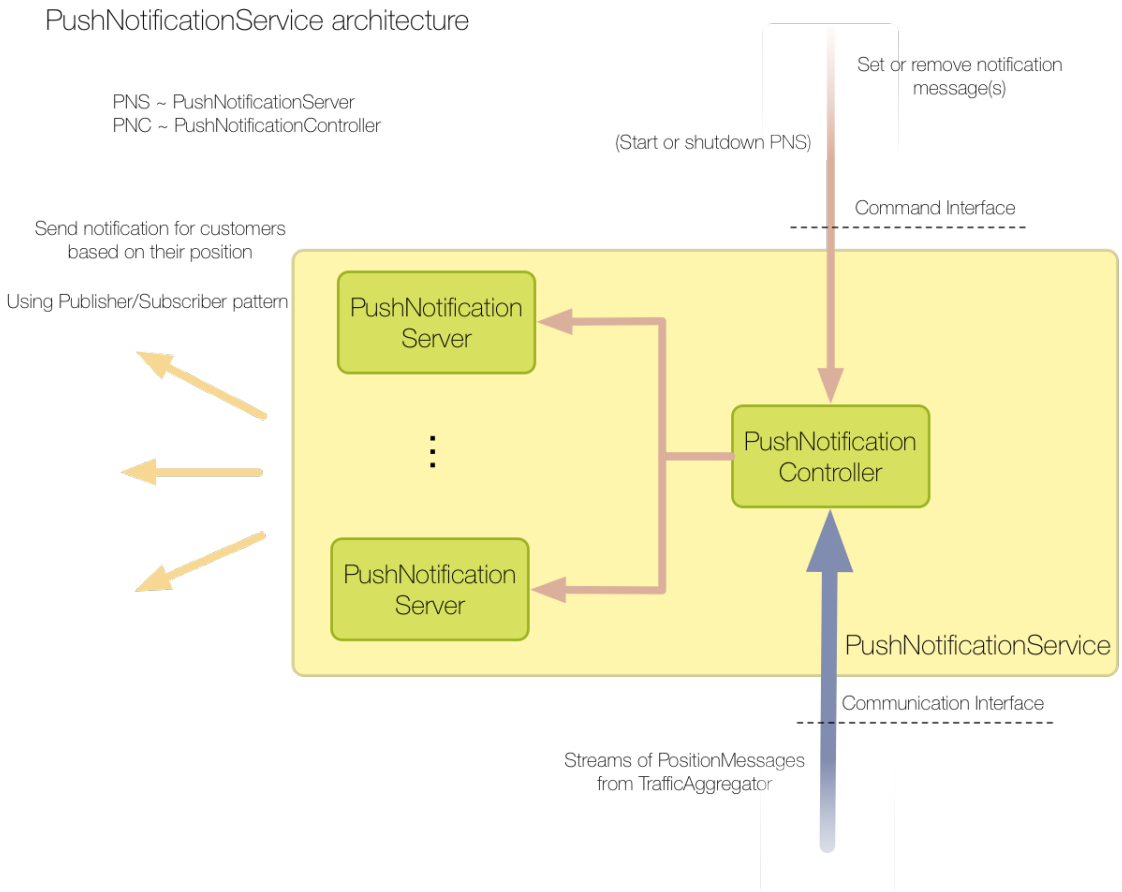
A komponens két különböző típusú elemből épül fel (4.7. ábra), tartalmaz egy Push Notification Controller-t (PNC) és egy vagy több Push Notificiation Server-t (PS). A PNC feladata a nevéből is adódóan egyrészt az, hogy vezérelje a PS példányok működését, amelyek közvetlen kapcsolatban állnak a résztvevők készülékeivel, másrészt felelős a látoga-

tóknak szánt üzenetek menedzseléséért is. A PNC megvalósításában egyszerre több szállal töltöm le az üzeneteket a TAS-tól (a szálak mennyisége paraméterezhető). A letöltött üzenetek egy szálbiztos FIFO-ba kerülnek, ahonnan aztán további szálak (ezek a mennyisége is paraméterezhető) osztják szét a PositionMessage-ket a PushServer-ek között. Ehhez a PNC-ben nyilván kell tartanom az aktuálisan kapcsolódó PS-ket, amik egyszerre több TCP kapcsolaton keresztül is kommunikálnak a vezérlővel, hogy még hatékonyabb és gyorsabb legyen a működés. A PNC-ben továbbá biztosítanom kellett egy felület a szervezőktől érkező kéréseknek is. Ehhez implementáltam egy szervert, ami a beérkezett parancsokat lekezeli illetve ha szükséges továbbítja azokat a PS-ek felé.

A PS-eket úgy valósítottem meg, hogy egyenként egyszerre több tízezer résztvevővel is képesek legyenek kapcsolatot fenntartani, és tárolják, hogy egy bizonyos cellához kapcsolódóan mik az aktuálisan érvényes üzenetek.

Felmerül a kérdés, hogy erre a struktúrára miért volt szükség? Mivel a Push Notification Server-ek még nagyobb terhelésnek vannak kitéve, mint a TrafficAggregatorService, hiszen nekik állandó kapcsolatot kell fenntartaniuk az eszközökkel, fel kell készülni arra, hogy biztosan több példányt kell futtatni egy éles környezetben, hogy ki tudjuk szolgálni az igényeket egy nagyméretű rendezvény esetén. Hogy a PS-ek munkája összehangolt legyen, szükség volt még egy vezérlő csomópontra.

Amikor egy eszközön elindul a megfigyelő alkalmazás az első lépésben mindig a PNS-



4.7. ábra. *A PushNotificationService belső architektúrája.*

hez próbál csatlakozni. Ha ezt sikeresen megtette, elküldi a szervernek azt a 20 bájtos azonosítóját, amit később a mérési üzenetekbe is el fog helyezni, hogy ez alapján legyen azonosítható. Ilyenkor a PushServer beregisztrálja magához az eszközt, értesíti a vezérlő csomópontot, hogy mostantól felé kell továbbítani a készüléknek érkező üzeneteket és válaszul visszaküldi az eszköznek a jelenleg érvényesen használható térképet (cellás felbontást GPS koordinátákkal), illetve még egyéb inicializáló üzeneteket, ha arra szükség van.

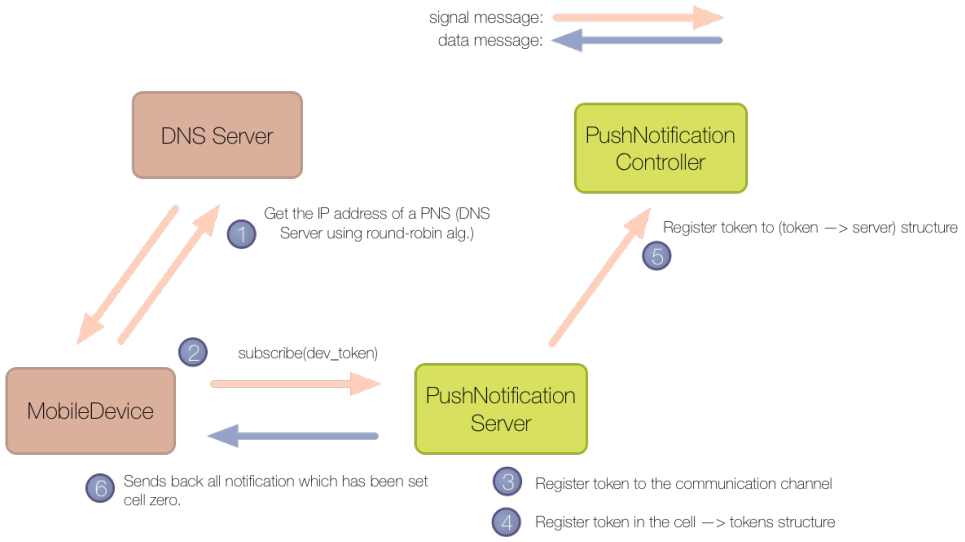
Későbbiekben amikor egy eszköz adatokat közöl a rendszerrel, azok először a Traffic Aggregator Service-be érkeznek meg, ahol minden egyes üzenetről készül egy PositionMessage duplikátum is, melyek továbbítódnak a Push Notification Controller-be (4.9. ábra). A vezérlő, mivel az eszköz regisztrációnál már megtanulta, hogy a készülék melyik szerverhez csatlakozik, a megfelelő PushNotificationServer irányába küldi tovább a PositionMessage üzenetet. Az adott PS szerver ezután beregisztálja a telefont az adott cellához és elküldi neki az összes olyan üzenetet (NotificationMessage), ami arra definiálva lett.

