Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Bartosz Borkowski
Nr albumu: 248235

Zastosowanie algorytmów plotkujących w grach komputerowych dla wielu graczy

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr Janiny Mincer-Daszkiewicz
Instytut Informatyki

Czerwiec 2011

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Streszczenie

W pracy przedstawiono konstrukcję i implementację protokołu sieciowego dla wieloosobowych gier komputerowych. Przeprowadzona analiza rynku gier komputerowych pod względem finansowym i naukowym wykazała zarówno znaczne ograniczenia stosowanych obecnie rozwiązań sieciowych, jak i rosnące zapotrzebowanie na rozwiązania umożliwiające jednoczesną grę wielu użytkownikom. Opisany w pracy protokół, oparty na rozproszonej architekturze i algorytmie plotkującym, powstał w celu przezwyciężenia tych ograniczeń. W pracy zaprezentowano jego strukturę, mechanizm rozprzestrzeniania danych po sieci oraz przeprowadzono weryfikację jego poprawności i wydajności.

Słowa kluczowe

algorytmy plotkujące, systemy rozproszone, gry wieloosobowe, protokoły sieciowe

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

C. Computer Systems Organization
C.2. Computer-Communication Networks
C.2.4. Distributed Systems

Tytuł pracy w języku angielskim

Application of gossip algorithms in massively multiplayer online games

Spis treści

Wprowadzenie 5

1. Analiza obecnych rozwiązań 7
1.1. Rynek gier komputerowych 7
1.2. Minimalne wymagania sprzętowe 8
1.3. Podstawowe pojęcia 10
1.4. Jakościowe metody oceny gier wieloosobowych 11
1.5. Architektura klient-serwer i jej ograniczenia 12
1.5.1. Stosowane optymalizacje 13
1.5.2. Wyniki empiryczne 14
1.6. Rozwiązanie Colyseus 16
1.7. Rozwiązanie Donnybrook 16

2. Silac: protokół oparty na algorytmie plotkującym 17
2.1. Przedmiot zainteresowania 17
2.2. Teoria 17
2.3. Algorytm plotkujący 18
2.4. Model wirtualnego świata 19
2.5. Struktura protokołu 20
2.5.1. Identyfikatory typów wiadomości 20
2.5.2. Nagłówek wiadomości 21
2.5.3. Prośba o dołączenie do rozgrywki 21
2.5.4. Akceptacja prośby o dołączenie do rozgrywki 22
2.5.5. Odrzucenie prośby o dołączenie do rozgrywki 23
2.5.6. Opuszczenie rozgrywki 23
2.5.7. Prośba o retransmisję danych o uczestniku 23
2.5.8. Informacja o interakcjach 24
2.5.9. Informacja o awatarze 24
2.5.10. Dane dotyczące protokołu 25
2.5.11. Informacja o aktywnych rozgrywkach 25
2.5.12. Informacja o aktywnej rozgrywce 26
2.5.13. Prośba o dodatkowe informacje o rozgrywce 26
2.5.14. Sygnał życia 26
2.6. Architektura 26
2.7. Oryginały i repliki 27
2.8. Obszar zainteresowania i ścieżka zainteresowania 29
2.9. Dobór odbiorców 29
2.9.1. Odległość 30

2.9.2. Funkcja celu . . . 30
2.9.3. Dostępne zasoby sieciowe . . . 31
2.10. Przekazywanie informacji . . . 31
2.11. Utrata wiadomości i mechanizm korekcyjny . . . 31
2.12. Równoważenie obciążenia . . . 32
2.13. Czas życia wiadomości . . . 33

3. Weryfikacja poprawności i wydajności protokołu . . . 35
3.1. Wyznaczanie parametrów protokołu . . . 35
3.1.1. Odległość i funkcja celu . . . 36
3.1.2. Odległość i przepustowość sieci . . . 37
3.1.3. Ponowny wybór węzła do komunikacji . . . 37
3.2. Szybkość rozprzestrzeniania się danych . . . 38
3.3. Ilość wysyłanych danych . . . 39
3.4. Ilość zgubionych informacji . . . 40
3.5. Liczba skoków wiadomości . . . 40
3.6. Spójność i sprawiedliwość . . . 41

4. Podsumowanie . . . 43

A. Załączniki na płycie CD . . . 45

Bibliografia . . . 47

Wprowadzenie

Zagadnienia związane z tematyką gier komputerowych są od dawna przedmiotem zainteresowania zarówno programistów, jak i użytkowników. Najwcześniejszą znaną grą interaktywną jest "Urządzenie do zabawy za pomocą kineskopu z działem elektronowym" [30], którą autorzy, Thomas T. Goldsmith Jr. *et al.*, opatentowali w roku 1947. Był to symulator rakiety, a rozgrywka bardzo przypominała obraz z radaru z czasów II wojny światowej. Rakietę reprezentowała pojedyncza kropka na ekranie, generowana przez działo elektronowe, natomiast cele znajdowały się na nakładce ekranu, przezroczystej folii, jako że nie posiadano wtedy możliwości tworzenia skomplikowanej grafiki.

Wraz z wprowadzeniem komputerów osobistych, gry komputerowe dotarły do szerokiego grona odbiorców. Dzisiaj występują w wielu odmianach, typach i wersjach, zaspokajając potrzeby i wymagania stale powiększającej i różnicującej się grupy odbiorców.

Na każdą grę komputerową składa się wiele subtelnych rozwiązań z szerokiego spektrum działów informatyki: sztucznej inteligencji, grafiki komputerowej czy sieci komputerowych. W każdej kolejnej generacji gier wykorzystywane są coraz bardziej wyrafinowane metody w celu jak najwierniejszego odwzorowania rzeczywistości w wirtualnym świecie i dostarczenia użytkownikowi satysfakcjonującej rozrywki.

Przedmiotem zainteresowania tej pracy są gry typu FPS (*ang. first-person shooter*) oraz RPG (*ang. role-playing game*). Pomimo wielu różnic w celach, jakie stawiają przed graczem, i w sposobach prowadzenia rozgrywki, posiadają one istotne cechy wspólne. Najważniejszą z nich jest możliwość prowadzenia rozgrywki, w której za pośrednictwem sieci uczestniczy wielu graczy.

W chwili obecnej komercyjne gry do prowadzenia rozgrywek wieloosobowych wykorzystują scentralizowaną architekturę klient-serwer. Takie rozwiązanie posiada liczne ograniczenia i tylko niewiele gier oferuje jednoczesną rozgrywkę więcej niż trzydziestu dwóm graczom. Prowadzone są jednak liczne projekty badawcze, wykorzystujące architekturę rozproszoną, mające na celu przezwyciężenie tych ograniczeń [2, 5, 6, 18].

Celem pracy jest opracowanie protokołu komunikacyjnego dla wymienionych typów gier, umożliwiającego przeprowadzenie rozgrywki, w której jednocześnie uczestniczy kilkuset graczy. Z powodu specyficznego charakteru gier komputerowych, protokół musi pozwalać na szybką i częstą wymianę dużych ilości danych. Z tego powodu protokół powinien funkcjonować z tylko niewielką ilością danych systemowych, niezwiązanych z rozgrywką. Jednocześnie nie jest wymagane zapewnienie niezawodności komunikacji, co obniża wymagania stawiane przed ewentualnymi mechanizmami korekcyjnymi.

Praca podzielona jest na trzy części. W części pierwszej została przeprowadzona analiza rozwoju oraz obecnego stanu rynku gier komputerowych. Opisałem najczęściej stosowane algorytmy umożliwiające rozgrywki wieloosobowe oraz wykazałem ich bardzo duże ograniczenia. Zaprezentowałem także dwa projekty badawcze, które wychodzą naprzeciw problemom związanym z obecnymi rozwiązaniami.

W części drugiej zaproponowałem nowe spojrzenie na problemy gier wieloosobowych i opi-

sałem autorski protokół, który pozwoli na ich rozwiązanie.

W części trzeciej pokazałem wyniki testów przeprowadzonych z zastosowaniem nowego protokołu oraz porównałem go z istniejącymi rozwiązaniami. Pracę zakończyłem podsumowaniem, a w załącznikach zaprezentowałem opracowane programy i algorytmy.

