



SVEUČILIŠTE U ZAGREBU



Fakultet
elektrotehnike i
računarstva

Diplomski studij

Računarstvo

Akademska godina
2022/2023

Umrežene igre

Umrežena simulacija



Pozvano predavanje – potvrđeno

- Dobrila Šunde – Ubisoft
- <https://www.linkedin.com/in/dobrila-sunde?originalSubdomain=hr>
- Primijenjeno računarstvo u izradi igara
- Pozvano predavanje će se održati 20. 12. 2022. u 18:00
- Moguće da će biti promjena dvorane (Siva vijećnica)
- O promjeni ćete biti pravovremeno obaviješteni

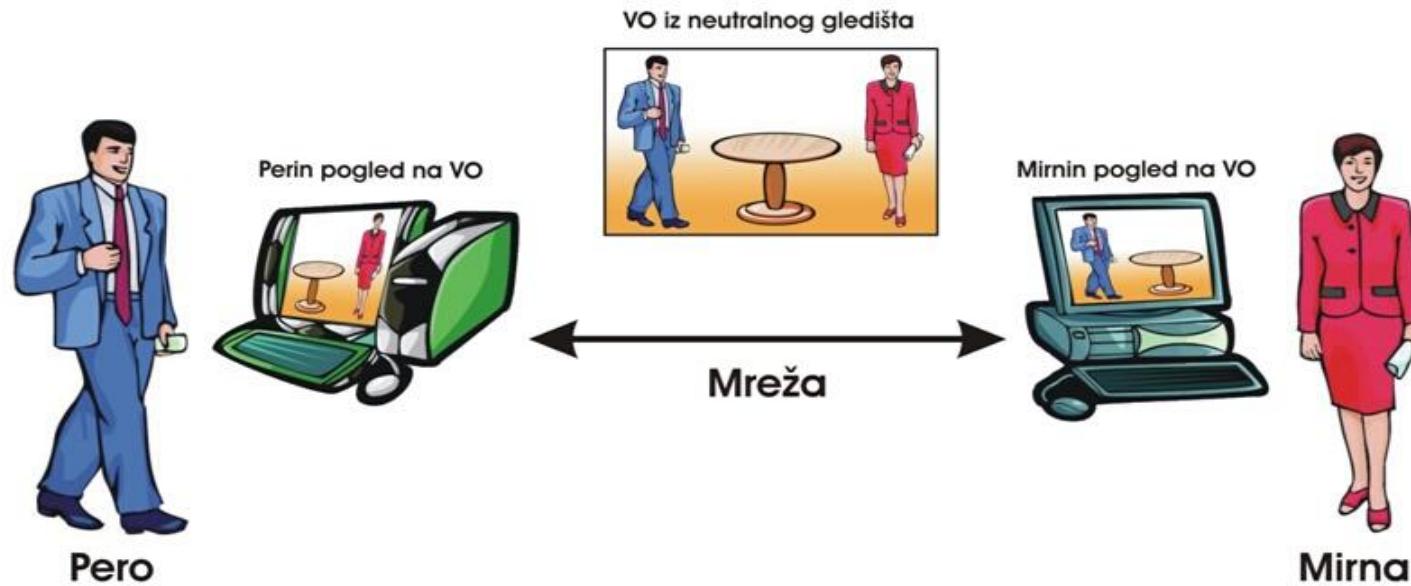
Sadržaj

- Umrežene igre
- Osnovni koncepti umrežene simulacije
- Zajedničko dinamičko stanje i konzistentnost
- Postupci za održavanje konzistentnosti

Terminologija

- **Umrežena virtualna okruženja (UVO)** (engl. **Networked Virtual Environments, NVE**) su sustavi koji omogućavaju da više fizički udaljenih korisnika sudjeluje u zajedničkom virtualnom okruženju
- **Umrežene videoigre** možemo definirati kao videoigre koje se igraju putem računalne mreže
- **Višekorisničke videoigre** možemo definirati kao videoigre u kojima sudjeluje više igrača.
- **Višekorisničke umrežene videoigre** su videoigre u kojima sudjeluje više igrača, a igraju se putem mreže.
- „Anomalije“
 - Diablo 3 je imao nužnu konekciju prema poslužitelju i za samo jednog igrača
 - Igre temeljene na računalnom oblaku su sve nužno umrežene iako ne moraju uključivati više igrača
 - Mnoge igre za mobitel zahtijevaju barem periodičku vezu prema poslužitelju iako su za jednog igrača

Umrežena virtualna okruženja



- Fizički udaljeni korisnici sudjeluju u zajedničkom virtualnom okruženju
- Svaki korisnik upravlja svojim 3D prikazom i okruženjem
- Sve kopije okruženja se međusobno sinkroniziraju putem mreže
- Korisnici vide jedni druge jer su grafički prikazani u okruženju

Kako korisnici doživljavaju UV-a?

- Doživljaj zajedničkog prostora
 - Korisnici osjećaju da su u istom (stvarnom ili zamišljenom) prostoru
- Doživljaj zajedničkog prisustva
 - Svaki od korisnika ima svoju reprezentaciju unutar virtualnog svijeta
- Doživljaj zajedničkog vremena
 - Događaji (izgledaju kao da) se događaju u isto vrijeme
- Mogućnost komunikacije
- Mogućnost interakcije
 - S virtualnim svjetom i drugim korisnicima

Primjena

- Virtualni svjetovi – popularni u 2000-tima, danas ih ponovno popularizira virtualna stvarnost
 - Druženje i društvene mreže (Horizon worlds- <https://www.oculus.com/horizon-worlds/>)
 - Virtualna telekonferencija/zajednički rad (Virtrend - <https://www.3dicc.com/>)
 - Učenje/obuka na daljinu (Second Life - <https://secondlife.com/>)
 - Virtualni svjetovi za djecu (Club Penguin – ugašen, ali postoji još slobodna verzija New Club Penguin <https://newcp.net/en/>)
 - ...
- Umrežene višekorisničke igre – primarna svrha je zabava



UMREŽENA VIRTUALNA OKRUŽENJA

Povijest umreženih videoigara

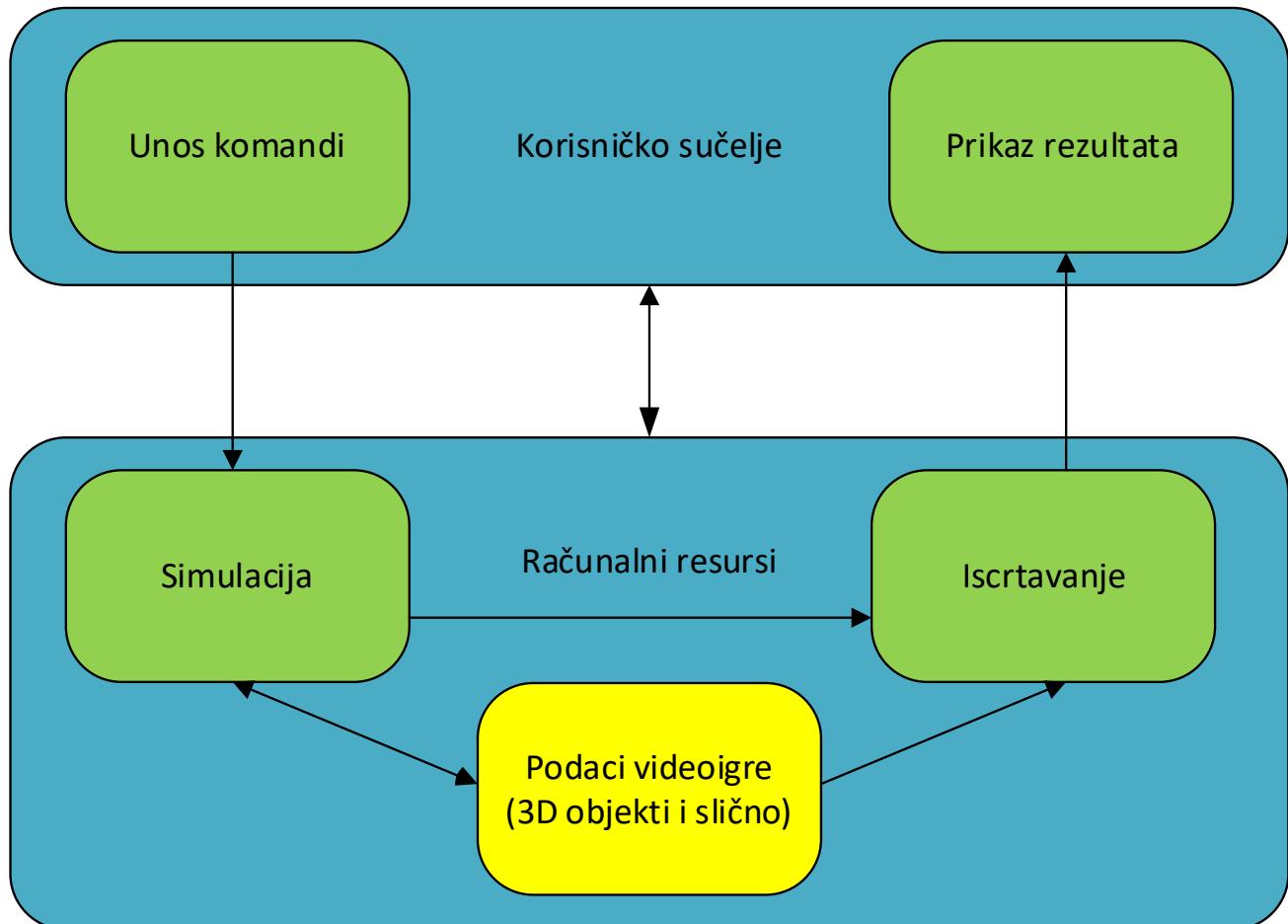
- **Videoigra Maze War**
 - Razvijena u periodu 1972.-73. godine
 - Igrači su lutali u labirintu te su mogli pucati jedni na druge
 - Prva videoigra u tri dimenzije (3D) u kojoj su igrači imali perspektivu iz prvog lica
 - Prva koja je se mogla igrati putem lokalne mreže između dva ravnopravna računala
 - Prva videoigra koja je kasnije implementirala koncept poslužitelja i klijenta
- **Višekorisničke tamnice (engl. Multi-User Dungeons skr. MUD)** su među prvim perzistentnim umreženim svjetovima iako su bile temeljene na tekstu te poznatoj igri na ploči Dungeons & Dragons
 - Tekstualne igre
 - Preteča MMORPG-ova



Osnovni koncepti umrežene simulacije

Osnovne funkcije

- Unos komandi,
 - tipkovnica
 - miš
 - ekran osjetljiv na dodir
 - igrači kontroler (engl. gamepad)
 - igrača palica (engl. joystick)
 - kontroler za miješanu stvarnost
 - dubinske kamere (primjerice Kinect)
- Simulacija za izračun novog stanja videoigre,
- IsCRTavanje novog stanja videoigre
- Prikaz novog stanja videoigre.
 - monitor
 - televizor
 - zaslon tableta
 - zaslon mobitela
 - zasloni naočala za miješanu stvarnost
 - slično



Punjene kantice

- Zamislimo jednostavnu videoigru **Punjene kantice**
 - Za od dva do četiri igrača
 - Na stolu stoje dvije to četiri prazne kantice (ovisno o broju igrača)
 - Sa svake četiri strane sobe su slavine koje toče vodu te je svakom igraču dodijeljena njegova slavina
 - Svaki od igrača ima za cilj napuniti kanticu koristeći postojeće čaše i prenoseći vodu od slavina do sredine sobe te punеći svoju kanticu