A szervezők természetesen a rendszer futása közben is meghatározhatnak vagy törölhetnek NotificationMessage-ket a cellákra (4.11. ábra). Ilyenkor az összes eszköz ami ezekben a cellákban helyezkedik el, megkapja azonnal az új üzeneteket.

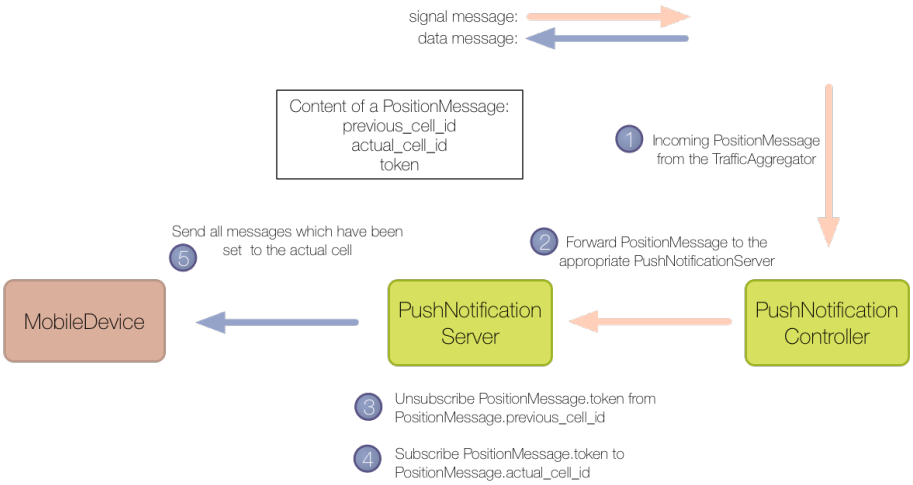
4.4. Adatbázis

4.4.1. Kiértékelések eredményeinek tárolása

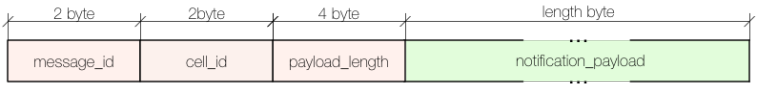
Az adatbázis szerepe meghatározó a rendszerben, hiszen itt végzem központilag az eredmények tárolását. Így egyrészt felületet biztosít a kiértékelő szervereknek, hogy indulásuknál a korábbi állapotok lekérdezésével felépíthessék az aktuálisat, valamint lehetőséget biztosít a későbbi periodikus kiértékeléseknél az eredmények mentésére. Ezzel párhuzamosan kiszolgálja adatokkal a felhasználói felület, emiatt ha új állapotot adnak hozzá a rendszerhez, arról azonnal (maximum 2-3 másodperces késleltetéssel) értesülnek a szervezők.



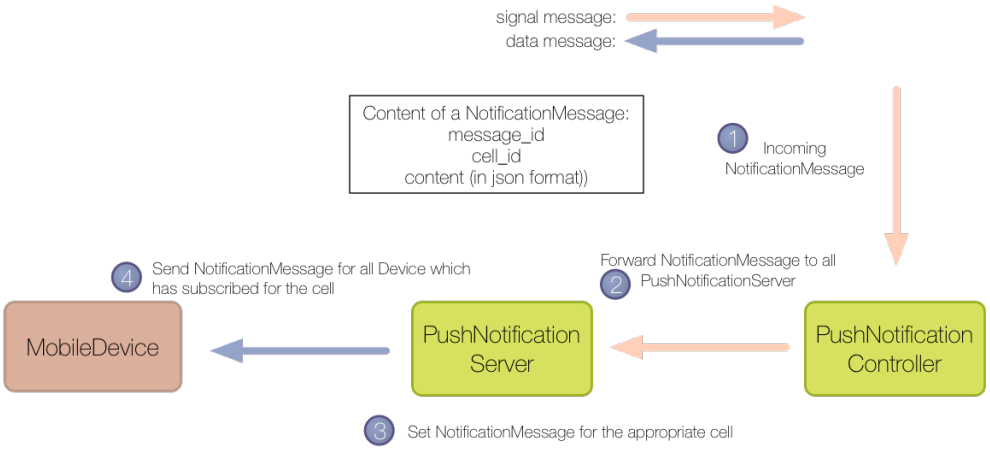
4.8. ábra. *Eszköz csatlakozásának lépései a PNS-hez.*



4.9. ábra. *Beérkezett PositionMessage kezelése.*



4.10. ábra. *Egy NotificationMessage üzenet struktúrája.*



4.11. ábra. *Új NotificationMessage beállításának lépései.*

A kiértékelt állapotokat két tábla segítségével tartom nyilván. Mivel ezek a táblák nagy igénybevételnek vannak kitéve, InnoDB motort [23] használok hozzájuk. Az első táblában (ms_state_infos) az adott állapotokról tárolok általános adatokat, melyek sora a későbbiekben biztosan bővülni fog. A második táblában (ms_states) az egyes állapotoknál cellákra lebontva tárolom a kiértékelésnél kiszámolt értékeket, tehát egy új állapot beérkezése itt annyi új rekordot fog eredményezni ahány cella található az aktuális cellás felbontásban.

ms_state_infos:

- *state_id*: egyedi állapot azonosító, int(11)
- *valid_from*: állapot érvényességének kezdete, datetime

ms_states:

- *state_sequence_id*: egyediséget biztosító azonosító, int(11)
- *state_id*: állapot azonosító, int(11)
- *cell_id*: cella azonosító, int(11)
- *headcount*: létszám, int(11)
- *velocity_variance:* sebesség ingadozás, float(11,5)
- *density*: sűrűség, float(11,5)
- *crowd_pressure*: tömegnyomás, float(11,5)

4.4.2. Értesítési üzenetek tárolása

Szükséges még nyilvántartani az összes eddigi regisztrált NotificationMessage-t is. Ez azért fontos, hogy a felhasználói felületen lehessen látni, hogy milyen aktuális értesítési üzenetek vannak a rendszerben, illetve ha véletlen meghibásodás miatt leállna a PushNotificationService, akkor újraindulásnál újra betölthesse az összes érvényes üzenetet. A lejárt üzeneteket természetesen nem törlöm a rendszerből, hanem naplózom őket így a későbbi vizsgálatoknál akár mérhető az is, hogy az egyes üzeneteknek milyen hatása volt a résztvevők mozgására.