Rozdział 1

Analiza obecnych rozwiązań

1.1. Rynek gier komputerowych

Rynek gier komputerowych jest obecnie drugim najszybciej rozwijającym się segmentem przemysłu rozrywkowego i medialnego. Według PricewaterhouseCoopers w ciągu lat 2010-2014 wartość tego rynku wzrośnie z 52 do 87 miliardów dolarów, osiągając skumulowaną roczną stopę wzrostu równą 10,6% [25]. Jedynym segmentem o większym przewidywanym tempie wzrostu są reklamy internetowe.

Rok	Przychód rynku gier komputerowych w USA (w miliardach dolarów)	Przychód rynku gier komputerowych na świecie (w miliardach dolarów)
1999	6,9	Brak jednoznacznych danych
2000	6,6	14,7
2001	11,0	Brak jednoznacznych danych
2002	11,7	20,7
2003	11,2	Brak jednoznacznych danych
2004	11,0	27,2
2005	11,5	29,0
2006	13,5	31,7
2007	18,9	41,7
2008	22,0	54,0
2009	19,7	Brak danych

Tabela 1.1: Rozwój rynku gier komputerowych w latach 1999-2009. Źródło: NPD annual sales figures [24] i opracowanie własne.

W tabeli 1.1 zawarto informacje publikowane corocznie przez NPD Group, Inc. oraz dane opracowane przez autora na podstawie raportów o przychodach rynków krajowych w USA, Japonii, Kanadzie, Korei Południowej oraz Wielkiej Brytanii. Niestety żadne inne dane na temat globalnej wartości rynku nie są publikowane. Ponadto nie istnieje jeden system wyceny. W powyższym opracowaniu przedstawiono łączny przychód ze sprzedaży gier komputerowych, gier na konsole oraz sprzętu i akcesoriów do gry. Z powodu braku jednoznacznych globalnych danych, opracowanie oparte będzie na rynku gier komputerowych w Stanach Zjednoczonych. Jest to uzasadnione również dlatego, że wartość rynku USA stanowi około 40% wartości rynku światowego. Dodatkowego komentarza wymaga nagły wzrost wartości rynku

w roku 2001. Było to spowodowane premierą trzech konsol do gier: Nintendo GameCube, Game Boy Advance oraz Xbox.

Tylko w 2009 roku rynek gier komputerowych w Stanach Zjednoczonych wygenerował przychód równy 20 miliardom dolarów, a jego wartość dodana do produktu krajowego brutto wyniosła 5 miliardów dolarów [14], co stanowiło 1% całego PKB kraju. W 2009 roku 67% gospodarstw domowych w Stanach Zjednoczonych posiadało komputer bądź konsolę do gier [13]. Odsetek ten od pięciu lat wzrastał o co najmniej 1p.p. rocznie. Chociaż tempo jego wzrostu stale maleje, rynek jeszcze się nie nasycił i możemy się spodziewać, że w nadchodzących latach liczba użytkowników gier komputerowych będzie nadal się zwiększać.

W swoim raporcie ESA [13] jednoznacznie pokazuje, że w społeczności graczy jest reprezentowana każda grupa demograficzna. Wbrew powszechnemu przekonaniu średni wiek gracza wynosi już 34 lata. Co więcej, przeciętny gracz gra od 12 lat, jest zatem, w szerokim rozumieniu tego terminu, lojalnym klientem. Co oznacza, że z dużym prawdopodobieństwem dokona kolejnego zakupu gry, sprzętu komuterowego bądź konsoli do gier.

Istotnym wnioskiem raportu ESA jest spostrzeżenie, że żaden gatunek gier komputerowych nie dominuje na rynku. Wyjątkiem są gry strategiczne, które stanowiły jedną trzecią wszystkich gier sprzedanych na komputery osobiste.

Wraz ze wzrastającą liczbą graczy coraz większą popularnością cieszą się gry dla wielu graczy, które w skrajnych przypadkach umożliwiają grę tysiącom osób naraz. Najlepszym przykładem takiej gry jest World of Warcraft, który posiada 12 milionów zarejestrowanych użytkowników [7]. 17% graczy regularnie płaci, aby grać w gry sieciowe, co tylko w Ameryce Północnej i Europie przyniosło dochód równy 1,4 miliardom dolarów [16]. Przy tak dynamicznie rozwijającym się rynku, zastanawiająca jest stagnacja jaka zapanowała w używanych przez niego technologiach. Wszystkie gry typu MMOG (*ang. massively multiplayer online game – gra sieciowa dla wielu graczy*) korzystają z architektury klient-serwer ze specjalnymi dedykowanymi serwerami. Gry, które nie wymagają dedykowanych serwerów i pozwalają graczom tworzyć rozgrywki ad-hoc, najczęściej nie umożliwiają wspólnej gry więcej niż 64 graczom.

1.2. Minimalne wymagania sprzętowe

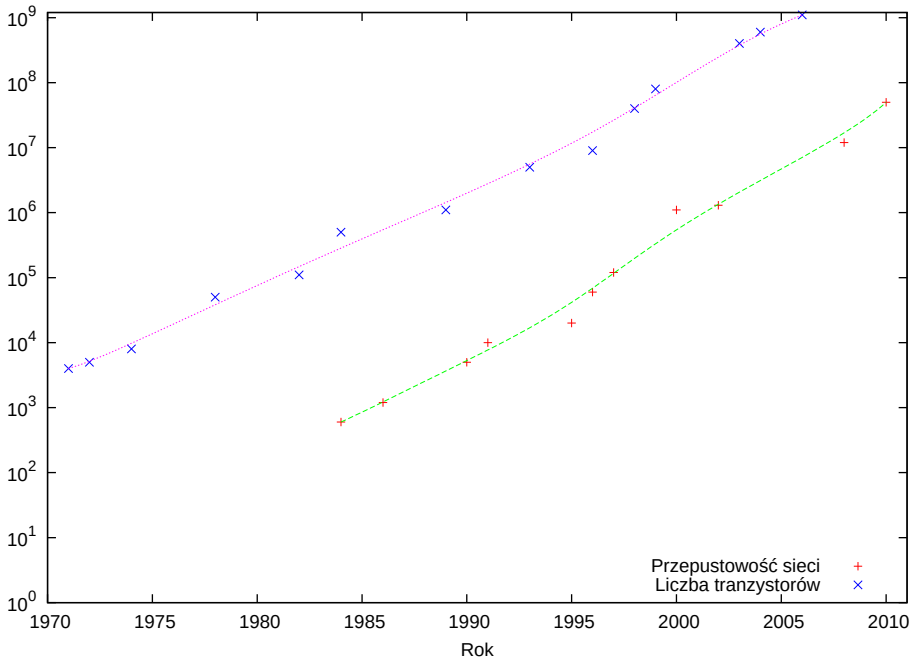
Gry komputerowe stoją przed takimi samymi problemami, jak wszystkie inne programy komputerowe: są ograniczone przez możliwości sprzętu. Możliwości gier komputerowych w największym stopniu zależą od: karty graficznej, procesora, pamięci podręcznej i przepustowości łącza.

Nazwa gry	Rok wydania	Pamięć karty graficznej	Procesor	Pamięć podręczna	Przepustowość łącza
Wolfenstein 3D	1992	256-color VGA	80286 25MHz	640KB	Brak
Quake	1996	1MB	66MHz	8MB	32kbps
Deus Ex	2000	4MB	300MHz	64MB	64kbps
Doom 3	2004	64MB	1,5GHz	384MB	64kbps
Far Cry 2	2008	256MB	4,2Ghz	2GB	128kbps

Tabela 1.2: Porównanie minimalnych wymagań sprzętowych. Źródło: opracowanie własne.

Z tabeli 1.2 wynika, że w trzech kategoriach zaobserwowano wzrost wykładniczy, który

związany jest z potencjalnymi możliwościami sprzętu komputerowego. Twórcy gier starali się możliwie najpełniej wykorzystać sprzęt dostępny w momencie tworzenia danej gry. Biorąc to pod uwagę, jak i również prawo Moore'a [20, 21], można z dużą pewnością założyć, że gry komputerowe w ciągu najbliższych lat będą rozwijać się w podobnym tempie.



Rysunek 1.1: Wzrost liczby tranzystorów oraz przepustowości łącza. Źródło: opracowanie własne na podstawie danych z Jacob Nielsen's Alertbox [23].