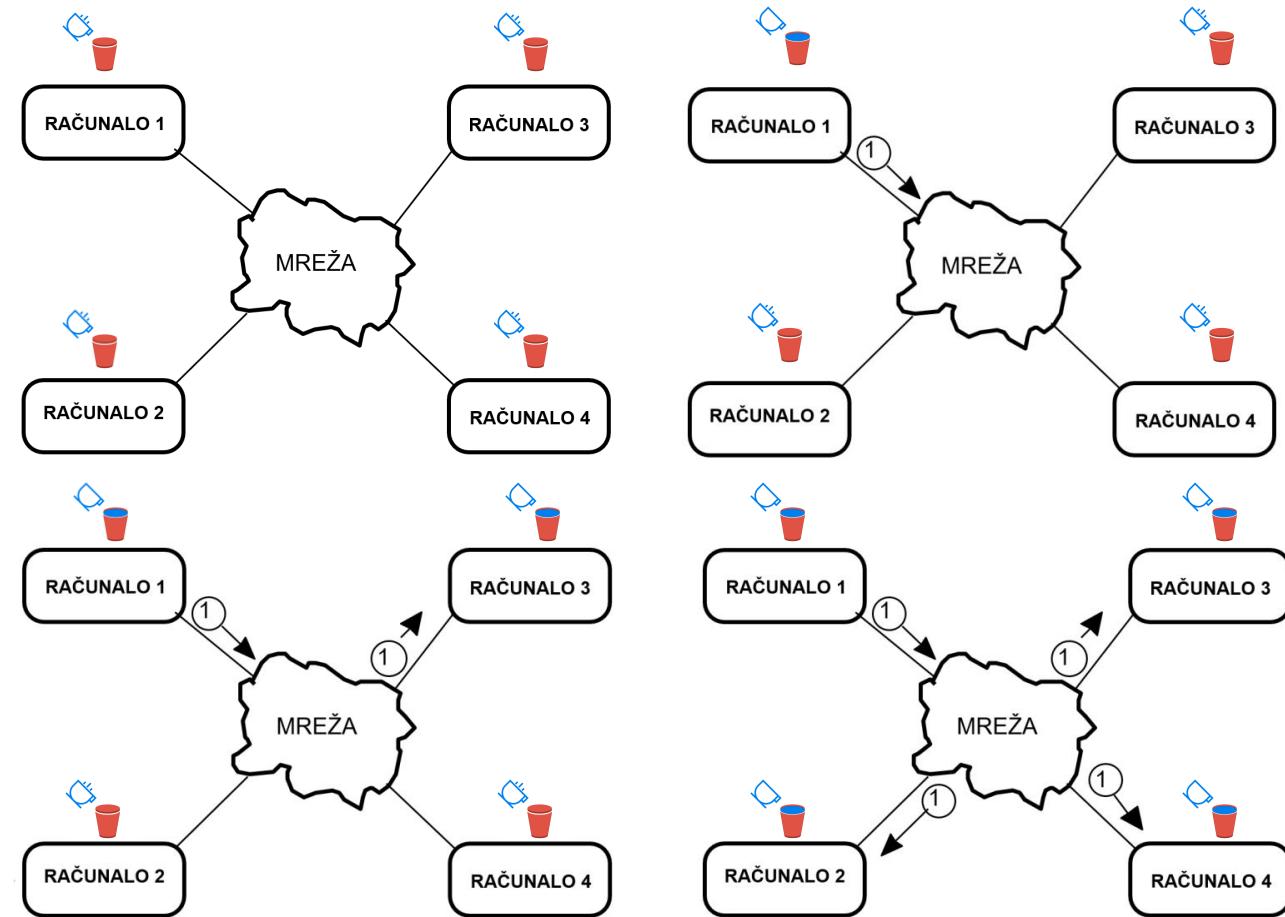


Tipovi informacija

- **Promjenjive informacije** su one koje se tijekom izvođenja igre mogu promijeniti, a to su u ovom slučaju
 - pozicija i orientacija svakog od igrača
 - količina vode u kantici svakog igrača
 - je li čaša igrača puna ili prazna
- **Nepromjenjive informacije** su one koje se tijekom izvođenja igre ne mogu promijeniti, a to su:
 - izgled igrača, stola i slavina
 - pozicija i orientacija stola, slavina i kantica
- Kako znamo koliko je puna kantica drugog korisnika? Kako znamo drži li praznu ili punu čašu? Dane informacije naša simulacija mora nekako dobiti, a to se realizira putem mreže.

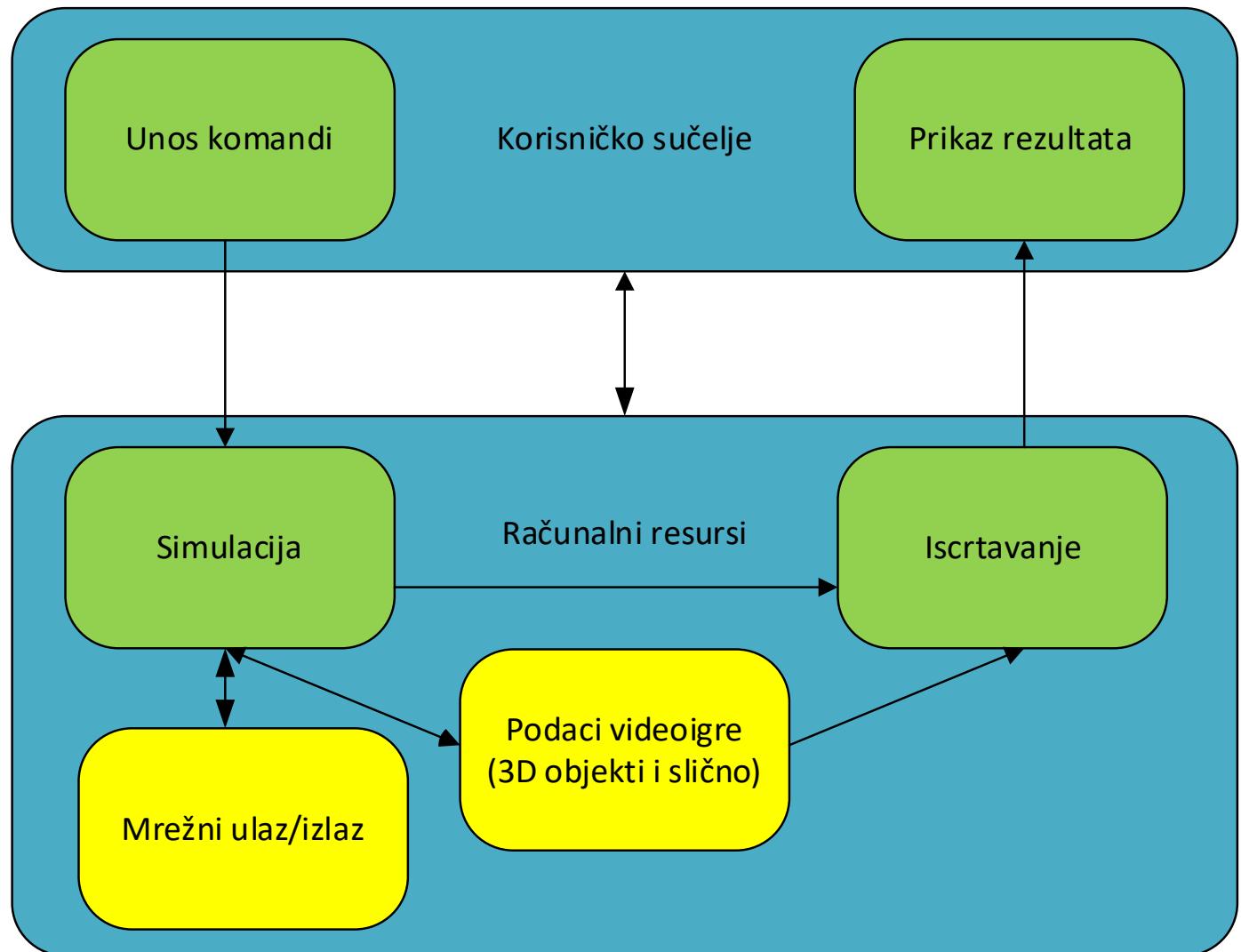
Razmjena informacije putem mreže

- U videoigri sudjeluju četiri korisnika svaki na svom računalu.
- Kada jedan od korisnika prelije svoju čašu u svoju kanticu u njoj se povećava količina vode
- Informacija o toj novoj količini vode, te da je korisnikova čaša sada prazna mora se prenijeti preko mreže
- Ona se spremi u mrežni paket i putem mreže dolazi do drugih računala koja onda osvježe i stanje u svojim simulacijama i prikažu ga
- Informacija dolazi s odgodom ovisno o mrežnom kašnjenju

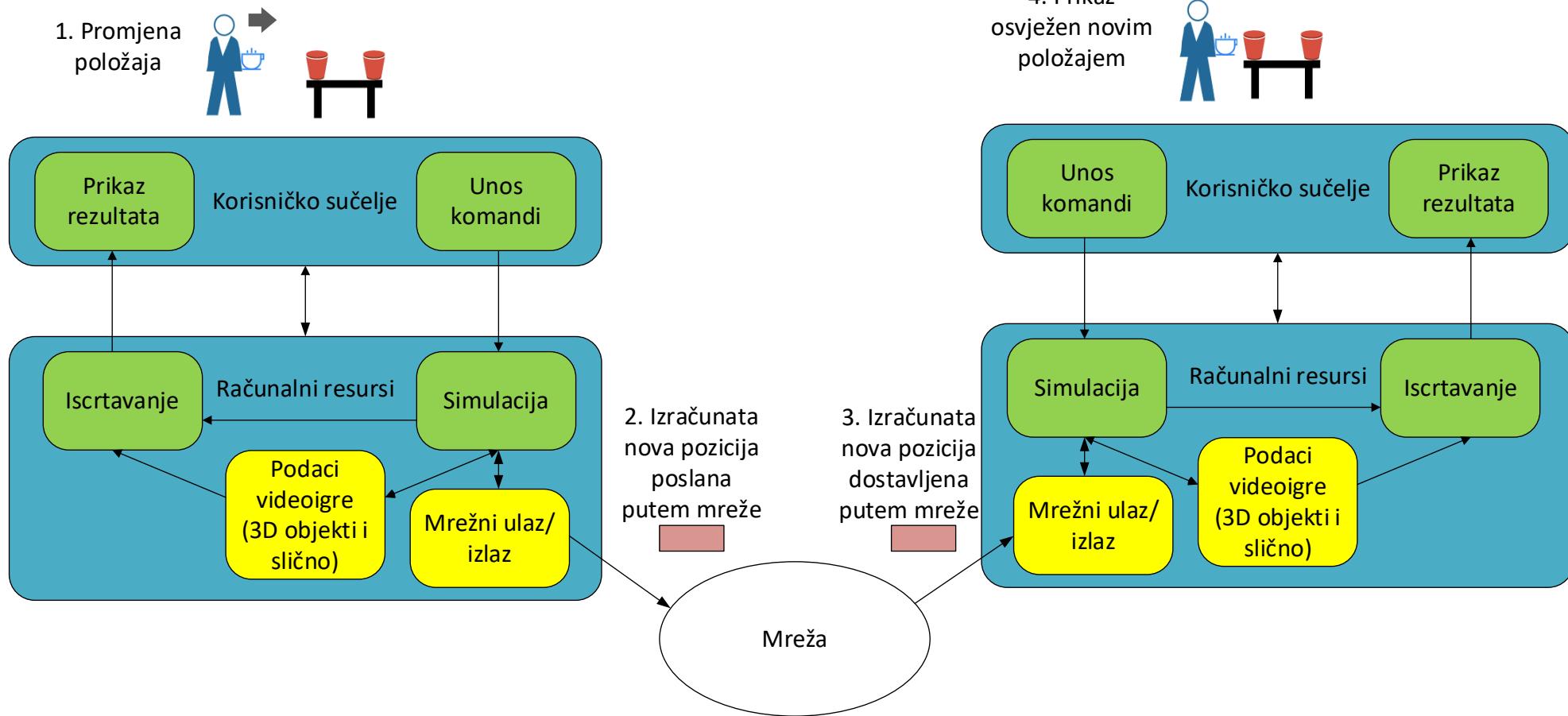


Prošireni model osnovnih funkcija

- Pošto dodatne informacije dolaze od udaljenih računala putem mreže potrebno je dodati poveznicu novu na simulaciju
- Na mrežni ulaz/izlaz dolaze informacije iz drugih simulacija odnosno šalju se informacije od naše simulacije
- Ovaj osnovni model je repliciran u većini umreženih igara



Prošireni model osnovnih funkcija u mreži



Prošireni model osnovnih funkcija u mreži

- Svaki korisnik ima pokrenutu instancu VO
- Ne moramo slati nepromjenjive informacije (npr. teksture, gotovi 3D objekti)
- Potrebno je dostaviti promjenjive informacije u razumnom roku
 - Razuman rok ovisi o tipu virtualnog okruženja i razini interaktivnosti
 - Primjerice, šah je dovoljno dostaviti informaciju kada se pomjeri figura (može biti samo jednom u par minuta), dok za igre gađanja to mora biti više desetaka puta u sekundi
- Problemi
 - Promet raste s brojem korisnika – skalabilnost (adresirat ćemo na kasnijim predavanjima)
 - Kašnjenje nije isto za sve korisnike – nekonzistentnost