Mivel az üzeneteket itt inkább csak backup célokra tárolom (a PNS-ben történik a NotificationMessage-k valós szétosztása) MyISAM motort [24] használok az ms_notification _messages táblához, amelynek struktúrája a következő:

- *message_id*: egyedi üzenet azonosító, int(11)
- *valid_to*: üzenet érvényességének lejárta, datetime
- *content*: üzenet tartalma, varchar(5000)
- *cell_id*: melyik cellához tartozik, int(11)
- *title:* üzenet címe, varchar(100)

4.4.3. Térkép tárolása

Ezeken kívül fontos még adatbázisban tárolni az aktuális térképet. Nagy valószínűséggel egy rendezvény térképe (cellás felbontása) nem változik gyakrabban mint naponta egyszer (például az előző napi eredmények alapján másnap reggelre módosítják), így a szervezőknek jelenleg nincs közvetlen felülete a szerkesztéshez. A térképet vagy annak részeit minden rendszerkomponens használja, a résztvevők eszközeiről nem is beszélve.

A térképet és a hozzá tartozó adatokat két táblában tárolom. Mivel ezeket a táblákat ritkán szerkesztik, ezekhez is MyISAM motort alkalmazok. Az első táblában tárolom verziókként megkülönböztetve, hogy egy adott cellához milyen adatok tartoznak (például terület), míg a másodikban a cella határok GPS koordinátáit.

ms_map:

- *map_sequence_id*: egyedi bejegyzés azonosító, int(11)
- *version_id*: verzió azonosító, int(11)
- *cell_id*: cella azonosító, int(11)
- *area*: terület értéke, float(11,5)
- *coordinate_id* hozzá tartozó koordinára azonosító, int(11)

ms_map_coordinates:

- *coordinate_seq_id*: egyedi bejegyzés azonosító, int(11)
- *coordinate_id*: koordináta azonosító, int(11)
- *coordinate_latitude*: szélességi fok, int(11)
- *coordinate_longitude*: hosszúsági fok, float(11,5)

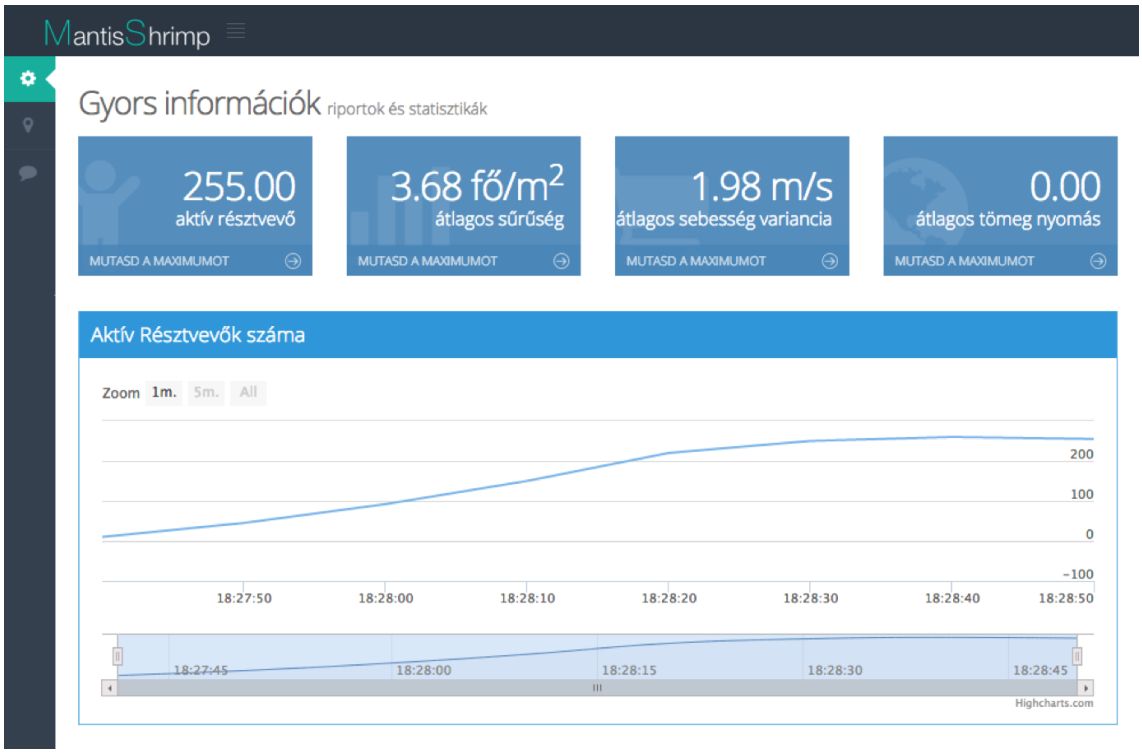
4.5. Felhasználói felület

A felhasználói felület kialakításánál egyrészt cél volt az, hogy a lehető legegyszerűbb formában, logikusan jelenítsem meg a kinyert információkat, szem előtt tartva, hogy a szervezők, rendfenntartók minél gyorsabban átlássák a kialakult helyzetet, legyen jó rálátásuk az aktuális állapotra. Másrészt fontos volt az is, hogy a felületet könnyen el lehessen érni és sok eszközzel legyen kompatibilis, emiatt a választásom a webes megoldásra esett. Manapság számos olyan ingyenes plugin érhető el, melyek segítségével relatíve gyorsan fejleszthetünk ergonomikus felületeket. Az oldal lekérdezéseket egy Apache HTTP Server szolgálja ki egy PHP-ban írt webes alkalmazással együtt, a reszponzív dizájn elkészítéséhez pedig a Twitter Bootstrap-et használom.

Hogy a felület elemek mindig az aktuális állapotot tükrözzék, a böngésző periodikusan lekérdezéseket küld a webszerver felé. Ezekben a lekérdezésekben csak státusz azonosítók közlekednek, amiket a szerver összehasonlít az adatbázisból elérhető aktuális állapottal. Ha nincs egyezés a kettő között, szól a böngészőnek, hogy friss információkat lehet letölteni, megjeleníteni.

Ha belép egy szervező az oldalra az Általános információk felület (4.12. ábra) fogadja kezdő képernyőként. A felületen gyors információkat kaphat a tömeg aktuális állapotáról, illetve egy diagrammon követheti, hogy hogyan változtott az aktív résztvevők száma az idő függvényében.