Z wykresu 1.1 wynika, że moc obliczeniowa procesorów rośnie szybciej niż przepustowość łącza. Dokładnie, przepustowość sieci co roku rośnie o 50%, podczas gdy moc obliczeniowa procesora rośnie o 60%. W ciągu 10 lat oznacza to stukrotny wzrost mocy obliczeniowej przy tylko pięćdziesięciosiedmiokrotnym wzroście przepustowości. Należy jednak zauważyć, że o wiele łatwiej jest wymienić procesor, a nawet cały komputer niż jedno łącze internetowe. Oznacza to, że w przypadku rozgrywek wieloosobowych głównym ograniczeniem sprzętowym jest i przez najbliższe lata pozostanie przepustowość łącza, a w efekcie ilość danych, które komputer jednego gracza może wysłać w ciągu jednej sekundy.

Należy zaznaczyć, że niemniejszy wpływ na jakość rozgrywki mają opóźnienia między kolejnymi pakietami informacji [3]. Bardzo trudno jednak podać obiektywne kryterium ilościowe dla tego parametru gry komputerowej, ponieważ jakość rozgrywki jest subiektywną oceną gracza, a rozgrywkę wieloosobową można z powodzeniem przeprowadzić nawet przy bardzo dużych opóźnieniach, tj. rzędu 1 sekundy. Z tego powodu maksymalne dopuszczalne opóźnienia nie zostały uwzględnione w tabeli 1.2.

Wolny wzrost wymagań przepustowości łącza jest spowodowany specyficznym wykorzystaniem tego zasobu przez gry. Mimo stałego wzrostu rozmiarów i stopnia skomplikowania wirtualnych światów, czego przykładem są gry opisane w tabeli 1.2, liczba elementów dynamicznych pozostaje prawie bez zmian. W omawianej sytuacji elementami dynamicznymi są gracze oraz niektóre, nieliczne elementy świata, na które gracze mogą oddziaływać i które mogą zmieniać. Zważywszy, że obecnie gry komputerowe umożliwiają rozgrywkę wieloosobową co najwyżej 64 graczom, ilość danych, które muszą zostać przesłane stanowi tylko niewielką część wirtualnego świata.

1.3. Podstawowe pojęcia

Każdy z graczy posiada swoje wirtualne alter-ego: awatara, nad którym sprawuje kontrolę. Nadrzędnym celem każdego z graczy jest zapewnienie, że jego awatar przeżyje do końca rozgrywki. W zależności od typu gry, warunkiem koniecznym do zwycięstwa może być eliminacja awatarów innych graczy lub zajęcie wyróżnionego elementu planszy. Podczas rozgrywki gracze mogą działać w drużynach, samodzielnie lub tworząc doraźne sojusze. W grach wieloosobowych głównym przeciwnikiem nie jest sztuczna inteligencja, lecz człowiek, który także chce wygrać, zatem możliwości przebiegu rozgrywki są praktycznie nieograniczone.

Awatary poruszają się po wirtualnym świecie, który pełni tutaj rolę planszy. Wraz z rozwojem sprzętu komputerowego, graczom oferowane są coraz większe, coraz bardziej realistyczne światy wirtualne.

Rysunek 1.2: Porównanie The Elder Scrolls 2: Daggerfall (1996) oraz The Elder Scrolls 4: Oblivion (2006). Źródło: Bethesda Softworks [4].

Rysunek 1.2 prezentuje skalę zmian, jakie zaszły w jakości odwzorowania świata w grach komputerowych w ciągu tylko dziesięciu lat. Dzisiaj gry oferują graczom wirtualne światy o rozmiarach ponad 40 kilometrów kwadratowych [9].

Istotną cechą rozważanych gier jest ich duża dynamika. Sytuacja w wirtualnym świecie zmienia się bardzo szybko wraz z ruchem awatarów, ich oddziaływaniem na otoczenie oraz na siebie nawzajem. Kwantem czasu w tych grach jest klatka (*ang. frame*), czyli pojedynczy, statyczny obraz wyświetlony na monitorze, wygenerowany zgodnie z obecnym stanem świata. Po wyświetleniu klatki gra analizuje wszystkie przychodzące dane i stan świata jest aktualizowany, po czym generowana jest nowa klatka. Liczba generowanych i wyświetlanych w ciągu jednej sekundy klatek nazywa się **szybkością klatek** (*ang. frame-rate*). Aby oszukać ludzkie oko i zapewnić mu złudzenie płynności obrazu, szybkość klatek musi wynosić co najmniej 24 klatki na sekundę.

Obecne rozwiązania, niezależnie od wykorzystywanej architektury, opierają się na wymianie informacji o elementach wirtualnego świata pomiędzy właścicielami oryginalnych elementów i właścicielami ich kopii. Właścicieli obiektów wirtualnego świata będziemy nazywać uczestnikami rozgrywki bądź węzłami. Oryginał jest w posiadaniu dokładnie jednego uczestnika rozgrywki (serwera bądź klienta), natomiast pozostali uczestnicy posiadają jego kopie, zwane potocznie replikami. Podczas rozgrywki komunikaty sieciowe są wykorzystywane do zapewnienia spójności stanów oryginałów i replik.

1.4. Jakościowe metody oceny gier wieloosobowych

W literaturze przedmiotu trudno spotkać opis metod oceny jakościowej wieloosobowych gier komputerowych. Istniejące metody ograniczają sie do prostego porównania wartości opóźnień między węzłami lub liczby elementów świata, o których utracono informacje w trakcie komunikacji [5, 6]. W pracy do oceny jakości gier wieloosobowych zaproponowano dwie metryki: **spójność** oraz **sprawiedliwość**.

Spójność oznacza stopień, w jakim wirtualne światy widziane przez różnych graczy są do siebie podobne oraz, w przypadku pojawienia się różnic, jak szybko są one korygowane.

Sprawiedliwość wyraża przewagę, jaką jedni uczestnicy mają nad innymi. Przy czym nie wynika ona z ich osobistych umiejętności bądź logiki samej gry. Dwa podane pojęcia pozwalają ocenić i porównać dwie gry wieloosobowe. Przede wszystkim spójność i sprawiedliwość są ściśle związane z sieciowym aspektem gry i tylko w niewielkim stopniu z jej pozostałymi elementy, takimi jak logika i cele rozgrywki czy sposób reprezentowania świata. Należy pamiętać, że bardzo istotny wpływ mają na nie przepustowość łączy, występujące opóźnienia oraz heterogeniczność całej sieci.

W celu porównywania dwóch gier względem zapewnianej przez nie spójności wprowadzę następującą metrykę: dla każdego uczestnika rozgrywki n zdefiniuję wartość Inc_n:

$$inc_n(t) = \sum_{r \in R_n} \frac{diff(r,t)}{d_{n,r}^2} \tag{1.1}$$

$$Inc_n = \frac{1}{T} \int_0^T inc_n(t) \mathrm{d}t \tag{1.2}$$

gdzie:

R_n	zbiór replik posiadanych przez uczestnika rozgrywki n,
$d_{n,r}$	odległość repliki r od awatara uczestnika rozgrywki n,
T	czas trwania całej rozgrywki,

$$diff(r,t) = \begin{cases} 1 & \text{gdy replika } r \text{ w chwili } t \text{ jest w innym stanie niż jej oryginał} \\ 0 & \text{wpp} \end{cases}$$