Zajedničko dinamičko stanje i konzistentnost

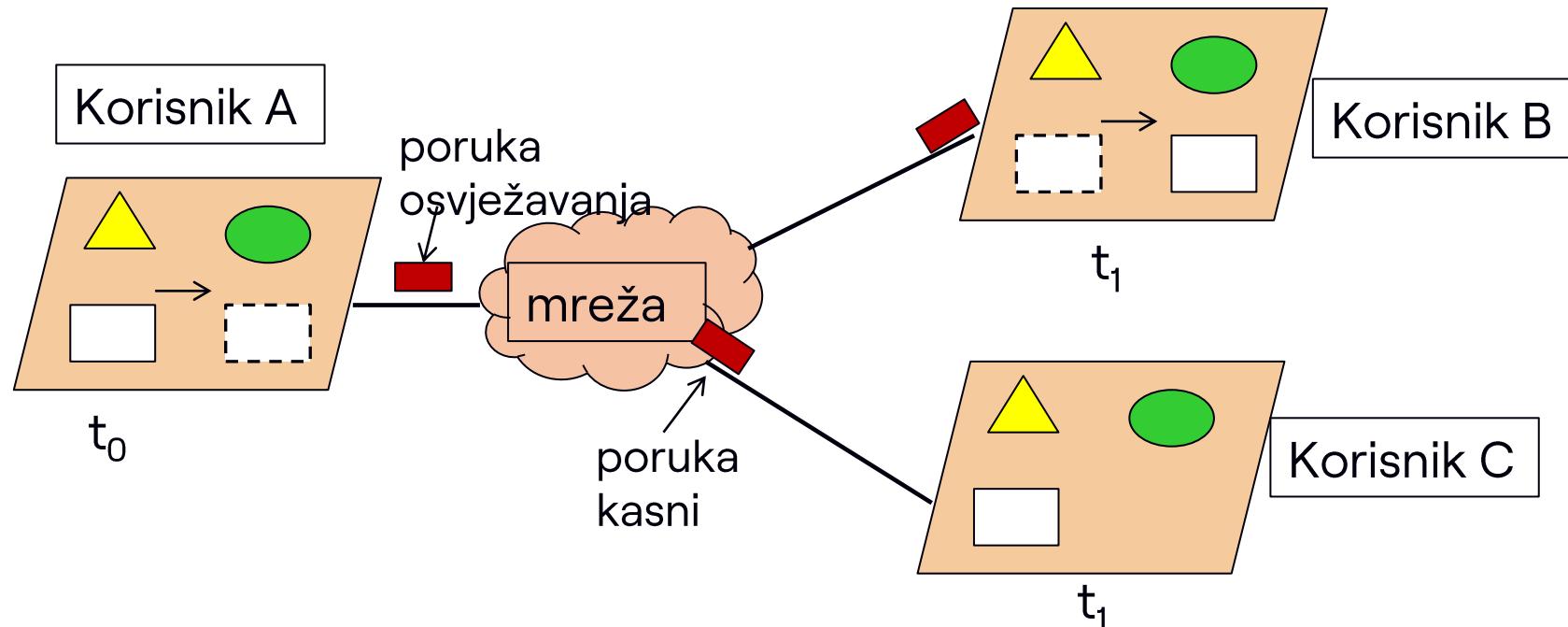
Zajedničko dinamičko stanje

- Svaki objekt koji se nalazi u umreženoj videoigri može se opisati odgovarajućim podacima
 - kako objekt izgleda
 - koja mu je lokacija u prostoru
 - koja mu je orientacija u prostoru
 - koja mu je brzinu ili ubrzanje...
- Kako su korisnici umrežene videoigre locirani na udaljenim lokacijama, svaka promjena u instanci virtualnog svijeta jednog korisnika mora se prenijeti drugim korisnicima putem mreže
- **Sve promjenjive informacije koje se izmjene u jednoj instanci virtualnog svijeta videoigre moraju se dostaviti drugim igračima koji sudjeluju u toj videoigri kako bi oni osvježili stanje svoje instance virtualnog svijeta videoigre**
- **Zajedničko dinamičko stanje umrežene videoigre** čini skup varijabli stanja svih pojedinačnih objekata koji se u videoigri mogu mijenjati, odnosno skup svih promjenjivih informacija.
- Na našem primjeru videoigre s nošenjem čaša vode to bi bili podaci o avataru svakog igrača, podaci o čašama svakog igrača i podaci o katnici svakog igrača. Podaci o stolu i slavinama nisu dio zajedničkog dinamičkog stanja jer se ne mijenjaju ni na koji način, stol je uvijek na istom mjestu i isto izgleda, a iz slavina konstantno teče voda

Promjene zajedničkog dinamičkog stanja

- Promjene na zajedničkom dinamičkom stanju nastaju **asinkrono** (neovisno jedna o drugoj), **dinamički** (nisu fiksirane u vremenu) i **raspodijeljeno** (nastaju na različitiminstancama virtualnog svijeta videoigre).
- Idealno, svi bi sudionici videoigre trebali imati **istu kopiju zajedničkog dinamičkog stanja**
- Zbog umreženosti i ograničenja na brzinama procesiranja i širenja signala ovo nije moguće uvijek ostvariti
- **Konzistentnost zajedničkog dinamičkog stanja** možemo definirati kao stupanj uskladenosti svih instanci virtualnog svijeta Videoigre
- Konzistentnost se procjenjuje objektivnim i subjektivnim metrikama
 - Iznos mrežnog kašnjenja – objektivna metrika
 - Koliko kašnjenje utječe na kvalitetu igre – subjektivna metrika

Problem nekonzistentnosti



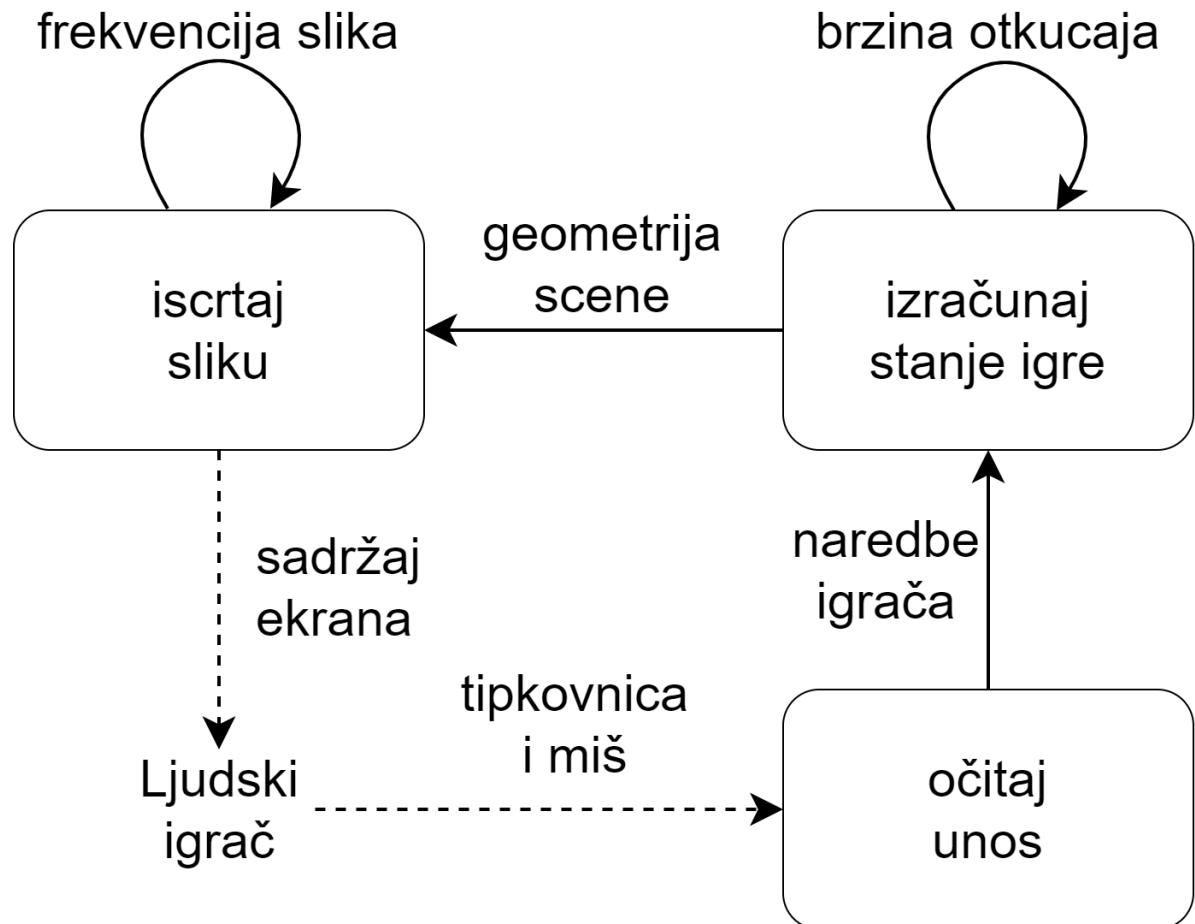
- Ključni izazov: održavanje raspodijeljenog zajedničkog stanja konzistentnim, ili približno konzistentnim, uz promjene koje nastaju (asinkrono, dinamički i raspodijeljeno)
- Zahtjeve na konzistentnost treba uskladiti s namjenom aplikacije
- Češće slanje poruka osvježavanja skraćuje potencijalni vremenski interval u kojem može doći do odstupanja, ali više opterećuje mrežu i proces za slanje i primanje poruka

Terminologija

- Zajedničko stanje se obnavlja u diskretnim vremenskim trenutcima unutar simulacije. Takvi trenutci se nazivaju **otkucaji** (engl. **tick**) dok se broj otkucaja u sekundi naziva **brzina otkucaja** (engl. **tickrate**)
 - Kod arhitekture ravnopravnih procesa najčešće se zajedničko stanje izračunava prilikom izračuna svakog okvira slike (engl. frame), a tickrate i framerate su onda isti
 - Kod arhitekture klijent – poslužitelj tickrate i framerate ne moraju biti istovjetni

Osnovna petlja u igri

- Primjer osnovne petlje u igri je prikazan na slici
- Petlja je primjenjiva i za umrežene i igre na jednom računalu
- U slučaju da je izračun stanja igre na udaljenom računalu imamo umreženu igru

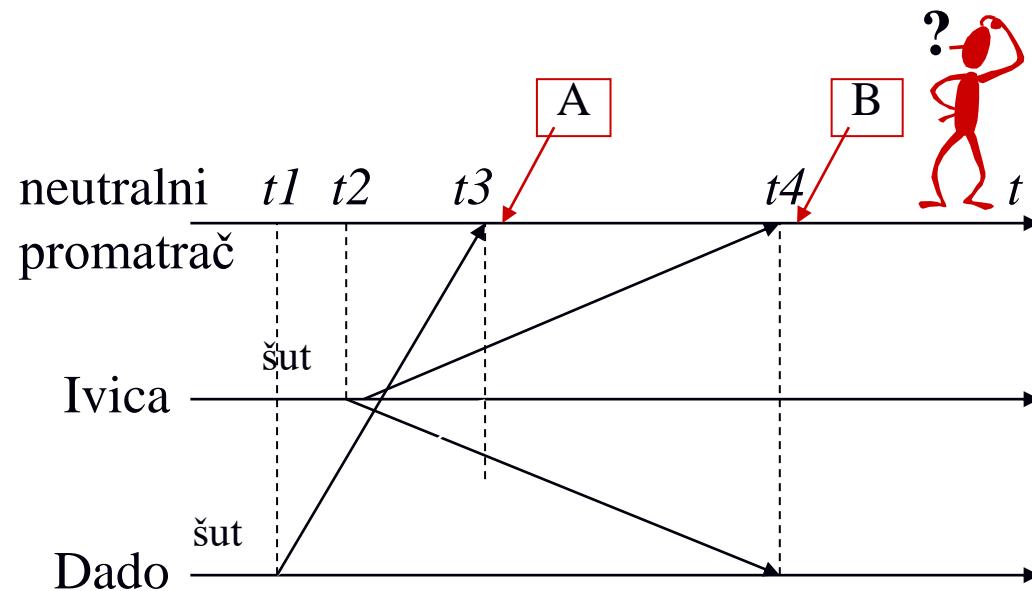
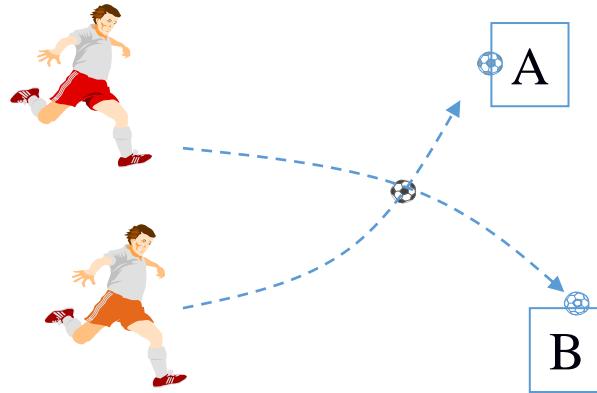


Autoritet i vlasništvo

- Kada nema potpune konzistentnosti dolazi do problema vlasništva nad objektima
- **Vlasništvo** nad objektom u videoigri označava mogućnost mijenjanja stanja tog objekta
- Treba spriječiti da više igrača istovremeno mijenja stanje entiteta [primjer 2, primjer 3]
 - Na primjer, promjena položaja je pisanje varijable stanja položaja entiteta (x, y, z)
- Uvodi se eksplicitno vlasništvo nad entitetom
 - Tipičan primjer je korisnikov *avatar*, čiji je vlasnik korisnik koji njime upravlja
 - Drugim objektima upravlja poslužitelj *autoritet*
- **Autoritet** označava pravo na donošenje konačne odluku oko promjene stanja nekog objekta u videoigri te osigurava osigurava da svaki entitet u zadanim trenutku ima samo jednog vlasnika