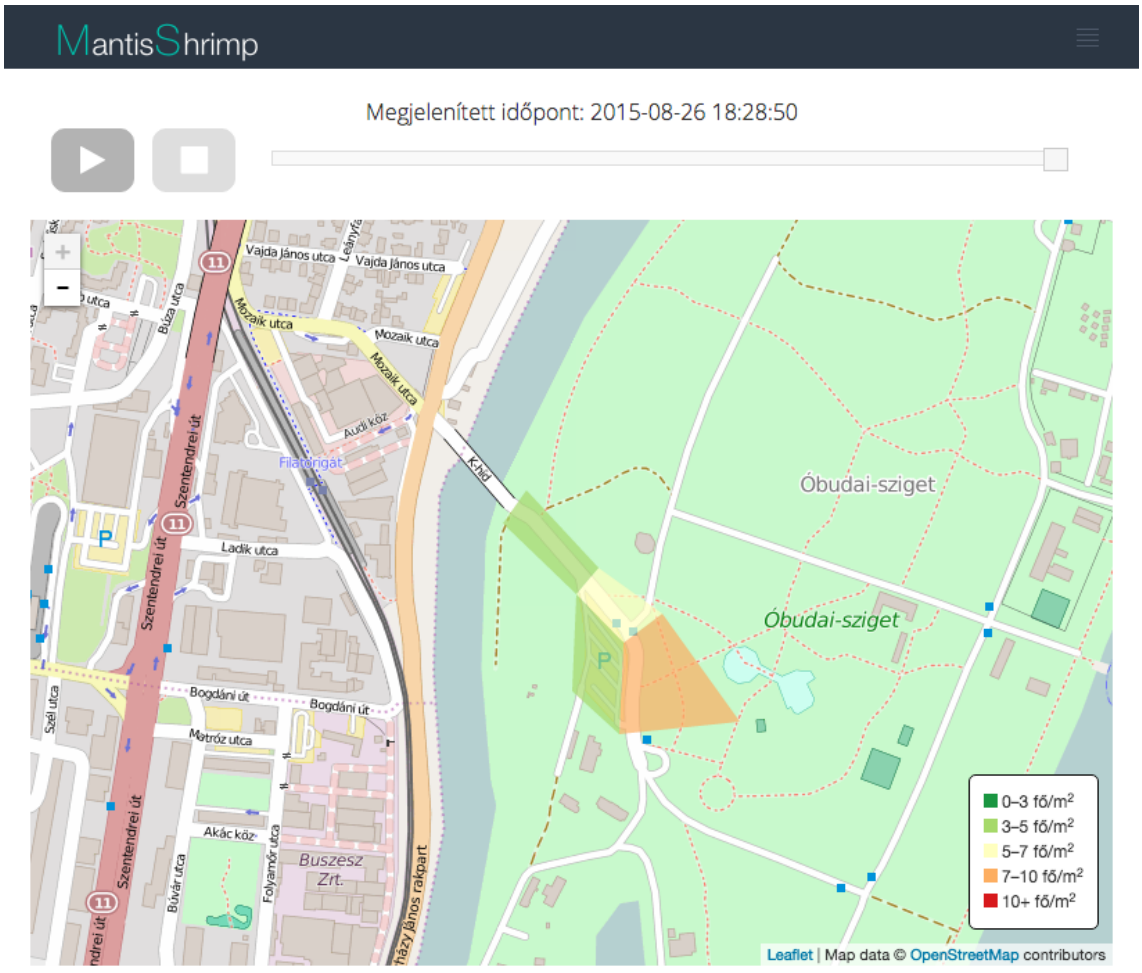
A következő oldalon, a Térképes felületen (4.13. ábra) a szervező láthatja az aktuális sűrűségi és tömegnyomási adatokat. A lejátszás gombra kattintva képes megtekinteni, hogy az időmúlásával hogyan változtak az állapotok, de bármikor megállíthatja, kereshet benne, ha úgy kívánja. Ha a térképen bármikor rákattint egy cellára, akkor megjelenik egy felugró ablak, ahol a kiválasztott időponttól függően számos hasznos információt jelenít meg. Diagrammokon nyomon lehet követni, hogy az idő múlásával hogyan alakult a cellában a résztvevők száma, a tömeg sűrűsége és nyomása. Hozzáférhetőek a cellára aktuálisan



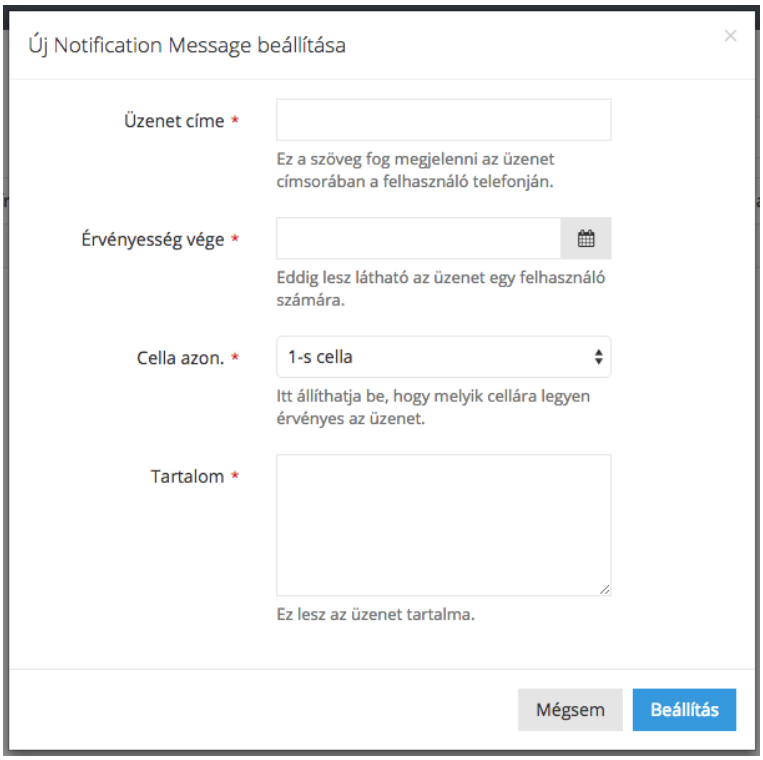
4.12. ábra. *Általános információs felület.*

érvényes, a szervezők által definiált üzenetek, amiket törölhetünk vagy ezekhez újakat adhatunk hozzá. Továbbá megjeleníthetőek még itt, kamerarendszerek vagy akár drónok élő videó, ami persze külön integrációt igényel.

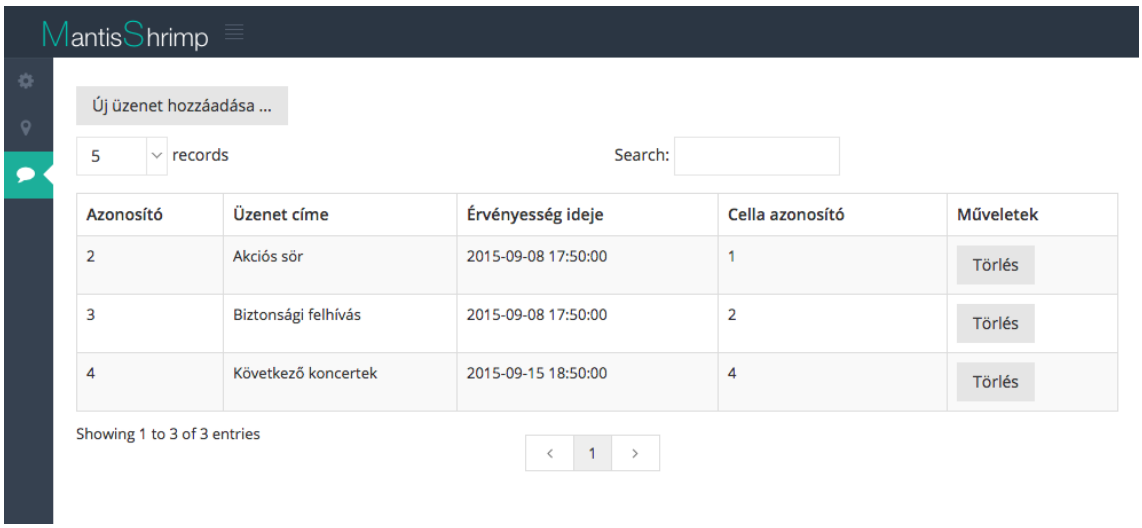
Ezen a felületen a szervezőknek lehetősége nyílik arra, hogy adott cellákhoz különböző üzeneteket (NotificationMessage-ket) állíthassanak be. Az új üzenet hozzáadása gombra kattintva a 4.14. ábra felugróablaka válik láthatóvá. Itt az üzenethez be kell állítani címet, érvényességi időt, cellát és tartalmat, ezután a beállítás gombra kattintva az üzenet azonnal mentődik és meg is jelenik az összes olyan eszköznek, amely az adott cellában tartózkodik vagy fog tartózkodni. A meghtározott idősávban az aktuális üzenetek a 4.15. ábrán látható táblázatban jelennek meg. Ha esetleg a lejárati idő előtt szeretnének törölni egy üzenetet azt a Törlés gomb segítségével tudjuk megtenni, amúgy minden üzenetre igaz, hogy a lejárati ideje után nem jelenik meg a résztvevőknek.