$inc_n(t)$ oznacza zatem ile replik, które istnieją u uczestnika n, nie jest w chwili t identycznych ze swoimi oryginałami. Mnożenie przez czynnik $1/d_{n,r}^2$ odzwierciedla fakt, że najistotniejsze dla uczestnika rozgrywki są repliki istniejące blisko niego. Możliwe jest wzmocnienie podanej metryki poprzez uwzględnianie także widoczności replik, jednak nie wprowadza to istotnych zmian, a znacznie komplikuje obliczenia. Inc_n jest średniosekundowym rozspójnieniem świata uczestnika n. Ostatecznie definiuję średnie rozspójnienie całego świata, rozumiane jako suma rozspójnień światów wszystkich uczestników rozgrywki:

$$Inc = \frac{1}{N} \sum_{k=1}^{N} Inc_k \tag{1.3}$$

W podobny sposób zdefiniuję metrykę dla sprawiedliwości. Niestety, ponieważ sprawiedliwość nie jest tak konkretnym terminem jak spójność, będę posługiwać się tylko jednym z jej aspektów. Przede wszystkim sprawiedliwość jest bardzo ściśle związana ze spójnością wirtualnych światów obserwowanych przez uczestników rozgrywki. Jeżeli spośród dwóch uczestników rozgrywki pierwszy doświadcza większego opóźnienia w aktualizacji stanu widzianych przez

siebie replik, to drugi zdobywa przewagę. Istotnie, pierwszy uczestnik obserwuje obraz wirtualnego świata drugiego uczestnika, który istniał pewien czas temu w przeszłości. W zależności od różnicy opóźnień, może być to obraz bardzo różny od aktualnego. Naturalnie opóźnienia, które wynikają z właściwości sieci nie powinny być uwzględniane przy badaniu sprawiedliwości danej gry. Zdefiniuję zatem dla uczestnika rozgrywki n wartość Inj_n:

$$inj_n(t) = \sum_{k=1}^{N} \frac{|Delay(n,k,t) - Delay(k,n,t)|}{d_{n,k}^2} \tag{1.4}$$

$$Inj_n = \frac{1}{T} \int_0^T inj_n(t)\mathrm{d}t \tag{1.5}$$

gdzie:

$d_{n,k}$	odległość awatara uczestnika n od awatara uczestnika k,
$Delay(n,k,t)$	opóźnienie uczestnika n względem uczestnika k w chwili t.

Ponownie mnożę przez czynnik $1/d_{n,k}^2$, aby odzwierciedlić większą wagę elementów, które są blisko awatara uczestnika rozgrywki. Otrzymuję zatem średnie opóźnienie jakiego doświadcza uczestnik n podczas trwania rozgrywki. Ostatecznie wyznaczam średnie opóźnienie doświadczane przez użytkownika:

$$Inj = \frac{1}{N} \sum_{k=1}^{N} Inj_n \tag{1.6}$$

Powyższą definicję sprawiedliwości można zastąpić mniej konkretnym pytaniem: "czy gracze są nierozróżnialni z dokładnością do przepustowości łącz i występujących opóźnień?". Zaletą takiego pytania jest to, że w prosty sposób, bez odwoływania się do rachunków i testów, wyraża istotę sprawiedliwości gry wieloosobowej. W niektórych przypadkach można udzielić na nie odpowiedzi znając tylko protokół sieciowy, z jakiego korzysta dana gra.

1.5. Architektura klient-serwer i jej ograniczenia

Obecnie dominującą architekturą wykorzystywaną przez gry wieloosobowe jest architektura klient-serwer. W tym rozwiązaniu każdy klient wysyła serwerowi dane dotyczące swojego lokalnego gracza, następnie serwer oblicza nowy stan wirtualnego świata, rozwiązując wszelkie konflikty, jakie mogły się pojawić pomiędzy graczami i odsyła wszystkim graczom zaktualizowany obraz całości, bądź tylko części, wirtualnego świata. Serwerem może być komputer jednego z graczy lub maszyna dedykowana, czyli specjalny, wydzielony komputer, który nie posiada swojego lokalnego gracza, a zatem nie musi przetwarzać operacji wejścia-wyjścia z kontrolera i monitora. Przy takiej architekturze najbardziej obciążone jest łącze internetowe serwera, przez które muszą przejść najpierw komunikaty od wszystkich klientów, a następnie odpowiedzi na nie.

Ilość danych, jakie serwer musi wysłać w ciągu jednej sekundy wyraża się wzorem:

$$S = O(n * s * f) \tag{1.7}$$

gdzie:

n	liczba dynamicznych elementów wirtualnego świata,
s	rozmiar dynamicznego elementu,
f	szybkość klatek.

Jeżeli założymy, że:

1. w wirtualnym świecie jedynymi dynamicznymi elementami są gracze, czyli cały świat jest statyczny i gracze nie mogą w żaden sposób na niego oddziaływać,
2. rozmiar pojedynczego gracza to 24 bajty, co pozwala na zakodowanie tylko położenia oraz zwrotu gracza,
3. szybkość klatek jest równa 24,
4. serwer wysyła pakiet danych tylko raz, niezależnie od liczby graczy, czyli trasowanie pakietów jest rozwiązane na poziomie sieci, a nie logiki gry,

to, aby przeprowadzić rozgrywkę 64 graczy, wymagany byłby serwer o przepustowości łącza równej co najmniej 288 kbps. Z założenia 4. można skorzystać tylko wtedy, gdy dostępny jest multicasting bądź podobny do niego mechanizm trasowania. W przypadku, gdy taki mechanizm nie jest dostępny, serwer musi wysłać pakiet do każdego z graczy osobno i wymagana przepustowość łącza wzrasta do 18 Mbps. Przy czym należy zwrócić uwagę, że poczyniono bardzo silne założenia co do charakteru wirtualnego świata.

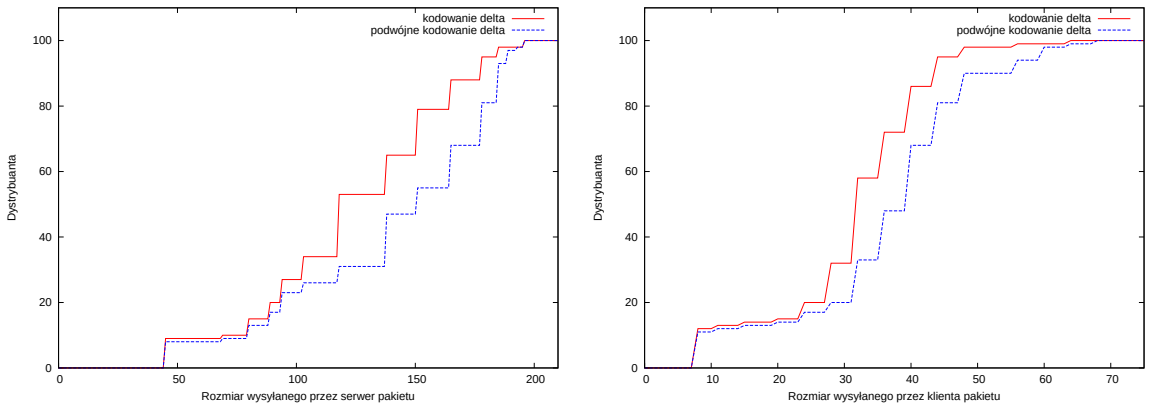
1.5.1. Stosowane optymalizacje

W praktyce stosuje się dwie optymalizacje, aby zmniejszyć ilość transmitowanych danych. Pierwszą z nich jest filtrowanie względem obszaru zainteresowania. Każdy z graczy dostaje informacje dotyczące tylko najbliższego otoczenia jego awatara. Jest to naturalne rozwiązanie wynikające z obserwacji, że gracza znacznie mniej interesuje stan elementów, które są w znacznej odległości od jego awatara.

Drugą optymalizacją jest kodowanie delta, czyli przesyłanie informacji tylko o zmianach, jakie zaszły w elementach wirtualnego świata, a nie o całkowitym stanie elementów. Brak informacji o danym elemencie oznacza, że nie dokonały się w nim żadne zmiany od czasu ostatniej transmisji danych. Dzięki temu, kosztem dodatkowych obliczeń polegających na aplikowaniu delt, istotnie zmniejsza się rozmiar przesyłanych pakietów. Ponadto nadawca i odbiorca muszą wiedzieć względem jakiego stanu obliczane są przesyłane delty, dlatego podczas każdej wymiany komunikatów przesyłają sobie dodatkową informację o ostatniej zaaplikowanej do swojego wirtualnego świata zmianie. W sytuacji utraty pakietu z powodu zawodności sieci wykorzystywany jest prosty mechanizm korekcyjny. Niedoszły odbiorca, zauważywszy, że nie otrzymał pakietu, zwraca się do nadawcy o powtórne przesłanie informacji w nim zawartych. Identyfikacja utraconego pakietu odbywa się najczęściej za pomocą zegarów logicznych. Nadawca może odpowiedzieć wysyłając pełną informację o elementach, których zmiany były zawarte w utraconym pakiecie lub zwiększoną deltę zawierającą informacje o zmianach względem ostatniego wspólnie uzgodnionego stanu. Niezależnie od wyboru mechanizmu korekcyjnego, przesyłanie tylko zmian stanów pozwala na znaczne zmniejszenie wykorzystywanej przepustowości łącza.