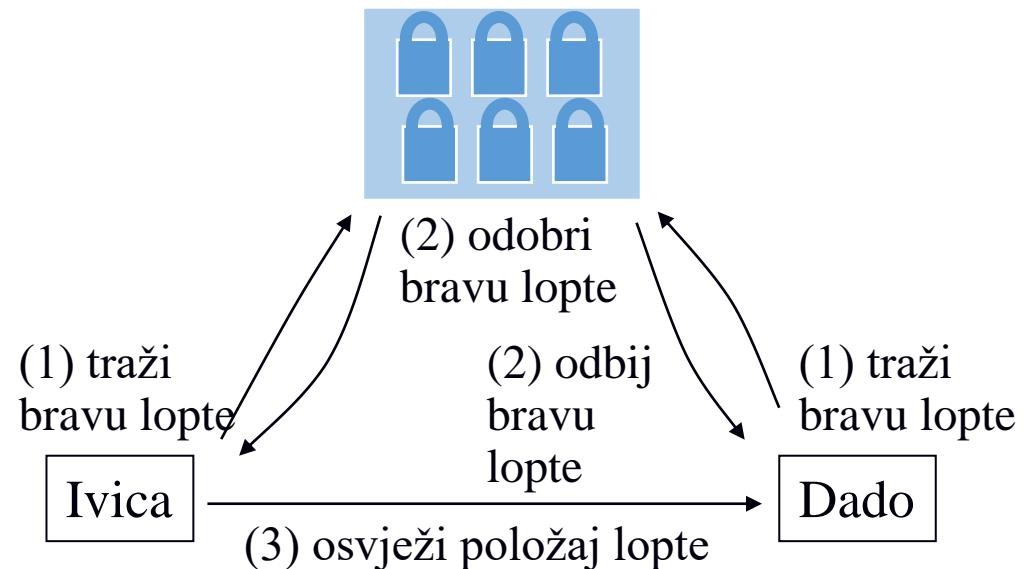
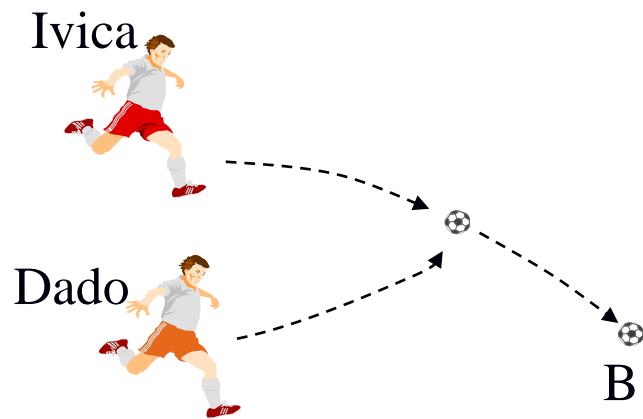
Primjer: nema vlasništva i autoriteta

Ivica i Dado igraju nogomet i pokušavaju istovremeni šutnuti loptu. Svaki hošt šalje poruku osvježavanja stanja lopte (položaj). Neki promatrači mogu prvo primiti Ivičinu poruku, a drugi Dadinu. Osim ako se Ivica i Dado slože oko toga tko je zapravo sutnuo loptu, svaki će nastavljati osvježavati položaj neovisno o drugome. Promatrači će vidjeti titranje lopte između novih položaja ovisno o pristizanju poruka osvježavanja od Ivice i Dade.



Primjer: ima vlasništva i autoriteta

- Ivica i Dado izdaju zahtjev za vlasništvom poslužitelju *lock manager* prije osvježavanja stanja lopte. Ovisno o politici upravljanja bravama (npr. FIFO, round-robin, prioriteti, itd.), *lock manager* daje bravu jednom od korisnika, u ovom primjeru, Ivici. Ivica onda može mijenjati položaj lopte dok god je u vlasništvu brave.



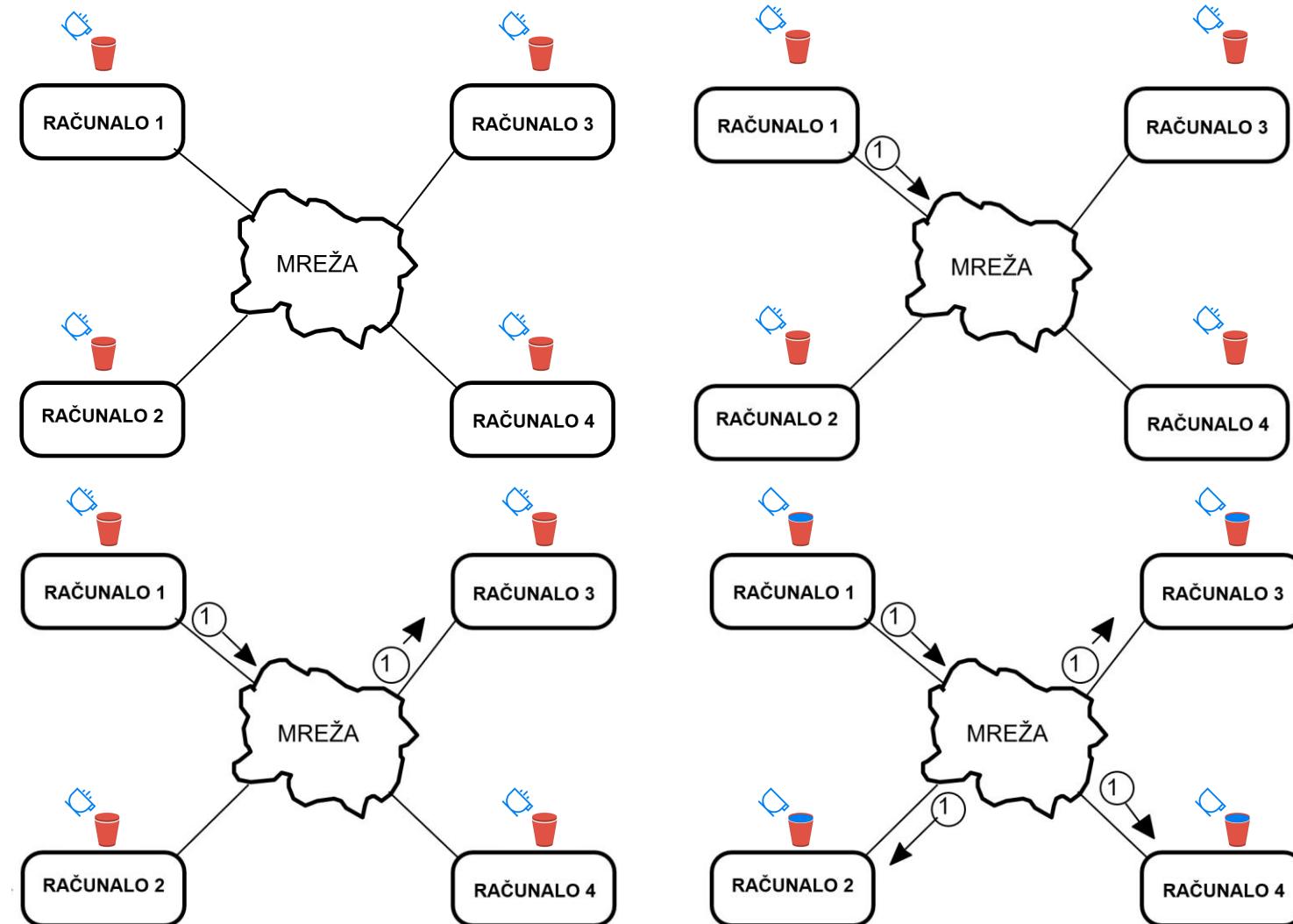
Postupci za održavanje konzistentnosti

Postupci za održavanje konzistentnosti

- Postupci za održavanje konzistentnosti mogu se podijeliti u dvije velike skupine
 - Jaka konzistentnost.
 - Slaba konzistentnost.
- **Jaka konzistentnost** podrazumijeva da je stanje virtualnog svijeta isto u svakom trenutku stanje u svakoj od instanci virtualnog svijeta na računalima svakog od prostorno distribuiranih korisnika.
- **Slaba konzistentnost** podrazumijeva da stanje virtualnog svijeta nije isto u svakom trenutku stanje u svakoj od instanci virtualnog svijeta na računalima svakog od prostorno distribuiranih korisnika.
- Zašto bi ikad htjeli slabu konzistentnost?
 - Jednostavno jer potpuna konzistentnost zahtjeva da informacija prvo dođe do svih u mreži i tada se u sviminstancama dešava promjena
 - To znači da je brzina promjena uvjetovana najsporijom vezom od svih sudionika u virtualnom svijetu što za interaktivnost određenih tipova igara nije dovoljno

Jaka konzistentnost

- Kada se desi promjena prvo se ista dostavlja svim sudionicima u simulaciji
- Tek nakon što je promjena dostavljena svima izvršava se na svim sudionicima (pa i na onom na kojem je nastala)
- Rezultira u kašnjenju pri unosu komandi (engl. input delay)

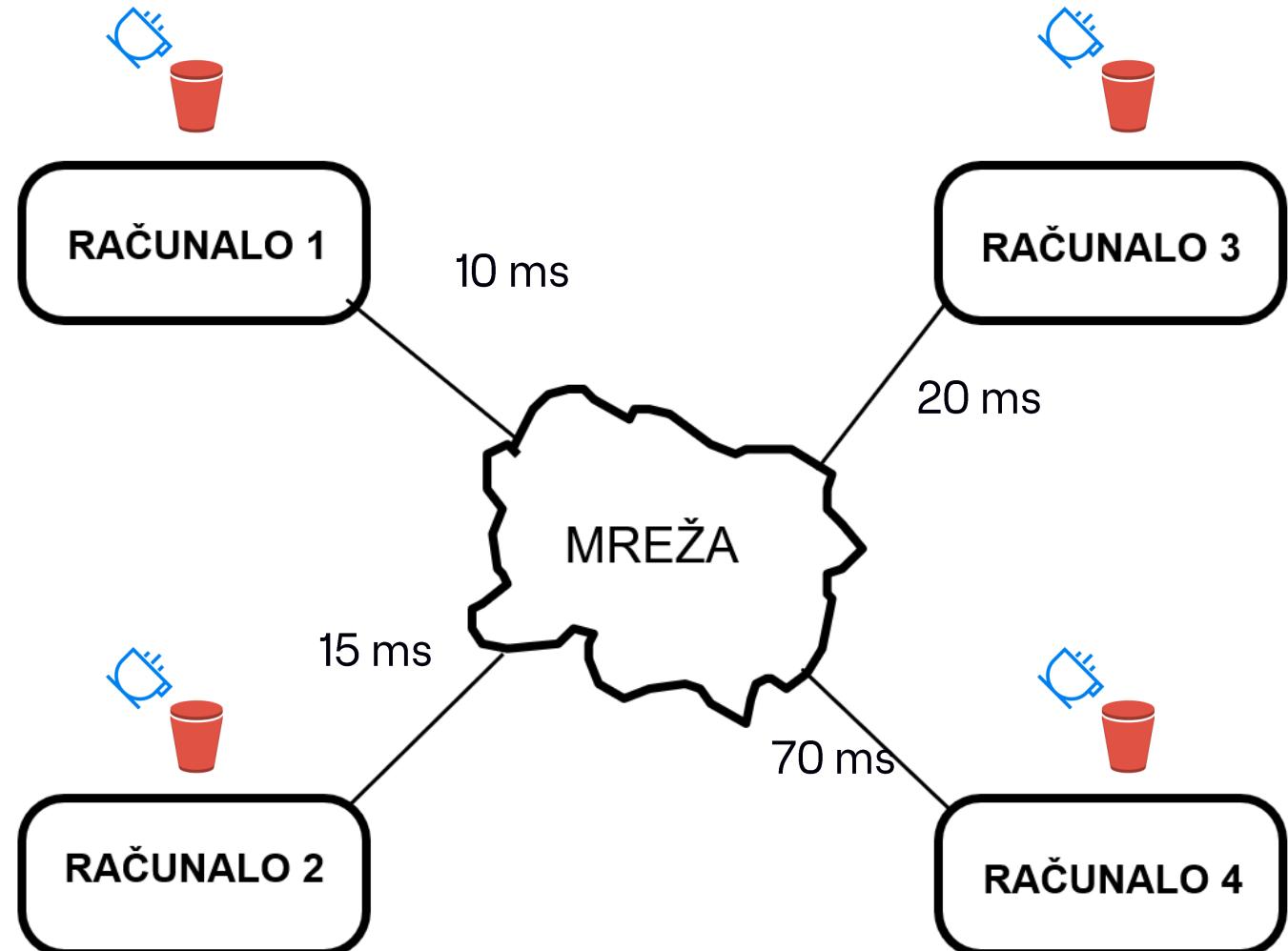


Distribuirana deterministička simulacija

- Implementacija jake konzistentnosti
- U distribuiranoj determinističkoj simulaciji
 - Svaki od klijenata skuplja komande korisnika kroz neki vremenski period te ih onda šalje ostalim korisnicima u virtualnom svijetu
 - Postoji međuspremnik za izvršavanje komandi (engl. playout buffer)
 - Određivanje vremenskog okvira u kojem se čeka u međuspremniku složen postupak
 - Najsporiji mrežni sudionik usporava cijelu simulaciju
 - Svaki klijent posjeduje ukupno stanje virtualnog svijeta te sve podatke
 - Međusobno svaki od klijenata izmjenjuju samo komande dane u videoigri
 - Deterministička simulacija označava da isti unos uz isto stanje uvijek daje iste rezultate (ne približne)