4.13. ábra. *Térképes felület.*



4.14. ábra. *Új üzenet hozzáadása felület.*



4.15. ábra. *Aktuális üzenetek listájának felülete.*

5. fejezet

Szimulációs vizsgálat

A szimulációs vizsgálatok során az volt a célom, hogy megbizonyosodjak arról, rendszerem helyesen működik-e, illetve mi az a maximális terhelés amely mellett még képes megfelelő minőségben működni alacsony késleltetéssel. A tesztek elvégzéséhez szükség volt egy eszköz szimulátor készítésére, ami képes arra, hogy több ezer mobil eszközt szimuláljon, amelyek amellett, hogy megfelelően képesek kommunikálni a rendszerrel, olyan algoritmus szerint mozognak, amely jól utánozza az emberek mozgását.

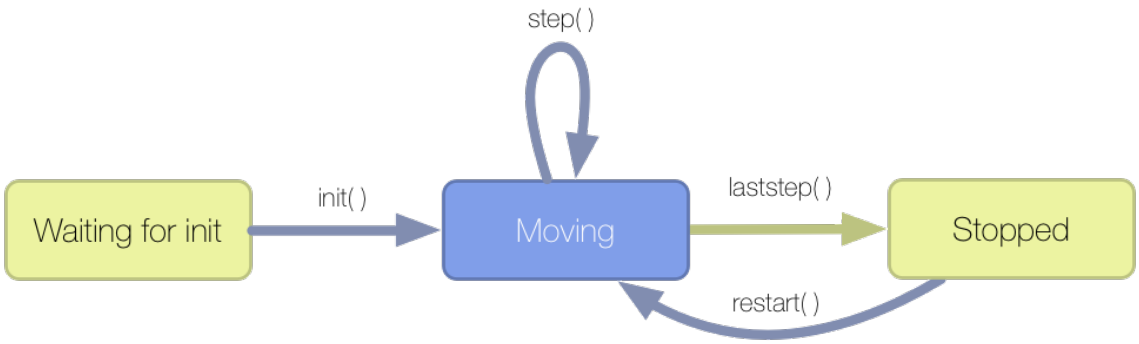
5.1. Az eszköz szimulátor

Két különböző szimulátort készítettem el, ugyanis a teljes kommunikáció elérése a rendszerrel nagyon erőforrás igényes, egyszerre körübelül csak 100 eszközt tudtam volna leszimulálni ilyen esetben. Ennek oka, hogy ilyenkor minden eszköznek szükséges fenntartania folyamatosan egy TCP kapcsolatot a Push Notification Service-szel, tudnia kell TCP és UDP mérési üzeneteket küldenie (természetesen ez nem igényel folytonos kapcsolatot), illetve figyelnie is kell, hogy a Traffic Aggregator Service nem küld-e vissza hibaüzenetet (ez csak UDP esetén jelent pluszt). Mivel az első, teljes kommunikációt megvalósító szimulátor nem volt képes arra, hogy egy valós tömeget szimuláljon szükséges volt egy második elkészítésére is, amely erre már képes, viszont annyi könnyebbséggel, hogy nem kell figyelnie a TrafficAggregatorService irányából érkező hibaüzenetekre. Ettől az egyszerűsítéstől továbbra is realisztikus marad a szimulátor, hiszen hibaüzenetek csak akkor érkezhetnek a TAS felől, ha nem megfelelő a mérési üzenetben feltüntetett térkép verziószáma. Ilyen helyzet jellemzően csak nap elején fordulhat elő vagy ha a felhasználó jóval később kapcsolja vissza az alkalmazást térkép verzió frissítés után. Ezek száma elenyésző a többi kommunikáció mennyiségéhez képest.

Több ezer eszköz párhuzamos szimulációja, amik folyamatosan kapcsolatban állnak a rendszerrel nem megoldható szinkron módon, hiszen minden esetben minimum a szimulált készülékek számával megegyező szálra van szükség a kommunikációhoz is, ami teljesen kimerítené az erőforrásokat. Emiatt mindenképpen aszinkron kellett megvalósítanom a hálózati kommunikációt, illetve a szimulált eszköz példányok sem futhattak külön-külön

szálban, hanem eseményvezérelten kellett végrehajtani a működésüket.

Ehhez először definiáltam az események halmazát, amelyek bekövetkezhetnek egy eszköznél. Indulásnál inicializálni kell önmagát ($INIT$), tehát le kell töltenie az aktuális térképet és fel kell építenie a cellák szomszédsági mátrixát, ami alapján a későbbiekben tudni fogja, hogy mely irányokba mozoghat. Ezután képes lépéseket szimulálni ($STEP$), amíg ki nem kapcsolja magát, ilyenkor lép egy utolsót ($LAST\_STEP$) majd alvó módba kerül. Innen egy bizonyos valószínűséggel még a későbbiekben visszatérhet ($RESTART$) és folytathatja a lépések szimulálását. (lásd 5.1. ábra)



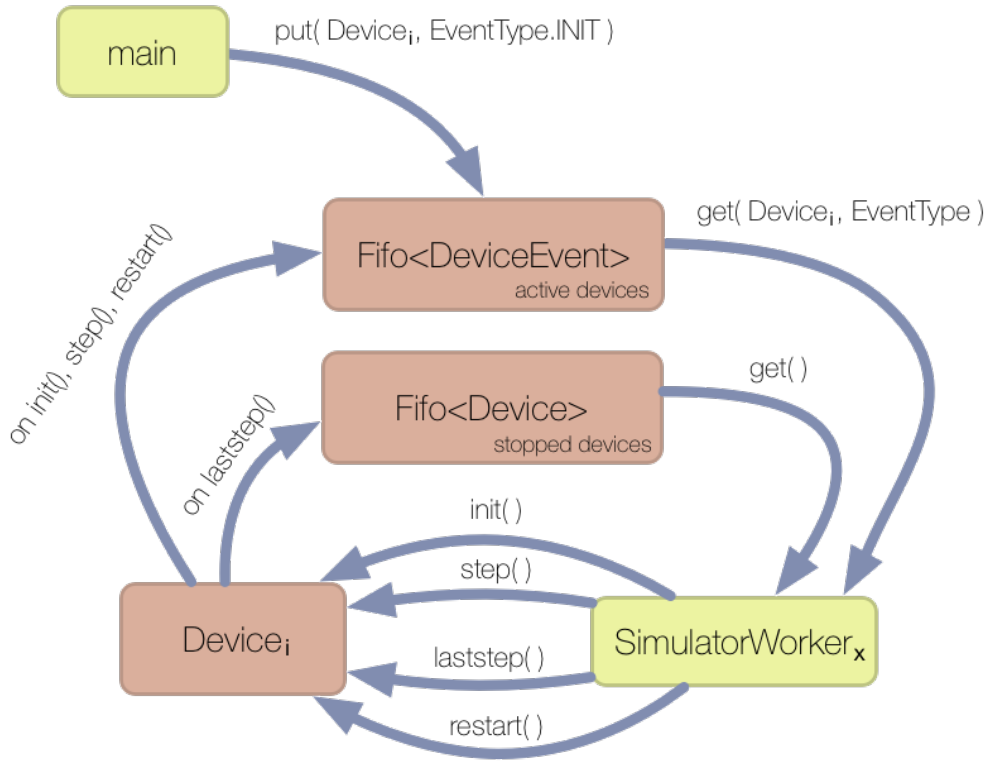
5.1. ábra. *Egy eszköz állapotdiagramja.*

Az esemény halmaz definiálása után szükség volt még arra, hogy az eszközöket képessé tegyem arra, hogy könnyen tudjanak különböző típusú mozgásokat szimulálni, hiszen rendezvénytől, szituációtól vagy akár kortól függően az emberek más és más módon közlekednek. Ennek a megvalósításához egy mozgásszimulátor felületet kellett definiálnom, amelyen keresztül egységesen elérhetőek a különböző típusok.