Istnieje rozwinięcie tej optymalizacji, nazywane podwójnym kodowaniem delta, które zakłada, że przesyłane są informacje o zmianach, które nastąpiły od czasu przedostatniej transmisji danych. Oznacza to, że każda informacja zostanie przesłana dwukrotnie, czyli mechanizm korekcyjny musi zostać zastosowany dopiero w przypadku utraty dwóch kolejnych pakietów.

Wykresy 1.3a oraz 1.3b prezentują dystrybuanty rozmiaru pakietów wysyłanych przez, odpowiednio, serwer oraz klienta w przypadku zastosowania pojedynczego i podwójnego kodowania delta. Rozmiar pakietów bez kodowania delta wynosi 196 bajtów w przypadku ser-



(a) (b)

Rysunek 1.3: Dystrybuanty rozmiarów przesyłanych pakietów dla serwera i klienta. Źródło: opracowanie własne na podstawie danych empirycznych Game Environments Internet Utilisation Study [15].

wera oraz 75 bajtów w przypadku klienta. Zatem zastosowanie kodowania delta pozwala na ponaddwukrotne zmniejszenie rozmiaru pakietu.

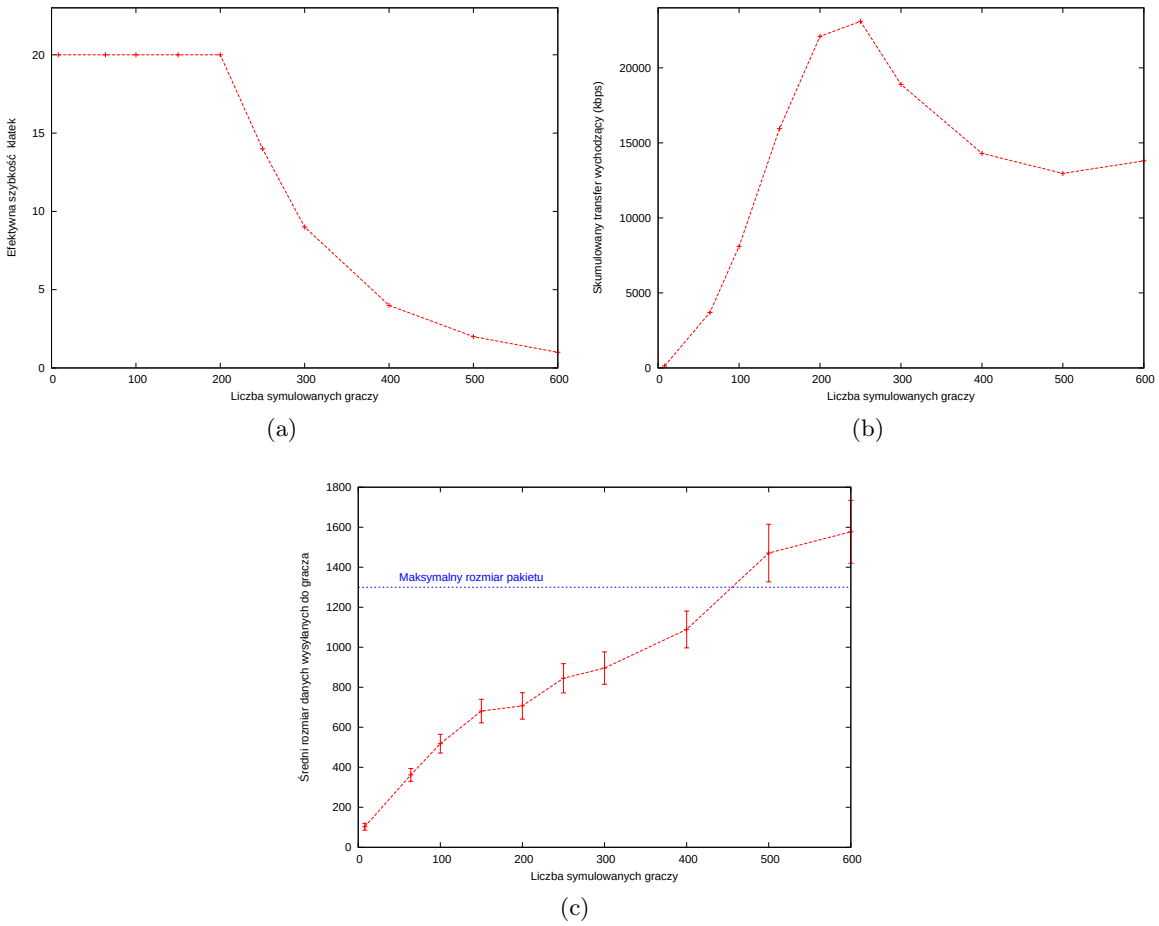
Dzięki zastosowaniu dwóch opisanych optymalizacji możliwe jest znaczne ograniczenie wysyłanych przez serwer danych.

1.5.2. Wyniki empiryczne

Przeprowadzono eksperyment na komputerze Pentium Core 2 Duo 2,1Ghz z 1GB RAM. Uruchomiono serwer gry Quake 2 [26] z określoną liczbą klientów. W eksperymencie klienci byli symulowani przez sztuczną inteligencję, której obliczenia odbywały się na serwerze. Mimo to serwer wysyłał do nich swoje dane tak, jakby byli prawdziwymi klientami, znajdującymi się na różnych maszynach. Przeniesienie wszystkich obliczeń na serwer miało na celu pomóc zidentyfikować główne ograniczenia architektury klient-serwer. Serwer korzystający z filtrowania względem obszaru zainteresowania oraz pojedynczego kodowania delta, początkowo został tak skonfigurowany, aby odświeżać stan świata, a zarazem wysyłać dane do klientów, dwadzieścia razy na sekundę. Każda rozgrywka trwała 10 minut.

Wykres 1.4a prezentuje efektywną szybkość klatek osiągniętą przez serwer w zależności od liczby klientów. Istotny spadek szybkości klatek przy 250 klientach oznacza, że obliczanie obszarów zainteresowań stało się operacją zajmującą najwięcej czasu. Istotnie, jest to operacja wymagająca czasu $O(n^2)$, gdzie n jest liczbą uczestników rozgrywki. Z drugiej strony, niezastosowanie filtrowania spowodowałoby opisany wcześniej kwadratowy wzrost liczby wysyłanych pakietów.

Wykres 1.4b pokazuje superliniową zależność pomiędzy ilością wysyłanych w ciągu sekundy danych a liczbą klientów. Od pewnego momentu widoczny jest spadek ilości wysyłanych danych spowodowany spadkiem szybkości klatek serwera, co oznaczało zmniejszenie liczby wysyłanych w ciągu sekundy pakietów. Nie oznacza to jednak, że możliwe jest prowadzenie satysfakcjonującej rozgrywki dla 500 graczy. Wraz ze spadkiem szybkości klatek serwera spada liczba aktualizacji stanu wirtualnego świata dokonywanych w ciągu sekundy, co, przy niezmienionej i znacznie większej szybkości klatek u klientów, prowadzi do znaczących opóźnień i rozspójnienia świata.



Rysunek 1.4: Wyniki eksperymentu. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Wykres 1.4c prezentuje średni rozmiar wysyłanego przez serwer pakietu danych. Naturalnie w pojedynczym pakiecie wysyłanym do danego klienta znajdowały się informacje dotyczące awatarów będących w jego obszarze zainteresowania. Oznacza to, że wraz ze wzrostem liczby graczy, przy stałych rozmiarach planszy, zwiększa się także liczba awatarów przebywających w obszarze zainteresowania pojedynczego gracza. Potwierdza to teorię, że w grach komuterowych gracze zachowują się podobnie jak ludzie w tłumie: mają tendencję do zbierania się w grupy [8, 22, 29]. Chociaż w przeprowadzonym eksperymencie rozmiar grup był zależny tylko subliniowo od całkowitej liczby graczy, to doprowadził do zwiększenia wielkości pakietu ponad dopuszczalny przez logikę gry maksymalny rozmiar. W takiej sytuacji dochodziło do fragmentacji komunikatów i dalszego zwiększania ich rozmiarów, spowodowanego wzrostem procentowego udziału danych systemowych w postaci nagłówków wykorzystywanych protokołów.