Primjer

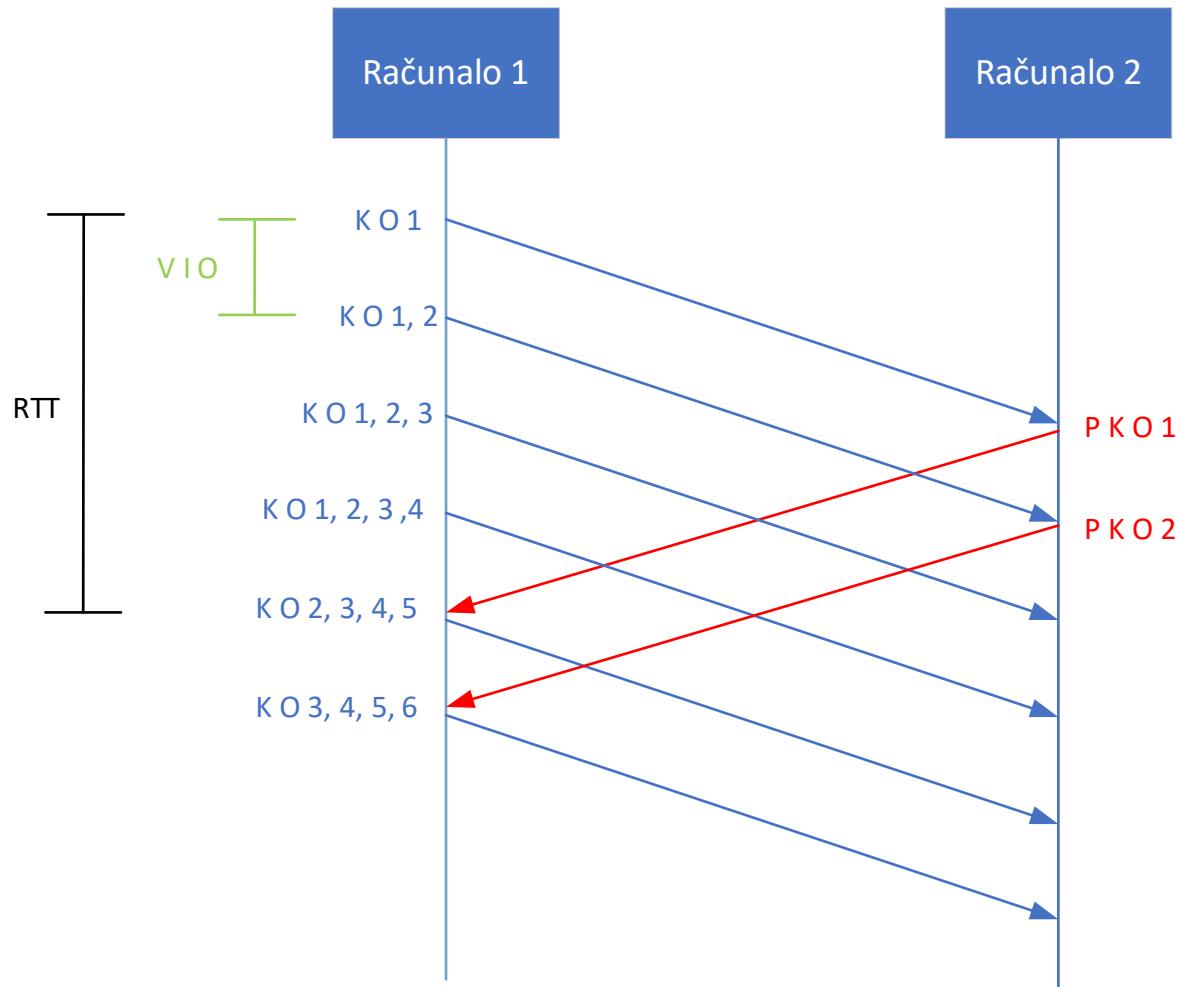
- Brzina osvježavanja okvira slike (engl. frame rate) je 60 okvira po sekundi
- Igrači imaju kašnjenja kao na slici (u jednom smjeru)
- Koliko okvira minimalno mora kasniti svaka simulacija samo da bi kompenzirala mrežno kašnjenje?
- Maksimalno kašnjenje koje je potrebno da dodu podaci do drugog računala je 90 ms (računalo 3 do računala 4)
- Broj okvira u 90 ms izračunamo iz proporcije:
$$1000 \text{ ms} : 60 = 90 \text{ ms} : x$$
$$x = 90 * 60 / 1000$$
$$x = 5,4 \text{ odnosno } 6 \text{ okvira}$$



Utjecaj mrežnog protokola

- Deterministička simulacija traži da svaka komanda dostavi do odredišta – koji protokol?
- Prvi odgovor je TCP jer osigurava dostavu informacije
- Što ako imamo gubitak paketa?
 - Paketi se ponovno trebaju poslati
 - Kašnjenje ukupno
 - Vrijeme detekcije gubitka (RTO ili vrijeme do tri duple potvrde)
 - Mrežno kašnjenje
- Što ako je ukupno kašnjenje veće od definiranog u playout bufferu?
 - Desinkronizacija – različite inačice virtualnog svijeta su dobole različite ulazne podatke?
 - Dinamičko povećavanje vremena čekanja dok ne dođu informacije od svakog sudionika – povećava se kašnjenje te dolazi do artefakata u izvođenju igre (zastajkivanje i potom ubrzano dolaženje na stvarno stanje)
 - Ignoriranje inputa?
- UDP uz osiguranje dostave svih podataka kroz protokol aplikacijske razine možda bolje rješenje
- Svaki UDP paket nosi sve podatke koji do tada nisu potvrđeni kao primljeni

Deterministička simulacija na protokolu UDP



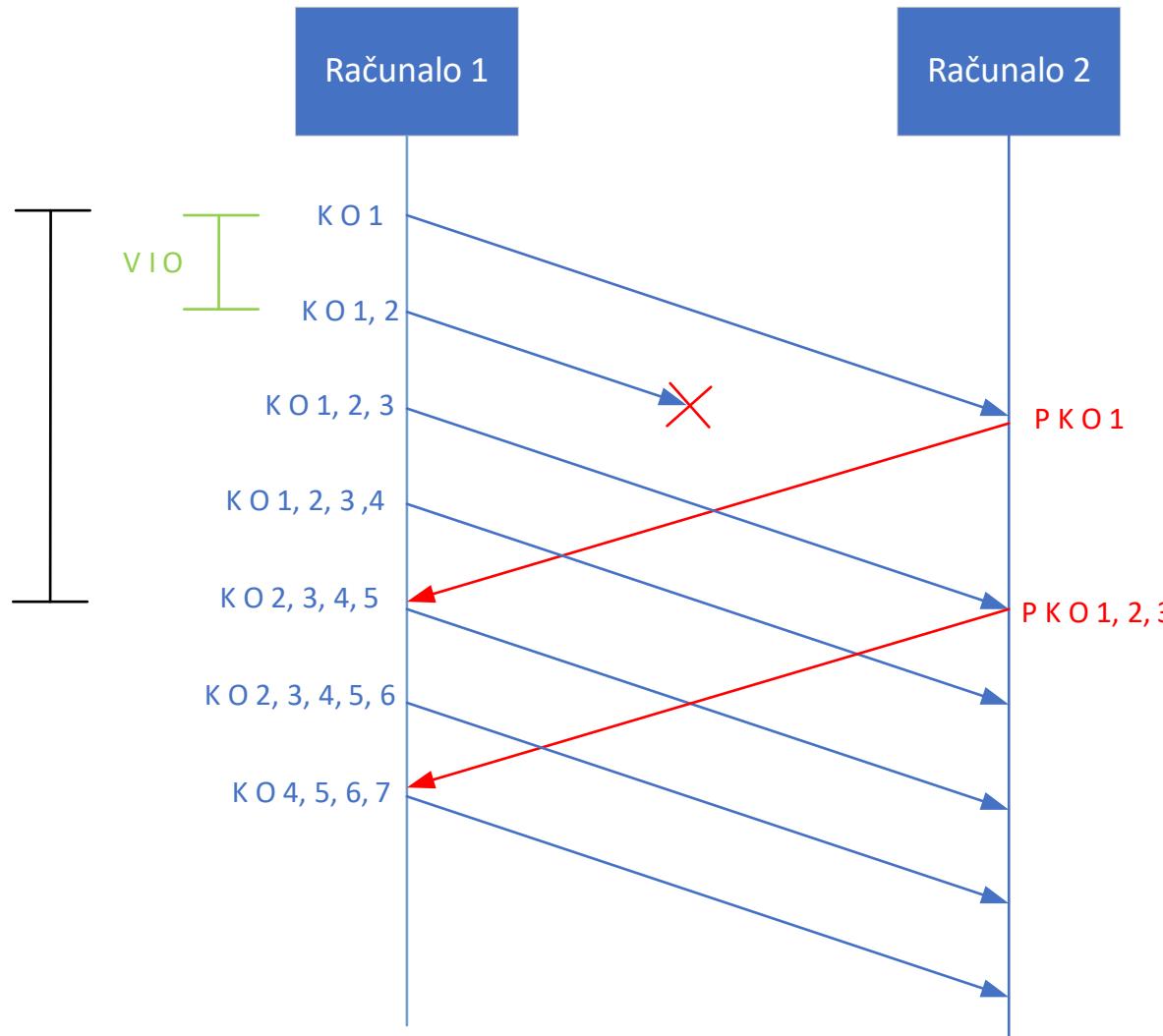
K O n – Komande Okvira n
P K O n – Potvrda Komande Okvira n
VIO – Vrijeme između okvira
RTT - Mrežno Kašnjenje od jednog računala do drugog i nazad (engl. Round Trip Time)

Unos iz koliko okvira se prenosi po jednom paketu?

- Ako RTT iznosi 160 ms, brzina okvira je 30 okvira po sekundi, a komande unutar jednog okvira su veličine 10 okteta, kako su distribuirane veličine korisne informacije u jednom paketu?
 - Vrijeme između okvira je $1000 \text{ ms} / 30 = 33,33 \text{ ms}$
 - Broj okvira koji se šalje u 160 ms = 4,8
 - 80% su paketi su veličine $5 * 10 \text{ okteta} = 50 \text{ okteta}$
 - 20% su paketi veličine $4 * 10 \text{ okteta} = 40 \text{ okteta}$
 - Niti jedan nije paket veličine 48 okteta (distribucija)

Što kad se izgubi paket?

- Nova informacija dolazi već nakon jednog VIO
- Jednostavnim aplikacijskim protokolom riješen problem
- Različita distribucija veličina poslanih paketa



K O n – Komande Okvira n

P K O n – Potvrda Komande Okvira n

V IO – Vrijeme između okvira

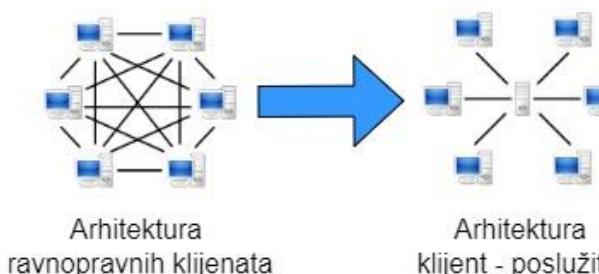
RTT - Mrežno Kašnjenje od jednog računala do drugog i nazad
(engl. Round Trip Time)

Deterministička simulacija – prednosti i mane

- Najveće prednosti ovog pristupa su:
 - Svi igrači u svakom trenutku vide isti virtualni svijet.
 - Mala količina razmijenjenih podataka unatoč velikom broju kontroliranih entiteta u virtualnom svijetu.
- Najveće mane ovog pristupa su:
 - Odgođeno izvođenje komandi
 - Nemogućnost primjene na igre koje ovise o jakom brzim interakcijama te su vrlo dinamične, primjerice igre gađanja iz prvog lica (FPS).
- Česta primjena u strategijskim igramama (RTS) ili akcijskim strategijskim igramama (ARTS) poput Dota 2, Starcraft 2 itd.



StarCraft (1998. - 2010.)



StarCraft II (2010. - danas)

Razvojna podrška za determinističku simulaciju u Unity pogonskom sustavu

- <https://www.photonengine.com/quantum>
- Nije još besplatan, ali uskoro bi trebao biti
- Imamo SDK koji nam je ustupio Photon
 - razvoj sljedećeg labosa na Quantumu
 - prilika za dodatne bodove



Razvojna podrška za determinističku simulaciju u Unity pogonskom sustavu

Quantum Disrupts Multiplayer Development

Deterministic Game Engine

Physics/Math, Pathfinding/Steering, ECS, Job System, ...

Zero-Lag and E-Sport Grade

Predict/Rollback, Replays, Spectating

Cheat Protection

State manipulation impossible (Input exchange and checksum), Server Referee Simulation

No Netcode

Same Code for Single-, Local- and Multiplayer

Low Bandwidth and High Freq.

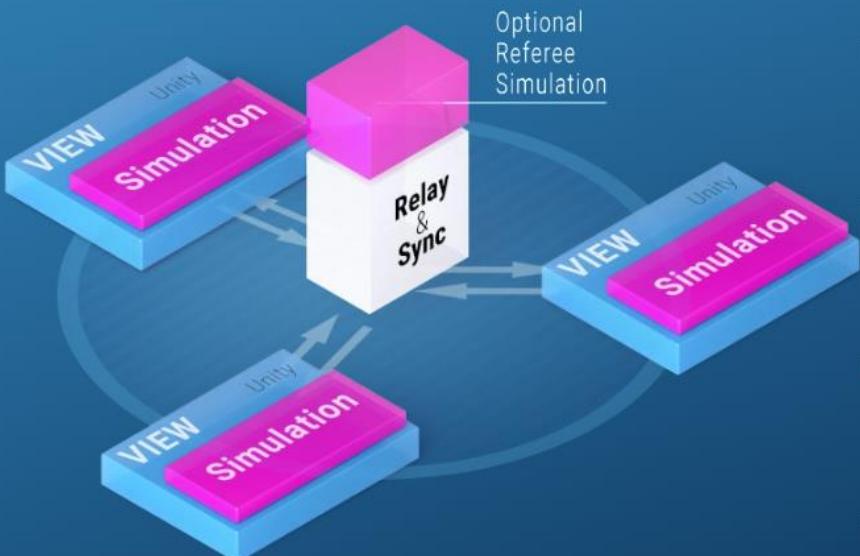
Only Input is Exchanged at 30/60Hz

Blazing Fast Performance

Zero Runtime Allocations, Memory Aligned Data, Blittable Game State, Custom IL Emitter ...