Amikor elindul egy szimuláció, a szimulátor paraméterben megkapja, hogy hány darab és milyen mozgástípust követő eszközt kell szimulálnia. Miután példányosította a megfelelő számú készüléket, beteszi őket egy eszköz-esemény párokat tároló FIFO-ba. Innen a különböző események végrehajtásának vezérlését az úgynevezett SimulationWorker-ek veszik át, melyek példányai különböző szálakban futnak és feladatuk, hogy végrehajtsák az eszközökön a kiválaszott eseményeket. Ha egy eszköz úgy dönt, hogy ki szeretne lépni, akkor a laststep() meghívása után a leállított eszközök FIFO-jába kerül, ahonnan a későbbiekben még adott valószínűséggel visszakerülhet. (lásd 5.2. ábra)

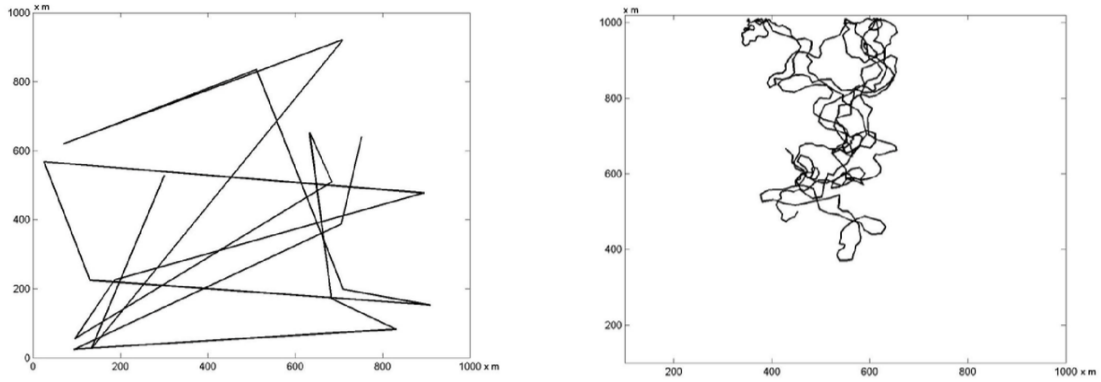
5.2. Mozgást szimuláló algoritmusok

A mozgást szimuláló algoritmusok implementálásánál szerettem volna jól bevált megoldásokat alkalmazni. Emiatt olyan algoritmusokat kerestem, melyeket a mobil távközlésben alkalmaznak mobil cellás felbontások vizsgálatához. Ezek közül én kettőt választottam ki, a Random Waypoint és a Gauss-Markov mobilitás modellt, amelyeket módosítanom kellett, hogy az általam kialakított környezetben használhatóak legyenek. Mindkét modell esetén feltételezem, hogy egy eszköz már letöltötte az aktuális térképet és felépítette belőle a



5.2. ábra. *Szimulációs motor architektúrája.*

cellák szomszédsági mátrixát.



5.3. ábra. *A bal oldali ábrán az eredeti Random Waypoint modell mozgási mintázatát lehet látni, amíg a jobb oldalin a Gauss-Markov modellét.*

5.2.1. Módosított Random Waypoint mobilitás modell

Az eredeti modellt először a DSR protokoll teljesítmény tesztelénél alkalmazták [26]. Itt a mobil csomópontok egy véletlenszerűen választott pontból kezdik meg a mozgásukat. Innen elindulnak egy véletlenszerűen sorsolt pontba $[0, MaxSpeed]$ közötti egyenletes sebességgel,majd ha odaérnek kivárnak egy az inicializálásnál meghatározott időtartamot mielőtt újra elkezdenek mozogni.

Az általam kifejlesztett tömegdinamikai modell szempontjából feleslegesen bonyolult az, hogy egy pontot határozzunk megs a rendezvény területén ahova az eszköz el szeretne jutni, mert akkor valamilyen legrövidebb út kereső algoritmussal meg kéne határoznom az útvonalon érintett cellákat, ehelyett elegendő, ha csak a szomszédos cellák közül választok egyet. Az eszköz kiindulási pontja mindig egy bejárat cellája az adott rendezvény területén, mozgásnál pedig minden lépésben négy feladatot kell végrehajtania, feltételezve hogy jelenleg a $Cell_i$ cellában tartózkodik:

1. Le kell kérdeznie $Cell_i$ cella szomszédjainak listáját
2. Ezek közül véletlenszerűen választ egy $Cell_x$ cellát
3. Ki kell számolnia a sebesség ingadozását vel_dif_{act}, amely egyenletes eloszlás szerint egy 0 és 0,2 m/s közötti érték [25].
4. Ki kell számolnia az áthaladási időt, hogy mennyi idő alatt fog átérni $Cell_i$ cellából $Cell_x$ cellába, amit a következő képlet alapján számol ki:

$$t = \frac{|\overrightarrow{Cell_i.centroid, Cell_x.centroid}|}{1,3\frac{m}{s} + vel\_dif_{act}} \tag{5.1}$$

5. Ha ezekkel végzett, a kiszámolt értékeket elküldi a rendszernek, majd vár míg letelik az áthaladási idő és visszatér az első lépéshez.

5.2.2. Módosított Gauss-Markov mobilitás modell

A Gauss-Markov mobilitás modellt [27] Liang és Haas mutatta be, amit aztán több kutatásban is felhasználtak. Lényeges különbség az előző algoritmushoz képest, hogy itt az aktuális irány és sebesség függ az előző állapottól az alábbi képlet alapján.

$$s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + (1-\alpha^2)\sqrt{s_{x_{n-1}}} \tag{5.2}$$

$$d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + (1-\alpha^2)\sqrt{d_{x_{n-1}}} \tag{5.3}$$

s_n és d_n a sebesség és az irány értéke n időpillanatban. s_{n-1} és d_{n-1} a sebesség és az irány értéke az $n-1$ időpillanatban, α konstans érték $[0,1]$ intervallumon értelmezve. Minél nagyobb az értéke annál jobban függ az aktuális állapot az előzőtől. $\bar{s}$ és $\bar{d}$ is konstans értékek, melyek az átlagos sebességet és átlagos irányt jelölik. $s_{x_{n-1}}$ és $d_{x_{n-1}}$ Gauss eloszlású változók.

Változásokat kellett eszközölnöm az algoritmus további részeiben, illetve meg kellett határoznom a konstans értékeket, hogy a modell illeszkedjen a cellás felbontásomhoz. Az eszköz kiindulási pontja itt is mindig egy bejárat cellája az adott rendezvény területén. Ezután minden lépésben 5 feladatot kell végrehajtania, feltételezve hogy jelenleg a $Cell_i$ cellában tartózkodik az n. időpillanatban:

1. Ki kell számítania a következő d_n irányt

2. Le kell kérdeznie, hogy $Cell_i$-től viszonyítva a d_n irányban melyik a következő $Cell_x$ cella. Előfordulhat, hogy d_n irányban már nincsen szomszéd. Ilyenkor véletlenszerűen választ egy cellát a szomszédos cellák között (feltételezhető, hogy nincsenek izolált cellák a rendszerben).