Opisany eksperyment dowodzi, że architektura klient-serwer ma duże ograniczenia. Co więcej, próby zmniejszenia obciążenia jednego zasobu prowadzą do wprost proporcjonalnego zwiększenia zapotrzebowania na inny.

Mimo istotnych ograniczeń ilościowych, architektura klient-serwer posiada pewne zalety. Przede wszystkim jest prosta w implementacji i zapewnia bardzo dużą spójność świata. Istotnie, ponieważ wszelkie obliczenia związane z logiką gry odbywają się na serwerze i są jed-

nocześnie rozsyłane do wszystkich uczestników rozgrywki, różnice pojawiają się tylko wskutek opóźnień lub utraty pakietu.

W ostatnich latach próbowano przezwyciężyć ten problem poprzez zastosowanie rozproszonej architektury. Mimo to w chwili obecnej nie istnieje komercyjne rozwiązanie, a projekty przedstawione w kolejnych punktach są projektami badawczymi.

1.6. Rozwiązanie Colyseus

Bharambe *et al.* [5] proponują rozwiązanie oparte na rozproszonych tablicach haszujących. Ich projekt, nazwany Colyseus, zakłada podzielenie wirtualnego świata pomiędzy klientów. Wykorzystując fakt, że większość wirtualnego świata jest statyczna i niezmienna, Colyseus zarządza tylko elementami dynamicznymi. Każdy taki element istnieje jako jedna kopia główna (oryginał) na jednym z klientów oraz kilkanaście kopii drugorzędnych (replik) na pozostałych klientach. Przy czym wszystkie kopie główne posiadają swoje obszary zainteresowania. Za każdym razem, gdy replika znajdzie się w obszarze zainteresowania kopii głównej, klient korzysta z tablicy haszującej, aby znaleźć kopię główną napotkanej repliki. Następnie uspójniane są informacje bezpośrednio pomiędzy repliką a jej kopią główną.

Rozwiązanie powoduje, że każdy klient aktywnie poszukuje interesujących go obiektów i inicjuje u pozostałych klientów transfer tylko niezbędnych dla siebie danych. Ceną tego jest relatywnie wysoka ilość wysyłanych danych systemowych podczas poszukiwania kopii głównych, która wynosi od 30 do 50% wszystkich przesyłanych danych.

1.7. Rozwiązanie Donnybrook

Bharambe *et al.* [6] w ramach Microsoft Research zaproponowali rozwiązanie oparte na bardzo ograniczonych obszarach zainteresowania. U podstaw ich projektu, nazwanego Donnybrook, leży obserwacja, że uwaga jaką gracz poświęca elementom wirtualnego świata jest nie tylko ograniczona, ale także skupiona na niewielkiej ich liczbie. Każdy klient ocenia jak bardzo jest zainteresowany pozostałymi klientami. Następnie wysyła wiadomość do kilku najbardziej interesujących go klientów, aby wysyłali do niego informacje o sobie. Gdy jakiś klient przestanie być interesujący, wysłana do niego zostanie wiadomość, aby zaprzestał o sobie informować. Dzięki temu każdy klient wie, którzy z pozostałych klientów są nim zainteresowani i tylko do nich wysyła dokładne informacje o sobie. Pozostałym klientom wysyła tylko wskazówki pozwalające na w miarę poprawne działanie replik.

Ponadto Donnybrook implementuje metodę równoważenia obciążenia, która zapewnia klientom o mniejszej przepustowości sieci pomoc innych klientów. Informacja zostanie najpierw wysłana do klientów wspomagających, którzy następnie przekażą ją do pozostałych zainteresowanych. Jest to konieczne, ponieważ o ile każdy klient posiada górne ograniczenie na liczbę elementów, które go interesują, to nie jest ograniczona liczba klientów, którzy nim będą się interesować. Możliwa jest zatem sytuacja, gdy bardzo wielu klientów oczekuje informacji od pojedynczego klienta, który nie posiada wystarczającej przepustowości łącza, aby im jej udzielić.

Rozdział 2

Silac: protokół oparty na algorytmie plotkującym

2.1. Przedmiot zainteresowania

Przedstawiony w pracy protokół, nazwany przez autora Silac, został opracowany dla gier typu FPS oraz RPG. Pomimo istotnych różnic między grami tych typów posiadają one zestaw wspólnych cech i wynikających z nich wymagań, które stawiają wykorzystywanym przez siebie protokołom. Tymi cechami są m.in.:

- rozgrywka toczy się w wirtualnym świecie,
- uczestnicy rozgrywki biorą w niej udział za pomocą awatarów,
- wirtualny świat i awatary zmieniają się wskutek działań uczestników rozgrywki,
- wirtualny świat zmienia się często (co najmniej kilka razu na sekundę),
- każdy uczestnik w danej chwili widzi tylko kawałek wirtualnego świata,
- uczestnicy rozgrywki dołączają do niej i opuszczają ją w dowolnym momencie.

Częstotliwość zmian wirtualnego świata zależy od indywidualnej gry, ale w większości przypadków jest niemniejsza niż 10 (dla gier typu RPG) i 20 (dla gier typu FPS) razy na sekundę.

Protokół dla gry wieloosobowej nie może istnieć bez wirtualnego świata i poruszających się w nim awatarów. Prezentowany w pracy protokół został opracowany dla logiki gry FPS o tytule Quake 3 Arena [27]. Chociaż gra została wydana w 1999 roku, nadal cieszy się niesłabnącą popularnością oraz, co ważniejsze, na jej bazie powstało wiele późniejszych tytułów gatunku. Ponadto zastosowane w niej rozwiązania i pomysły stały się nieoficjalnym standardem. Dzięki temu opracowany protokół można, przy niewielkich modyfikacjach, z powodzeniem zastosować do wielu innych gier typu FPS czy RPG.

2.2. Teoria

Badania przeprowadzone w ramach tej pracy wykazały, że scentralizowana architektura nie jest w stanie zapewnić jednoczesnej gry kilkuset graczom. W pracy zastosowałem rozproszoną architekturę w celu przezwyciężenia tych ograniczeń. Należy stwierdzić, że takie rozwiązanie nie jest doskonałe i posiada swoje własne ograniczenia. Przede wszystkim podlega twierdzeniu

CAP, zwanego też twierdzeniem Brewera [19]. Zgodnie z nim niemożliwe jest, aby rozproszony system oferował jednocześnie wszystkie trzy cechy:

- spójność,
- dostępność,
- tolerancję na podział danych.

W przypadku rozważanych gier komputerowych wszystkie trzy cechy są istotne, co oznacza, że każde rozwiązanie oparte na rozproszonej architekturze, będzie w pewnym sensie wadliwe.

Ze względu na wcześniej wymienione cechy gier leżących w obszarze zainteresowania tej pracy, uznałem, że spośród trzech wymienionych cech najważniejsze są dwie ostatnie: awaria jednego węzła nie może prowadzić do awarii całej sieci oraz gra musi działać pomimo utraty niektórych informacji. Na bazie tych założeń powstało oprogramowanie pośredniczące (*ang. middleware*) pomiędzy logiką gry wieloosobowej a protokołem internetowym. Zaimplementowane rozwiązanie składa się z systemu zarządzania komunikacją w rozproszonym środowisku oraz autorskiego protokołu wymiany wiadomości wykorzystującego UDP (*ang. User Datagram Protocol*). Całość została nazwana protokołem Silac.