Razvojna podrška za determinističku simulaciju u Unity pogonskom sustavu

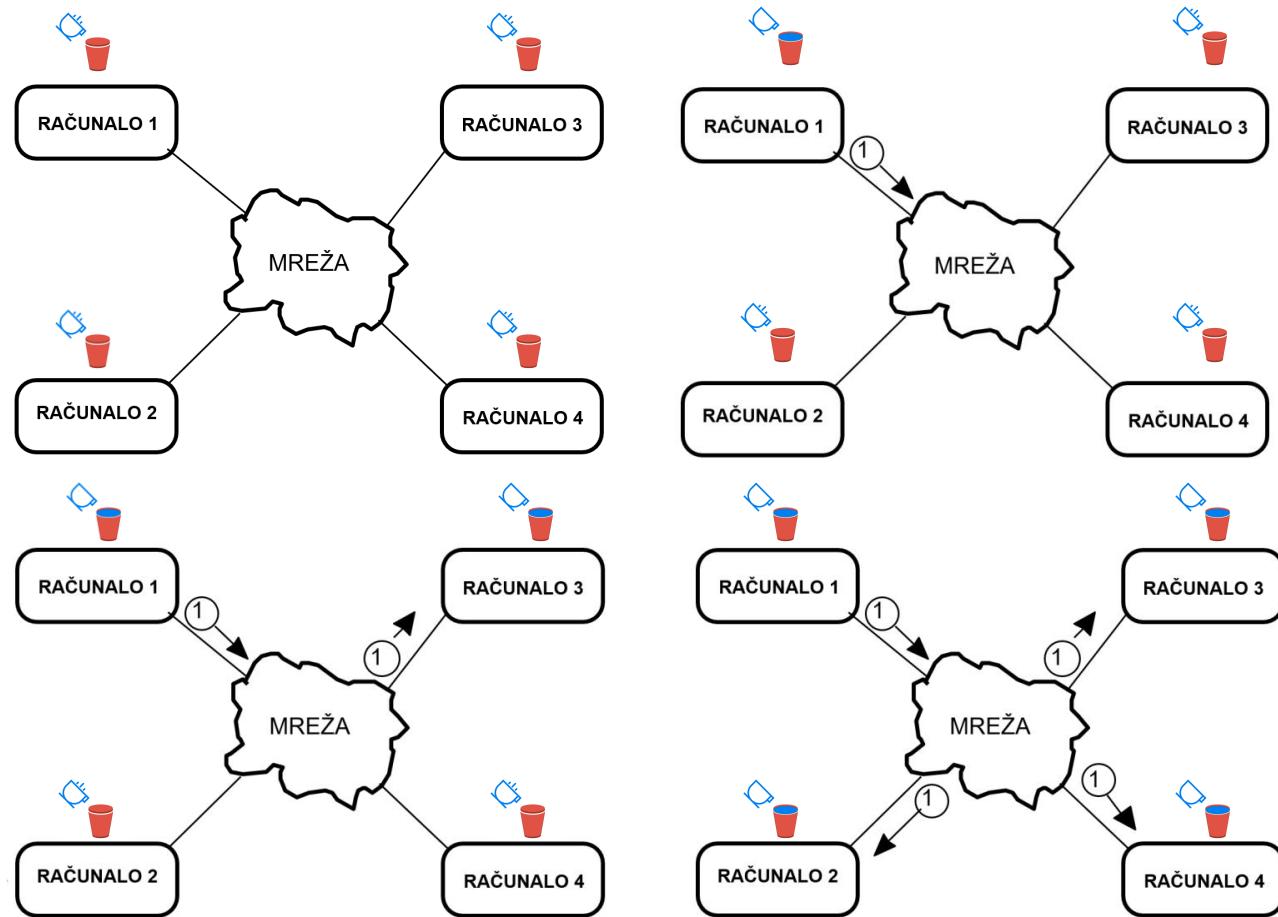
Only Input is **Exchanged**



- Relay **exchanges input** to ALL players
- Magic: **Invalidate** input for slow clients (e.g. 200ms)
- Simulation has no dependency on Unity and can **run anywhere**
- **Desync detection** via checksums (checksum debugger)
- Custom **hyper fast UDP** protocol (reliable, unsequenced)

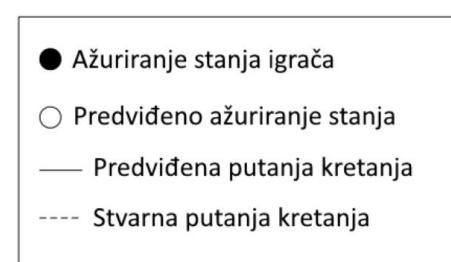
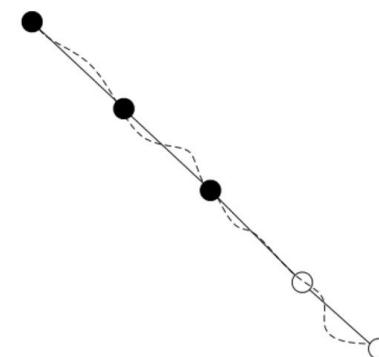
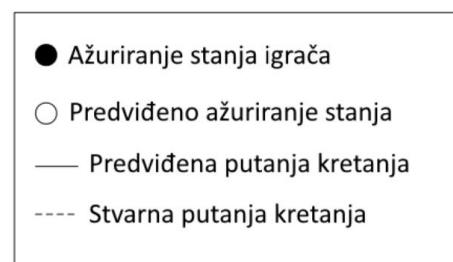
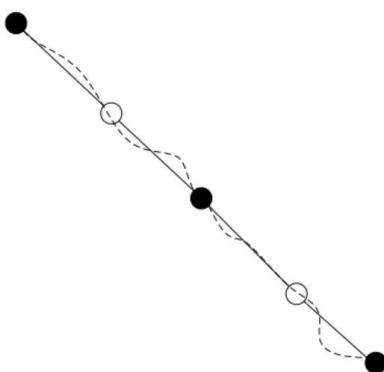
Slaba konzistentnost

- Kada se desi promjena ista pošalje svim sudionicima u simulaciji
- Svaki sudionik u simulaciji izvršava promjenu čim je dobije
- Rezultira u nekonzistentnosti kod različitih sudionika, odnosno različitim stanjima u isto vrijeme



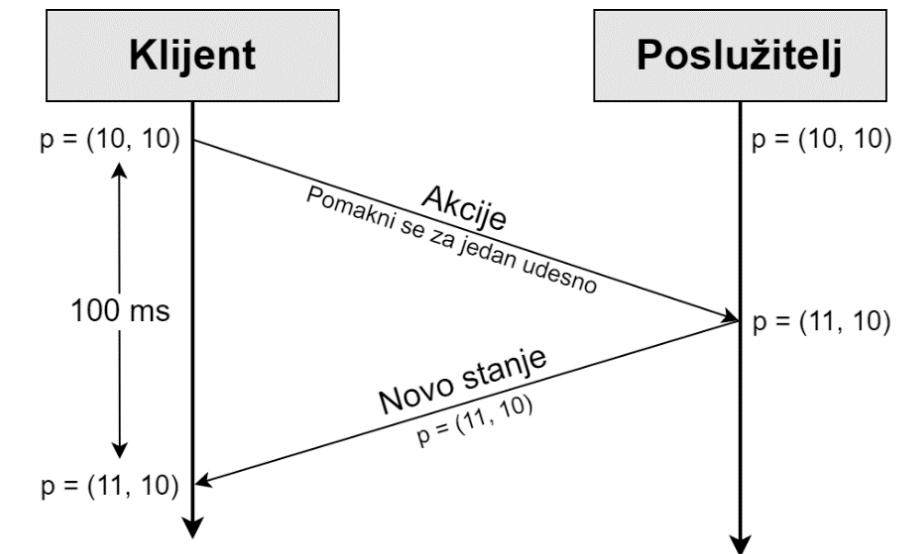
Interpolacija i ekstrapolacija

- **Interpolacija** označava proces u kojem se na temelju dvije poznate vrijednosti određene varijable izračunavaju među vrijednosti, odnosno omogućuje postupan prijelaz između tih vrijednosti,
- **Ekstrapolacija** označava proces procjene buduće vrijednosti određene varijable na temelju prethodnih vrijednosti.



Sinkronizacija stanja

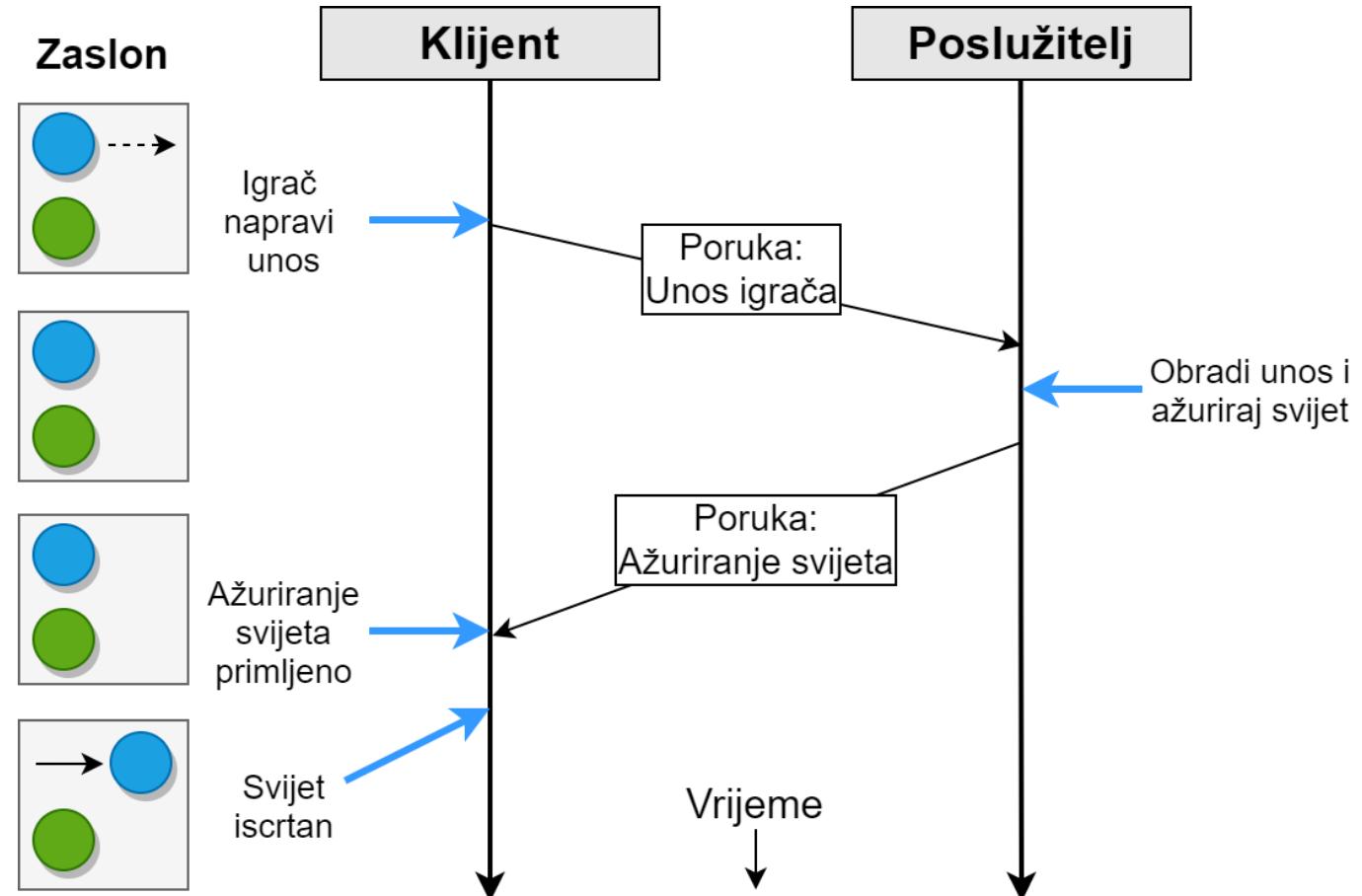
- Implementacija slabe konzistentnosti
- Kod sinkronizacije stanja
 - Na svakom klijentu postoji instanca simulacije virtualnog svijeta.
 - Klijenti šalju svoje komande i nova stanja prema poslužitelju, a poslužitelj ih provjerava i odobrava te izračunava interakcije s drugim klijentima te distribuira nova odobrena stanja svim korisnicima
 - Može biti napravljena i na arhitekturi ravноправних процеса
- Sinkronizacija stanja je najčešći pristup rješavanju problema konzistentnosti u igrama danas