3. Ki kell számítania ezután az s_n sebességet

4. Ki kell számolnia az áthaladási időt, hogy mennyi idő alatt fog átérni $Cell_i$ cellából $Cell_x$ cellába, amit a következő képlet alapján számol ki:

$$t = \frac{|\overrightarrow{Cell_i.centroid, Cell_x.centroid}|}{s_n} \tag{5.4}$$

5. Ha ezekkel végzett, a kiszámolt értékeket elküldi a rendszernek, majd vár míg letelik az áthaladási idő és visszatér az első lépéshez.

5.3. Teljesítmény-hatékonysági tesztek

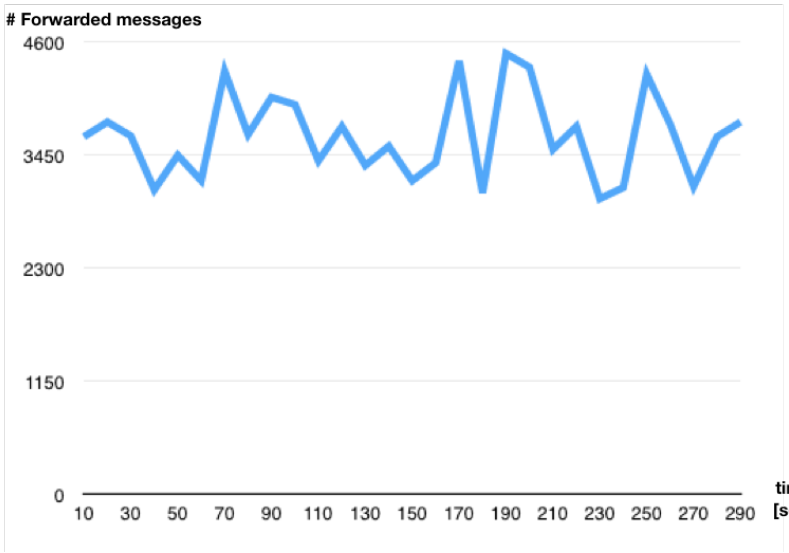
Miután elkészült a szimulátor és implementáltam a mozgástípusokat, első lépésben megyőződtem arról, hogy a különböző komponensek tényleg helyesen működnek-e együtt, majd a rendszert teljesítmény-hatékonysági teszteknek vetettem alá. Mivel nem volt lehetőségem erősebb szerveren tesztelni, a saját gépemen, egy 2013-as Macbook Air-en (CPU: 1.3GHz i5, Memória: 4GB 1600Mhz DDR3) futtattam a teszteket, ahol egyszerre működött a rendszer minden komponense (természetesen képesek fizikailag külön is működni) és az eszköz szimulátor.

5.1. táblázat. *Rendszer beállítások a tesztek elvégzésénél.*

Rendszer komponens	Mennyiség	Megjegyzés
TrafficAggregatorService	1 db	normál üzemmódban
SparkEvaluationService	1 db	10 másodperces kiértékelési ablakok
PushNotificationController	1 db	-
PushServer	1 db	-

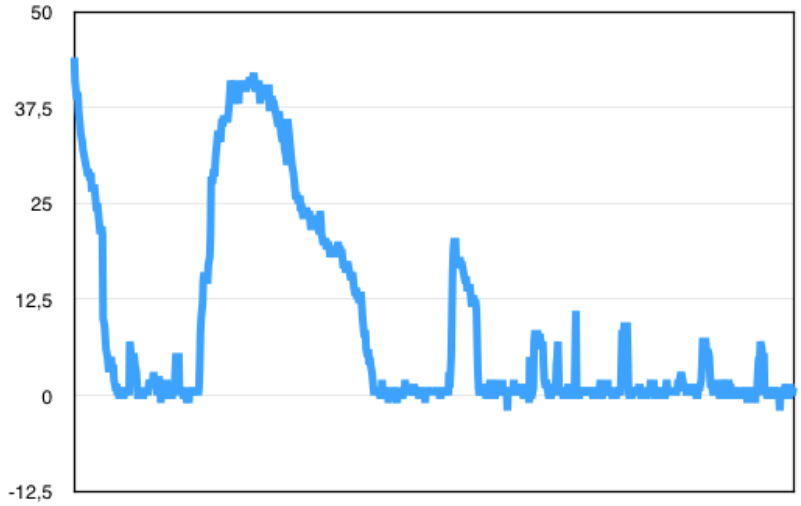
Sajnos a tesztek során belefutottam egy korlátba, ahol az operációs rendszer maximálisan 2560 fájl leírót engedett hozzárendelni egy folyamathoz, ami nagymértékben lecsökkentette a szimulálható eszközök számát. Ezt az értéket át is írhattam volna, viszont erre nem volt szükség. Mivel ekkor már meggyőződtem arról, hogy tényleg minden rendben működik nem kellett érvényes adatokat küldjek a szervernek, hiszen a teljesítmény vizsgálatánál elég ha csak ugyanazt az üzenetet küldöm el, az üzenetek tartalma nem befolyásolja a rendszer teljesítményét. Sőt ebben a megközelítésben nem kell új cella pozíciókat, sebességeket, esetleg irányokat számolni, így jóval több üzenettel lehet terhelni a rendszert. Az első teljesítménytesztek után úgy tűnt, hogy a teszthardverrel nem tudom elérni a rendszer teljesítőképességének felső határát, de hogy erről megbizonyosodjak meg kellett vizsgálnom, hogy nem-e a Traffic Aggregator Service viselkedik szűk keresztmetszetként a rendszerben, mert elképzelhető volt, hogy csak annyi üzenetet engedett át, amit a többi komponens még

le tudott kezelni.



5.4. ábra. *Másodpercenként 3500-4000 üzenet képes továbbítani egy TrafficAggregator példány. 1 millió mérési üzenetet 290 másodperc alatt tudott átengedni.*

Hogy részletesebben megvizsgálhassam a TrafficAggregatorService teljesítményét felvértem egy Benchmark modullal, aminek segítségével megmértem, hogy a mérési üzenet FIFO mennyire van megtelve egy adott pillanatban, másodpercenként hány üzenetet fogad a komponens, illetve mennyi idő alatt ér át az eszköztől az üzenet a kiértékelő szerverhez.



5.5. ábra. *Késleltetés az eszközök és a kiértékelő szerver között.*

Minden teszt esetén 1 millió üzenetet küldtem a rendszer felé, amely 4,5 - 5 perc alatt ért át rajta. Minden esetben a FIFO teljesen üres volt a szimuláció közben, tehát a rendszerben nem torlódtak fel az üzenetek és tényleg csak a teszthardver felső korlátait értem

el, nem pedig a rendszerét. Továbbá azt tapasztaltam, hogy másodpercenként 3500-4000 üzenet folyt keresztül a rendszeren (5.4. ábra), ami nagyon jó eredmény, hiszen ez az üzenet mennyiség egy valós környezetben is elképzelhető. Az eszköz és a kiértékelő szerver közötti késleltetések vizsgálatánál egy nagyon érdekes jelenségre figyeltem fel. Miután felállt a rendszer és elindult az üzenet folyam eleinte relatíve magas, akár 38 ms késleltetést is mértem, majd az első 2000-3000 üzenet után csillapodtak az értékek, végül 4000-5000 üzenet után már 0-1 ms között mozogtak (A 5.5. ábrán csak az első 10000 üzenetet ábrázoltam, hogy az ingadozás jólláthatóbb legyen). Ennek oka a Java Hotspot adaptív optimalizációja, ahol az aktuális végrehajtási profil alapján optimalizálja a bájt kódot, de ennek hossza elenyésző, körübelül 2-3 másodpercet vesz igénybe.