2.3. Algorytm plotkujący

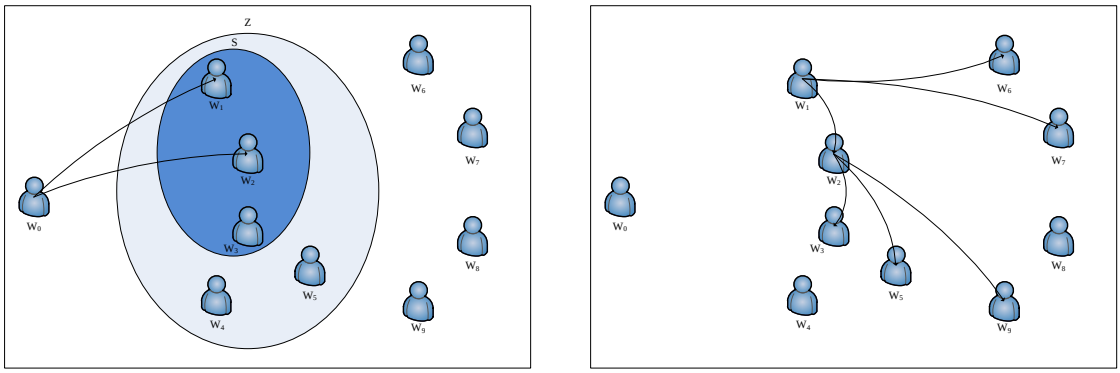
U podstawy projektu leży algorytm plotkujący, który odpowiada za rozprzestrzenianie informacji pomiędzy uczestnikami rozgrywki. Wykorzystano pomysł przedstawiony przez Nielsa Drosta *el al.* w ich artykule o ARRG (*Actualized Robust Random Gossiping*) [12]. Spośród popularnych rozwiązań [1, 31] ARRG wyróżnia się nie tylko prostotą implementacji i działania, ale także niezawodnością komunikacji w przypadku utraty węzłów i wiadomości. Zaimplementowany w pracy algorytm posiada kilka istotnych różnic w porównaniu do ARRG. Przede wszystkim nigdy nie usuwa informacji o znanych sobie węzłach, co eliminuje potrzebę posiadania dodatkowej pamięci podręcznej (autorzy ARRG nazwali ją *fallback cache*), z której można odzyskać dane o starym, usuniętym węźle, gdy niepowiedzie się próba nawiązania komunikacji z nowym. Ponadto w ogólnym przypadku, komunikacja nie wymaga, aby węzły potwierdzały fakt odebrania wiadomości. Wprowadzony został także autorski mechanizm wyboru odbiorców wiadomości, szczegółowo opisany w sekcji 2.9.

Ogólna zasada działania zaimplementowanego algorytmu jest opisana w algorytmie 1.

Algorytm 1 Rozprzestrzenianie informacji przez węzeł

```
1:  loop
2:    for all wiadomość w do nadania do
3:      S ← losowy podzbiór znanych węzłów
4:      wyślij w do każdego elementu z S
5:    end for
6:    for all odebrana wiadomość w do
7:      przetwórz wiadomość w
8:      oznacz wiadomość w jako wiadomość do nadania
9:    end for
10: end loop
```

Rysunki 2.1a i 2.1b prezentują sposób rozprzestrzeniania informacji w sieci za pomocą algorytmu plotkującego. W pierwszej fazie (2.1a) węzeł W_0 wybiera ze zbioru znanych sobie



(a) (b)

Rysunek 2.1: Schemat działania algorytmu plotkującego. Źródło: opracowanie własne.

węzłów Z podzbiór S, do którego elementów wyśle swoją wiadomość. Wiadomość po wysłaniu dotarła do węzłów W_1 oraz W_2. Z powodu zawodności komunikacji nie otrzymał jej węzeł W_3. W następnej fazie (2.1b) węzły W_1 oraz W_2 wybierają w analogiczny sposób swoje zbiory odbiorców i przekazują do nich wiadomość otrzymaną od węzła W_0.

2.4. Model wirtualnego świata

Podczas przeprowadzania testów protokołu korzystano z sieci składających się nawet z kilkuset węzłów, niemożliwe zatem było wykorzystanie awatarów kontrolowanych przez ludzi. Ponadto dzisiejsze gry wieloosobowe nie zakładają jednoczesnego uczestnictwa takiej liczby graczy, więc nie oferują plansz stosownych rozmiarów. Natomiast próby wykorzystania dostępnych plansz skutkowały niespotykaną w normalnych warunkach gęstością zaludnienia wirtualnego świata, co prowadziło do zaburzenia wyników testów.

Wykorzystano zatem sztuczne plansze z losowo wygenerowanymi przeszkodami i awatarami zarządzanymi przez sztuczną inteligencję. Awatary poruszały się zgodnie z modelem aktywności opartym na diagramach Woronoja [17]. Diagramy konstruowano za pomocą algorytmu Fortune'a [11], gdzie punktami wejściowymi były wierzchołki przeszkód. Powstały diagram Woronoja wzbogacono wstawiając wierzchołki w miejscach gdzie krawędzie diagramu przecinały się z krawędziami przeszkód (co symbolizowało drzwi) lub z krawędziami planszy (aby wyeliminować nieskończone krawędzie).

Sztuczna inteligencja awatarów działa w sposób opisany w algorytmie 2.

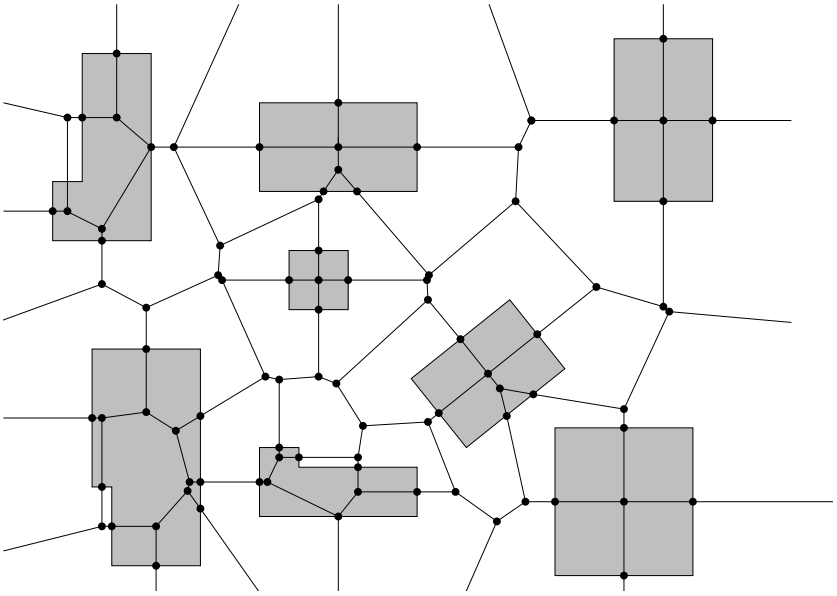
Algorytm 2 Sztuczna inteligencja awatara

```
loop
    w ← losowy wierzchołek grafu
    while awatar nie znajduje się w wierzchołku w do
        podążaj najkrótszą ścieżką do wierzchołka w
    end while
    pozostań przez losowy czas t w wierzchołku w
end loop
```

Kontrolowane przez sztuczną inteligencję awatary posiadają pewną swobodę w poruszaniu

się i dopuszczalne było, aby oddalały się nieznacznie od wyznaczonej ścieżki.

Bharambe *et al.* wykazali, że w przypadku gier FPS model ten bardzo dobrze odpowiada rzeczywistości [5].



Rysunek 2.2: Model planszy wraz z diagramem Woronoja. Źródło: opracowanie własne.

Rysunek 2.2 prezentuje losowo wygenerowaną planszę. Przeszkody oznaczone są na szaro. Na rysunku widoczne są węzły diagramu, które podczas rozgrywki staną się punktami orientacyjnymi dla sztucznej inteligencji. Natomiast krawędzie diagramu staną się ścieżkami, po których będą poruszać się kontrolowane przez nią awatary.

Możliwe jest rozszerzenie modelu poprzez przypisanie wag do wierzchołków grafu, odzwierciedlających wartość taktyczną posiadaną przez różne miejsca na planszy. Dzięki temu ruch awatarów na planszy staje się bardziej celowy i bliższy zachowaniom prawdziwych graczy.

2.5. Struktura protokołu

W ramach pracy opracowano i zaimplementowano protokół służący do wymiany wiadomości pomiędzy uczestnikami rozgrywki. Przesyłane wiadomości składają się z trzech części: nagłówka, danych dotyczących rozgrywki oraz danych dotyczących protokołu.

2.5.1. Identyfikatory typów wiadomości

Każdy typ wiadomości posiada swój unikatowy numer, który go jednoznacznie identyfikuje. Wiadomości są przetwarzane przez odbiorcę pole po polu, zatem niezmienne podawanie typu wiadomości na jej początku pozwala odbiorcy poprawnie zinterpretować otrzymane dane. Wykorzystywane przez Silac identyfikatory wiadomości zostały opisane w tabeli 2.1.