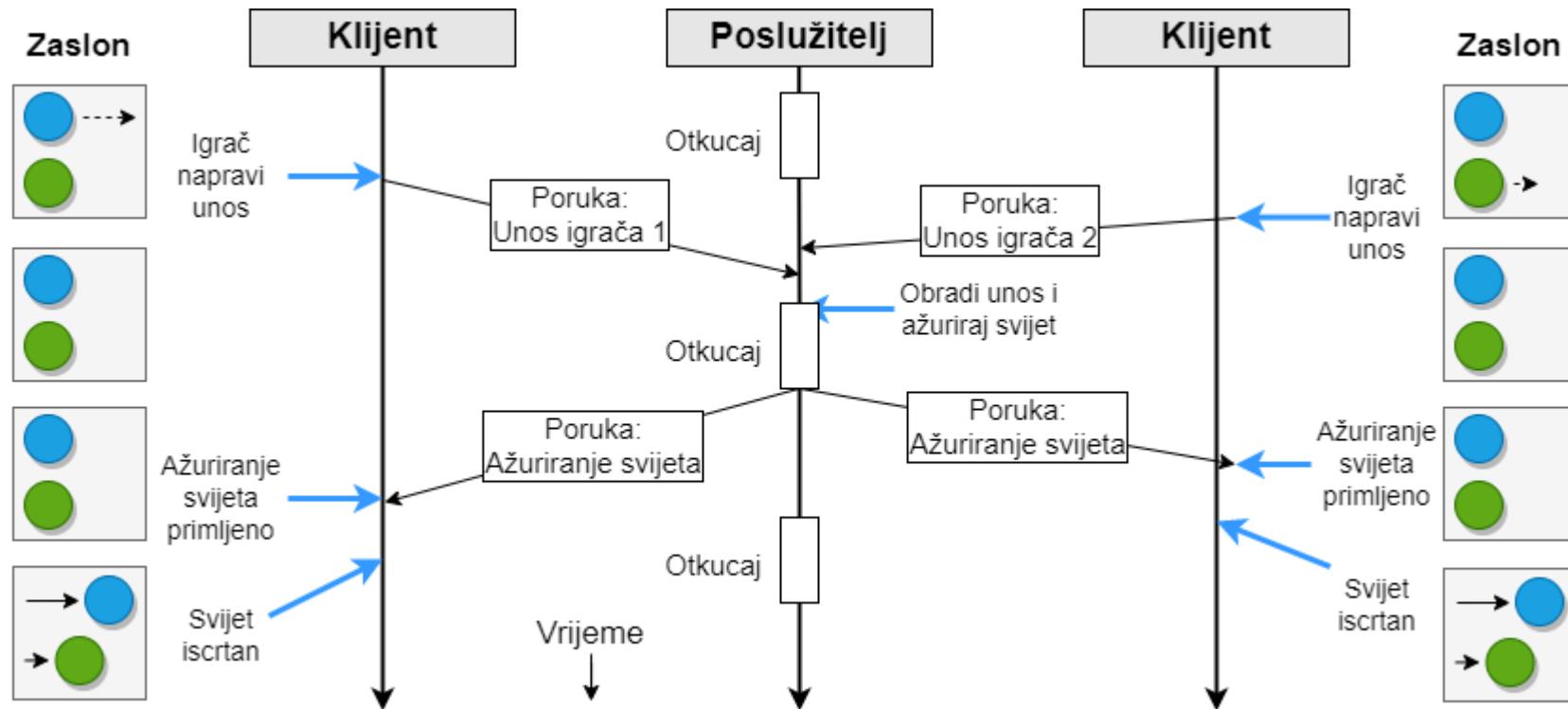
Karakteristike sinkronizacije stanja

- Pošto se šalje i komande i stanje s klijenta – ne moraju svi sudionici biti u potpunosti sinkronizirani
- Predikcija na strani klijenta – klijent može odmah izvršiti svoju komandu ne čekajući na osvježenje stanja od strane poslužitelja (ako dođe do razlike mora se na neki način razlika riješiti)
- Kad se šalju osvježenja može se slati samo promjena, odnosno samo oni objekti koji su se promijenili
- Otpornost na gubitke paketa – simulacija se može nastaviti i ako ne dolaze redovno osvježenja s druge strane

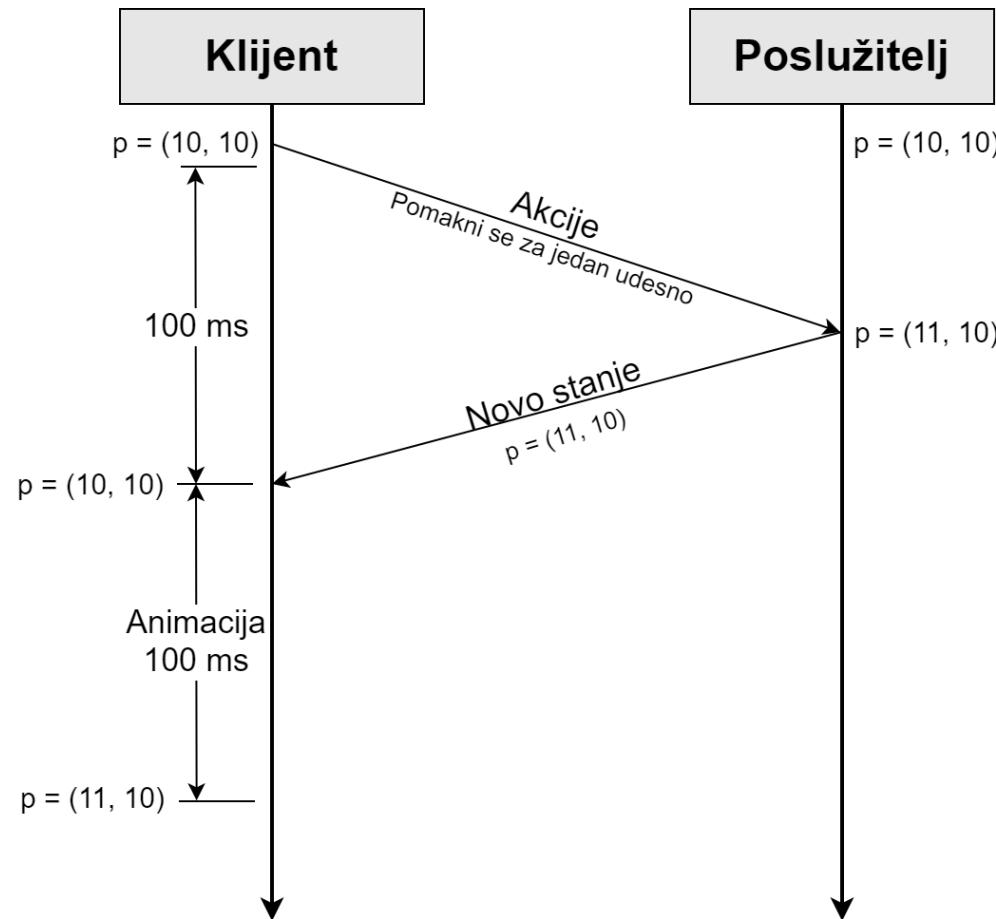
Sinkronizacija stanja



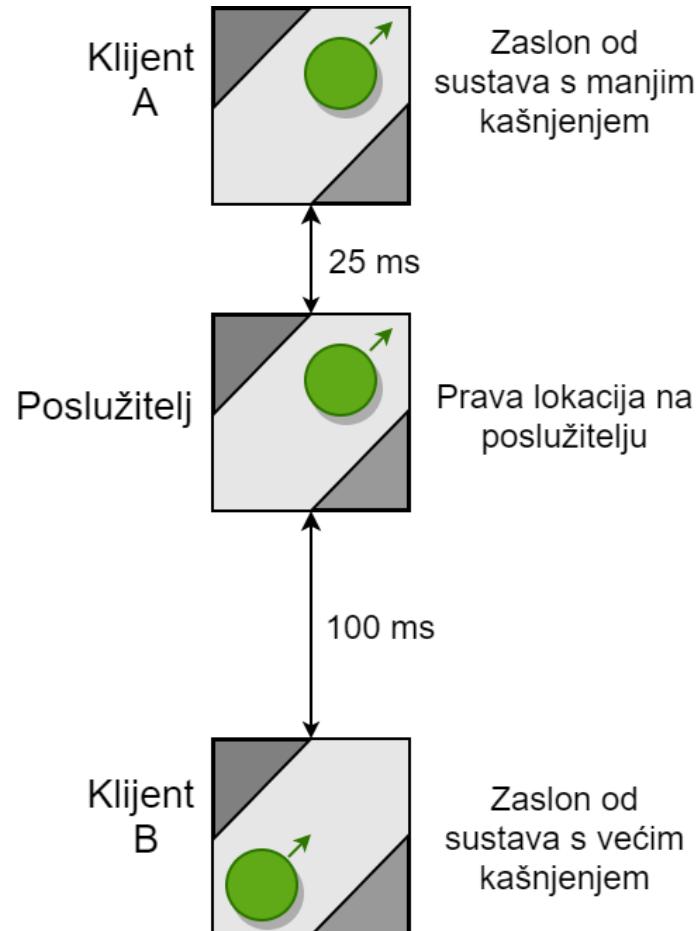
Sinkronizacija stanja – dva igrača



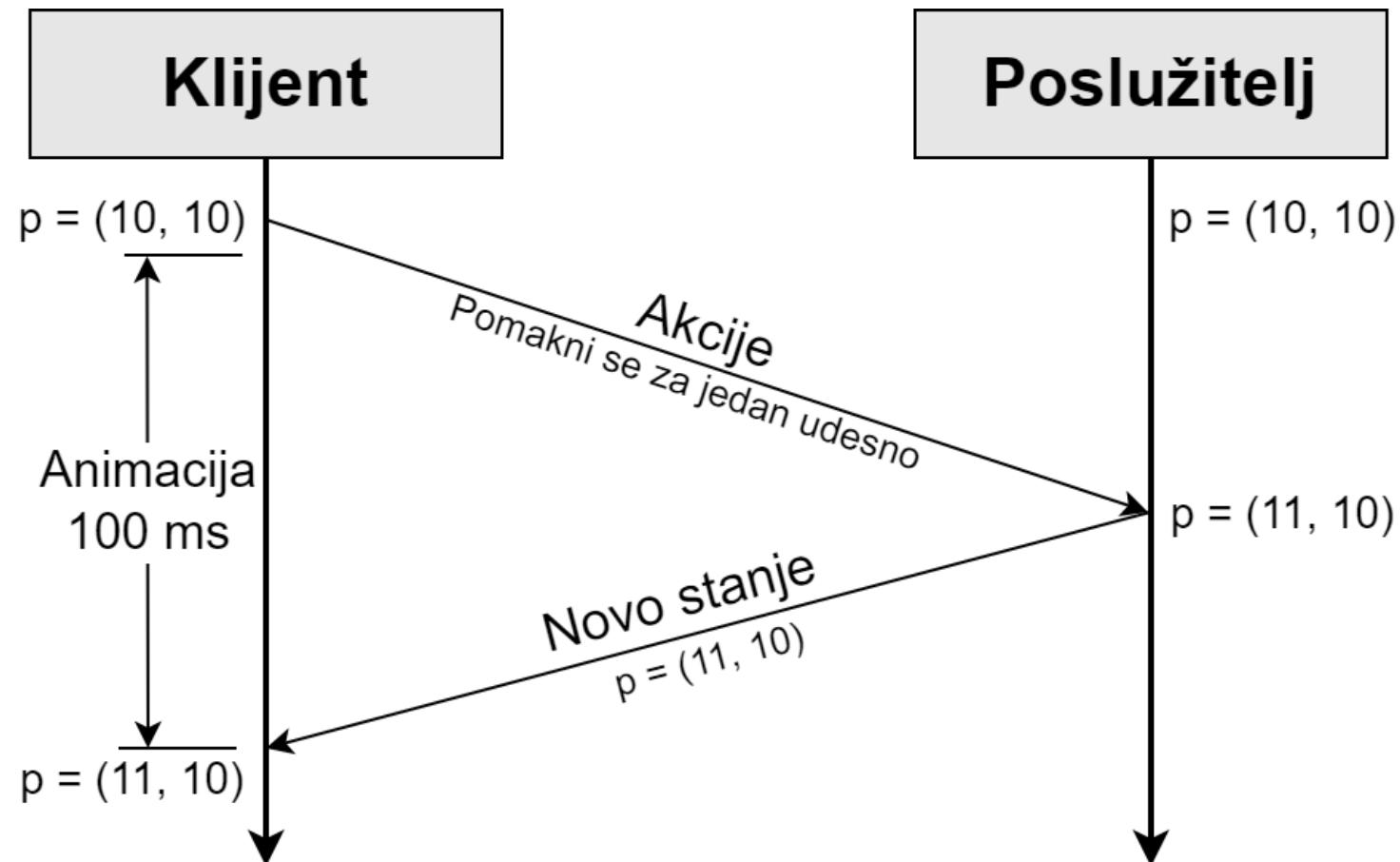
Sinkronizacija stanja – kašnjenje zbog autorizacije



Sinkronizacija stanja – razlika kašnjenja



Sinkronizacija stanja - predikcija

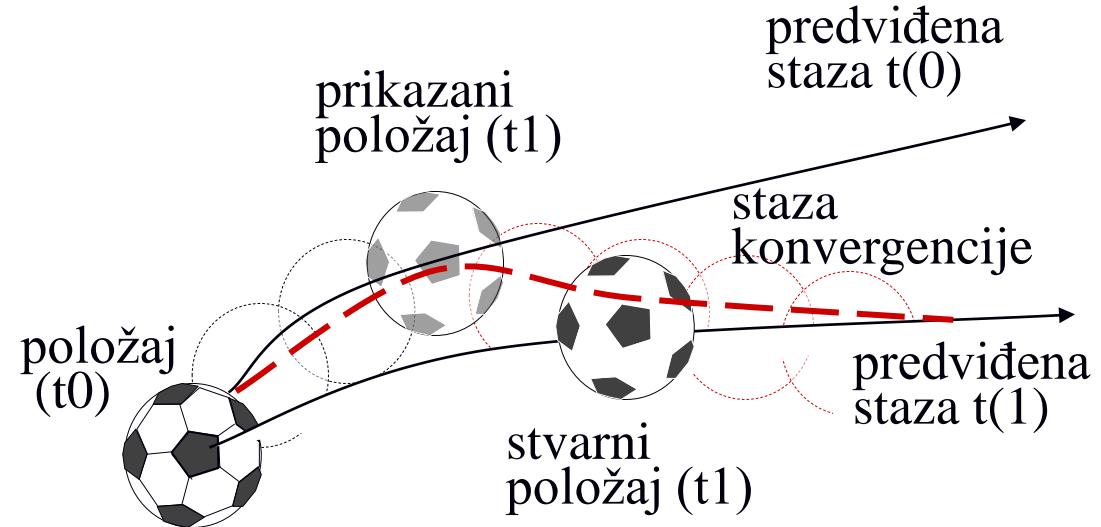


Što ako predikcija nije u redu?