6. fejezet

Konklúzió

Manapság a tömegrendezvények óriási látogatottságnak örvendenek, ugyanakkor számtalan veszélyt is hordoznak magukban. Nagy tömegekben egy kisebb pánik is beláthatalan következményekkel járhat. Ezek a váratlan helyzetek bárhol és bármikor kialakulhatnak, melyek a rendfenntartó szervektől a lehető leggyorsabb reakciót kívánják. Különböző rendezvények más-más elvárásokat támasztanak a szervezők elé, de minden esetben garantálni kell az eseményen résztvevő emberek biztonságát. Sajnos minden évben előfordulnak tömegkatasztrófák, melyeknek egy része megelőzhető lenne, ha a szervezők számára több információ állna rendelkezésre, bár vannak olyan előre nem megjósolható események is, amelyekre nagyon nehéz és körülményes felkészülni.

A technológiai fejlődésnek köszönhetően a látogatók nagy része már rendelkezik olyan mobil készülékkel, amelyben különféle szenzorok találhatóak. Ezek segítségével olyan nem személyes jellegű adatokat lehet gyűjteni, amelyeket kiértékelve átfogó képet kaphatunk a rendezvényen lévő tömeg aktuális állapotáról.

Léteznek már olyan rendszerek, amelyek tömegfelügyelettel foglalkoznak de ezeknél sokkal pontosabb és hatékonyabb rendszer fejleszthető. Ehhez első lépésben megismerkedtem a szakirodalomban szereplő tömegmozgási modellekkel. Ezek azt tűzik ki célul, hogy megpróbálnak minél valósághűbb leírást adni a tömeg mozgására (folyamatos és diszkrét esetben is), hogy a velük végzett szimulációkkal evakuációkat tervezhessenek vagy észrevegyenek olyan pontokat egy alaprajzon, amelyek veszélyesek lehetnek. Sajnos ezek alkalmatlanok olyan helyzetekben, amikor nem az emberek mozgását kell reprodukálni, hanem a gyalogosoktól érkező valós adatokat kell belehelyezni a modellbe, és azokat kiértékelni.

Emiatt létrehoztam egy olyan tömegfelügyeleti modellt, amely erre alkalmas. Ezek után kifejlesztettem egy olyan komplex tömegfelügyeleti rendszert, amely képes a résztvevők készülékeiktől másodpercenként több ezer üzenet fogadására, ezek kiértékelésére illetve üzenetek visszaküldésére a pozíciók alapján. A rendszerhez készítettem egy webes felhasználói felületet is, amin keresztül a szervezők egy ergonomikus interfészen képesek bepillantást nyerni a tömeg aktuális állapotába.

Az implementált rendszer végül validációs és teljesítmény teszteknek vetettem alá, amelyek igazolták, hogy több ezres felhasználói bázis esetén is működőképes a rendszer és valós időben képes feldolgozni állapot üzenetek ezreit. Célom, hogy az általam fejlesztett tömegfelügyeleti rendszert élesben is teszteljem a jövőben.

Irodalomjegyzék

[1] `http://www.securitynewsdesk.com/how-many-cctv-cameras-in-the-uk/` (2015 október 18.)

[2] Donna Nelson, „Proximity Information Resources for Special Events", *Traffax Inc.*, 2012

[3] `http://www.traffaxinc.com/content/blufax-concept` (2015 október 18.)

[4] Jie Yin, Andrew Lampert, Mark Cameron, Bella Robinson, és Robert Power, „Using Social Media to Enchance emergency Situation Awareness", *IEEE Intelligent Systems*, vol. 27, no. 6, pp. 52–59, 2012

[5] Martin Wirz, Eve Mitleton-Kelly, Tobias Franke, Vanessa Camilleri, Matthew Montebello, Daniel Roggen, Paul Lukowicz, and Gerhard Troster, „Using Mobile Technology and a Participatory Sensing Approach for Crowd Monitoring and Management During Large-Scale Mass Gatherings", *Co-evolution of Intelligent Socio-technical Systems*, pp. 61–77, 2012

[6] Dirk Helbing, Anders Johansson and Habib Zein Al-Abideen, „The Dynamics of Crowd Disasters: An Empirical Study", *Physical Review E*, vol. 75, iss. 4, 2007

[7] Anders Fredrik Johansson, *Data-Driven Modeling of Pedestrian Crowds*, Drezdai Műszaki Egyetem, PhD disszertáció, 2009, pp. 38-42.

[8] D. Helbing, L. Buzna, A.Johansson, and T. Werner *Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions, Transportation Science*, vol. 39, 2005, pp. 1-24

[9] D.Helbing, *A mathematical model for the behavior of individuals in a social field*, Journal of Mathematical Sociology, vol. 19, no. 3, 1994, pp. 189-219

[10] D. Helbing, J. Keltsch, P. Molnár, *Modelling the evolution of human trail systems*, Nature 388, 1997, pp. 47-50.

[11] Anders Johansson, Dirk Helbing, Z. Al-Abideen and Salim Al-Bosta *From Crowd Dynamics to Crowd Safety: A Video-Based Analysis*, Advances in Complex Systems 11(4), pp. 497-527.

[12] L.F. Henderson., *On the fluid mechanics of human crowd motion, Transportation Research* , vol. 8, no. 6, 1974, pp. 509-515.

[13] D. Helbing, *A fluid-dynaminc model for the movement of pedestrians, Complex Systems*, vol. 6, 1992, pp. 391-415.

[14] D. Helbing, I. Farkas and T. Vicsek, *A menekülési pánik dinamikai tulajdonságainak szimulációja*, Fizikai Szemle, vol. 50, no. 10., 2000

[15] Andreas Schadschneider, Wolfram Klingsch, Hubert Klüpfel, Tobias Kretz, Christian Rogsch , Armin Seyfried, *Evacuation Dynamics: Emprirical Results, Modeling and Applications*, Springer, 2008

[16] Ulrich Weidmann, *Transporttechnik der Fussgaenger - Transporttechnische Eigenschaften des Fussgaengerverkehrs (Literatu rauswertung)*, ETH Zürich, Zürich, 1993

[17] `https://hadoop.apache.org/`(2015 október 18)

[18] `https://spark.apache.org`(2015 október 18)

[19] `http://netty.io`(2015 október 18)

[20] `http://javolution.org`(2015 október 18)

[21] `https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html`(2015 október 18)

[22] `https://developers.google.com/cloud-messaging/`(2015 október 18)

[23] `http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html`(2015 október 18)

[24] `https://dev.mysql.com/doc/refman/5.0/en/myisam-storage-engine.html`(2015 október 18)

[25] Sebestyén Varga, *Gyalogos mozgásvizsgálatok helyszíni felvételek alapján*, Budapesti Műszaki Egyetem, TDK dolgozat, 2006, pp. 22

[26] David B. Johnson, David A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, The Kluwer International Series in Engineering and Computer Science, vol. 353, pp. 153-181, 1996

[27] Liang and Z. Haas, *Predictive Distance-based MobilityManagement forPCS Networks*, IEEE INFOCOM 1999, vol. 3, pp. 1377-1384, 1999