- Ako klijent napravi predikciju stanja, a koja nije u skladu s novi izračunatim stanjem na poslužitelju dolazi do razlike u stanju koja se mora izmiriti
 - primjerice klijent je rekao da se pomjera ulijevo, ali neka efekt od drugog igrača ili kompjuterski kontroliranog lika na poslužitelju ga gurne u desno ili mu zaprijeći kretanje u lijevo
- **Izmirivanje stanja** je proces u kojem stanje na klijentu mora biti dovedeno u stvarno stanje na temelju osvježenja dobivenog od poslužitelja
 - **Postupno izmirivanje (konvergencija)** (engl. rubber banding) – je pristup u kome se poslužiteljska i klijentska simulacija je da se postupno dovedu u sinkronizirano stanje
 - **Instantno izmirivanje** (engl. snapping, teleporting) – je pristup u kojem se u idućem okviru klijentska simulacija odmah prilagodi poslužiteljskoj
- Postupno izmirivanje se provodi tako da se izračuna na temelju sadašnjeg poznatog stanja nekakvo buduće stanje te da se radi konvergencija iz sadašnjeg predviđenog stanja u buduće stanje kroz određenu krivulju
- U biti ta konvergencija je napravljena kroz proces ekstrapolacije i interpolacije..

Ekstrapolacija – Dead reckoning

- Sastoji se od dva osnovna dijela:
 - *Predikcija*: izračunavanje sadašnjeg stanja na temelju prethodno primljenih poruka osvježavanja
 - *Konvergencija*: korekcija staze dobivene predikcijom na temelju novo-primljenih poruka osvježavanja ("izglađivanje")
- Koristi se za entitete koje kontrolira udaljeni sudionik u simulaciji (primjerice poslužitelj)



Prednosti i mane

- Više o svim specifičnostima osvježenja stanja i tehnikama u kasnijim predavanjima
- Najveće prednosti ovog pristupa su:
 - Omogućuje najdinamičnije igre te kompletno sakrivanje mrežnog kašnjenja što se tiče akcija samog igrača.
 - Konceptualno jednostavne za implementirati.
- Najveće mane ovog pristupa su:
 - Visoka razina nekonzistentnosti ovisna o mrežnom kašnjenju koja može uzrokovati različite anomalije (primjerice „smrt iza zida“ kada se igrač sakrije iza zida, a drugi ga igrač pogodi kao da je na otvorenom).
 - Potreba za konvergencijom između stanja klijenta i poslužitelja u slučaju predviđanja na razini klijenta.
 - Lokalna simulacija omogućuje lakše varanje.

Reprodukcia snimaka

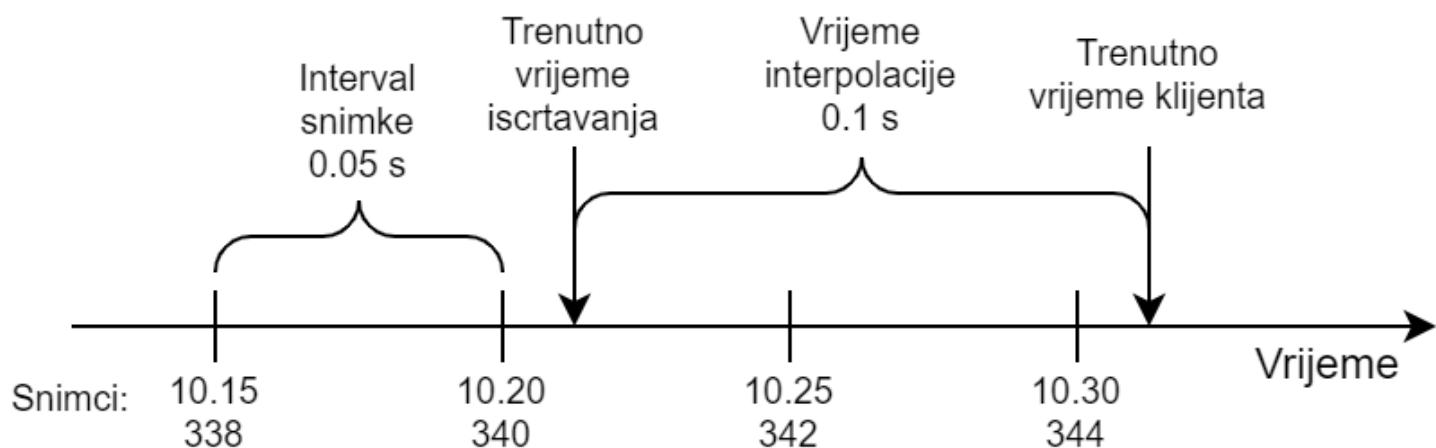
- Implementacija slabe konzistentnosti
- Reprodukcia snimaka je metoda u kojima se od poslužitelja klijentu šalju samo slike stanja virtualnog svijeta te klijent nema na svojoj samu simulaciju igre te iz danih snimaka klijent interpolira stanje svijeta za svaki okvir
- Snimak stanja cijelog svijeta može uzimati jako puno memorije jer su potrebni svi podaci da bi se napravila vizualna aproksimacija simulacije
- U laboratorijskoj vježbi svaka snimka je oko 25 000 okteta, s 28 oktetom za IP + UDP zaglavljem uz 60 okvira po sekundi dolazimo do oko 1 500 000 okteta po sekundi ~ 12 Mbita po sekundi
- Dodatno, ako šalju se paketi za svaki okvir može doći do gubitaka i promjene redoslijeda paketa
- Zbog toga je nerealistično slati snimke stanja za svaki okvir slike koja se računa

Interpolacija snimaka

- **Interpolacija** označava proces u kojem se na temelju dvije poznate vrijednosti određene varijable izračunavaju među vrijednosti, odnosno omogućuje postupan prijelaz između tih vrijednosti
- Ako ne prikažemo snimak odmah po primitku, već čekamo idući mogu se izračunati vrijednosti koje su između njih te tako možemo na račun dodatnog kašnjenja smanjiti opterećenje mreže, a i dobiti na smanjenju trzanja u animacijama
- Postavljajući interpolaciju na 10 snimaka po sekundi, smanjuje se brzina na oko 1,2 Mbit/s uz dodavanje 100ms kašnjenja
- Klijent mora imati barem dva snimka između kojih će interpolirati

Stvarni mrežni uvjeti

- Zašto u praksi nije dovoljno imati samo dva snimka?
- Gubitak paketa
- U slučaju da izgubimo paket u kojem dolazi novi snimak nećemo imati vrijednosti prema kojima ćemo izvršiti interpolaciju te će doći do zastajkivanja u animaciji
- U praksi se najčešće omogućuje da gubitak dva paketa zaredom ne dovodi do zastajkivanja simulacije, odnosno u našem slučaju spremnik bi bio 350 ms (dodatnih 50 ms zbog varijacije kašnjenja)
- Na slici je prikazan međuspremnik koji sadrži dva snimka u njemu
- Na ovom primjeru ako je izgubljena snimka 342 klijent može koristiti snimku 344



Primjer

- Izračunajmo vjerojatnost zastjkivanja u slučaju slanja snimaka frekvencijom 10 snimaka po sekundi, gubitkom paketa od 5% i veličinama međuspremnika od 150, 250 i 350 ms (1 paket, 2 paketa i 3 paketa u među spremniku) uz uvjet da je posljednji paket ispravno došao
- Ako imamo 1 paket u među spremniku taj mora doći ispravno da ne bi bilo zastjkivanja – šansa zastjkivanja je $1 - 0.95 = 0.05$
- Ako imamo 2 paketa u međuspremniku moraju oba ne doći da bi došlo do zastjkivanja – šansa zastjkivanja je $(0.05 * 0.05) = 0.0025$
- Ako imamo 3 paketa u međuspremniku moraju sva tri ne doći da bi došlo do zastjkivanja – šansa zastjkivanja je $(0.05 * 0.05 * 0.05) = 0.000125$

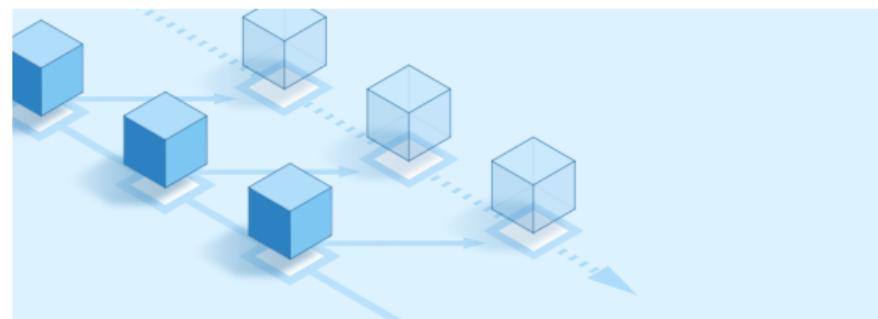
Prednosti i mane

- Najveće prednosti ovog pristupa su:
 - Onemogućuje varanje na strani klijenta.
 - Pojednostavljuje implementaciju klijenta.
- Najveće mane ovog pristupa su:
 - Dodaje određenu razinu kašnjenja čak i kod unosa komandi i kontrole igrača.
 - Nema potpune konzistentnosti.
- Najčešće se interpolacija snimaka ne koristi samostalno
- Koncept interpolacije snimaka se može koristiti i kao metoda unutar osvježenja stanja gdje se mogu interpolirati pojedini entiteti

Razvoj za osvježavanje stanja i interpolaciju snimana u Unity pogonskom sustavu

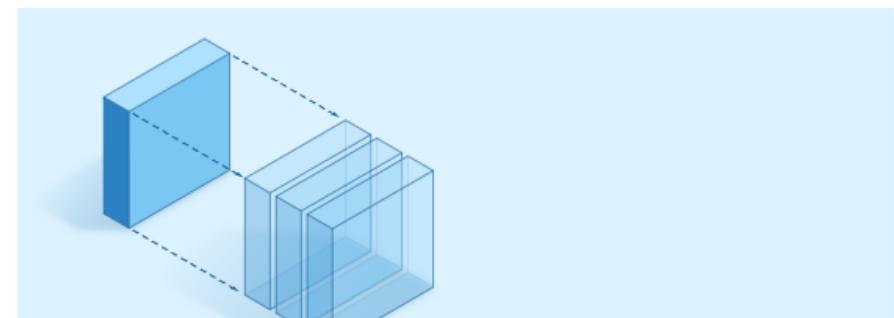
- <https://www.photonengine.com/en-US/fusion>
- Besplatan za korištenje

Powerful Features required to build the
Best Multiplayer Games:



Tick-based Simulation

Core feature of stable and accurate networking.
Foundation for client side prediction and snapshot interpolation.



Client-side Prediction

Give players an instant response to their own inputs without giving up server authority, even under high latency and network loss.

Razvoj za osvježavanje stanja i interpolaciju snimana u Unity pogonskom sustavu



Full Physics Prediction

Get the best experience out of complex physics interactions between players and objects by using Full Physics Prediction.



Snapshot Interpolation

Use automated or custom interpolation for smooth visual rendering even with bad internet conditions.



Lag Compensation

Built in Lag Compensated Hitboxes with easy to use API for Esport grade game mechanics.



Replication Systems

Best in class implementations of replication algorithms for Delta Snapshots and Eventual Consistency with Interest Management.

Zadatak za iduće predavanje

- Pročitajte dijelove Unity dokumentacije (sve iz Messaging System:
<https://docs-multiplayer.unity3d.com/netcode/current/advanced-topics/ways-synchronize>