Prośba o dołączenie do rozgrywki	0
Akceptacja prośby o dołączenie do rozgrywki	1
Odrzucenie prośby o dołączenie do rozgrywki	2
Opuszczenie rozgrywki	3
Prośba o retransmisję danych o uczestniku	64
Zmiana stanu awatara	9
Pełna informacja o interakcjach	10
Informacja o aktywnych rozgrywkach	32
Informacja o aktywnej rozgrywce	33
Prośba o dodatkowe informacje o rozgrywce	34
Sygnał życia	35

Tabela 2.1: Identyfikatory typów wiadomości

2.5.2. Nagłówek wiadomości

Schemat nagłówka wiadomości przedstawia rysunek 2.3.

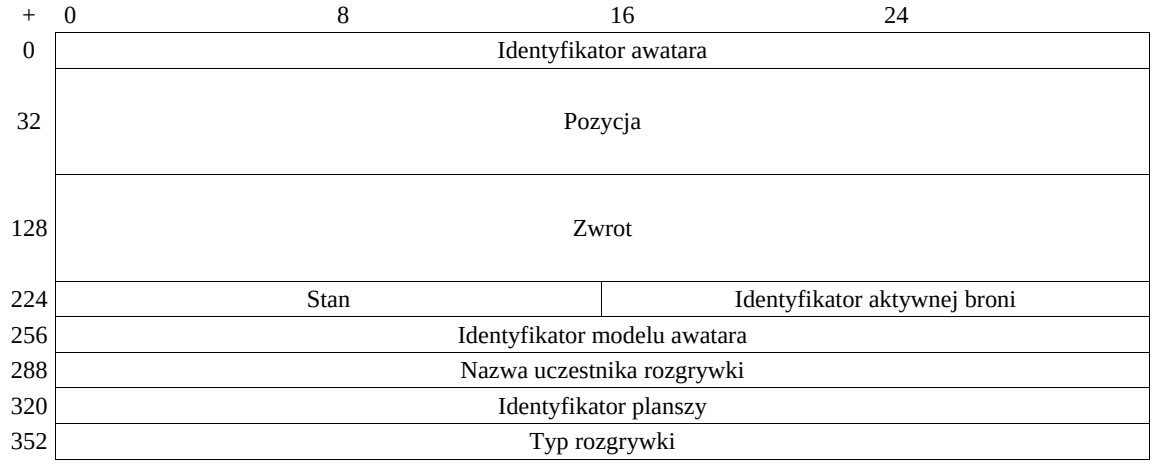
+	0	8	16	24
0	Identyfikator typu	Skok	Port nadawcy	
32	Identyfikator rozgrywki			
64	Numer wiadomości			
96	Numer ostatniej otrzymanej od odbiorcy wiadomości			
128	Rozmiar danych			

Rysunek 2.3: Nagłówek wiadomości

- Identyfikator wiadomości, zgodnie z tabelą 2.1 określa rodzaj i rozmieszczenie danych, które znajdują się za nagłówkiem.
- Skok oznacza, którym nadawcą z kolei dla następujących po nagłówku danych jest obecny węzeł. Nadawca pierwotny ma zawsze skok równy 0.
- Port nadawcy oznacza port, na którym nadawca oczekuje na odpowiedź.
- Identyfikator rozgrywki to liczba wygenerowana przez pierwszego uczestnika rozgrywki, jednoznacznie ją identyfikująca.
- Numer wiadomości to wartość zegara logicznego pierwotnego nadawcy wiadomości.
- Rozmiar danych oznacza rozmiar tylko danych dotyczących rozgrywki, bez rozmiaru nagłówka i danych dotyczących protokołu.

2.5.3. Prośba o dołączenie do rozgrywki

Prośba o dołączenie do rozgrywki jest pierwszą wiadomością wysyłaną przez nowego uczestnika rozgrywki do pozostałych, znanych mu uczestników. Zawiera pełne dane dotyczące awatara dołączającego gracza.



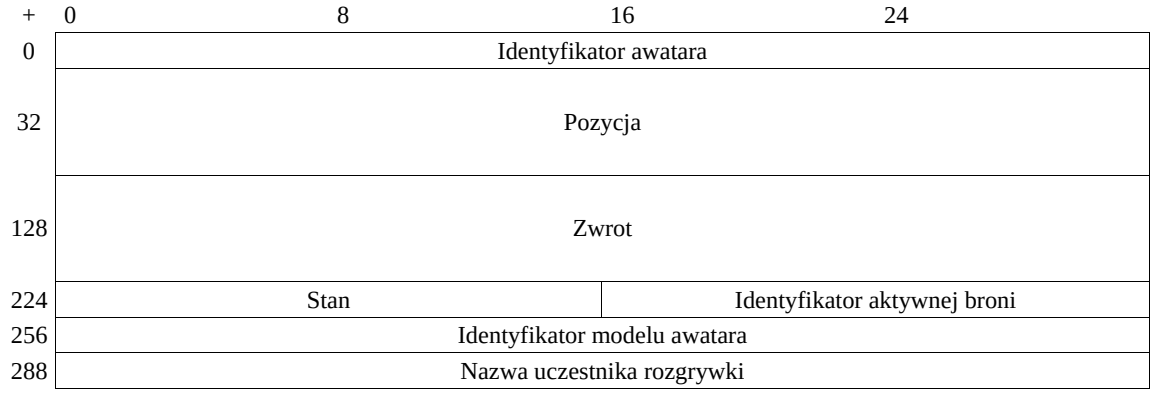
Rysunek 2.4: Prośba o dołączenie do rozgrywki

- Identyfikator awatara jest wyliczany na podstawie adresu oraz portu uczestnika rozgrywki, który jest jego właścicielem.

Poza pełną informacją o awatarze, wiadomość zawiera także adres i port pierwotnego nadawcy. Dzięki temu, niezależnie od liczby skoków, jakie wykona wiadomość, każdy odbiorca może odesłać odpowiedź bezpośrednio do pierwotnego nadawcy. Jest to ważne ograniczenie ilości danych przesyłanych w sieci, ponieważ odpowiedź na prośbę o dołączenie do rozgrywki jest nieistotna dla węzłów, które już znajdują się w rozgrywce.

2.5.4. Akceptacja prośby o dołączenie do rozgrywki

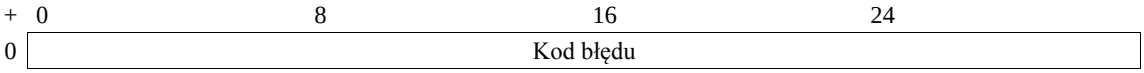
Wiadomość wysyłana przez każdy węzeł po otrzymaniu i zaakceptowaniu prośby o dołączenie do rozgrywki.



Rysunek 2.5: Akceptacja prośby o dołączenie do rozgrywki

2.5.5. Odrzucenie prośby o dołączenie do rozgrywki

W przypadku otrzymania przez uczestnika rozgrywki niepoprawnej prośby o dołączenie do rozgrywki, w odpowiedzi zostaje wysłana wiadomość o odrzuceniu prośby.



Rysunek 2.6: Odrzucenie prośby o dołączenie do rozgrywki

Stosowane w chwili obecnej kody błędu są opisane w tabeli 2.2.

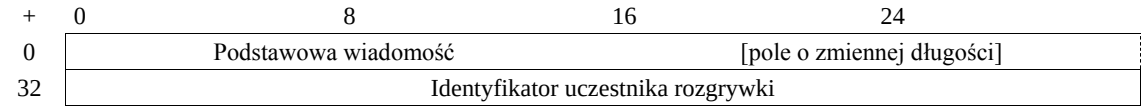
Niepoprawne dane dotyczące awatara	1
Awatar poza obszarem planszy	2
Niepoprawne dane dotyczące rozgrywki	3
Podanie nazwy używanej przez innego uczestnika	4
Niepoprawny adres nadawcy	5

Tabela 2.2: Kody błędu wiadomości

2.5.6. Opuszczenie rozgrywki

Ostatnia wiadomość wysyłana przez każdego uczestnika rozgrywki. Nie przekazuje żadnych danych.

2.5.7. Prośba o retransmisję danych o uczestniku



Rysunek 2.7: Prośba o retransmisję danych o uczestniku

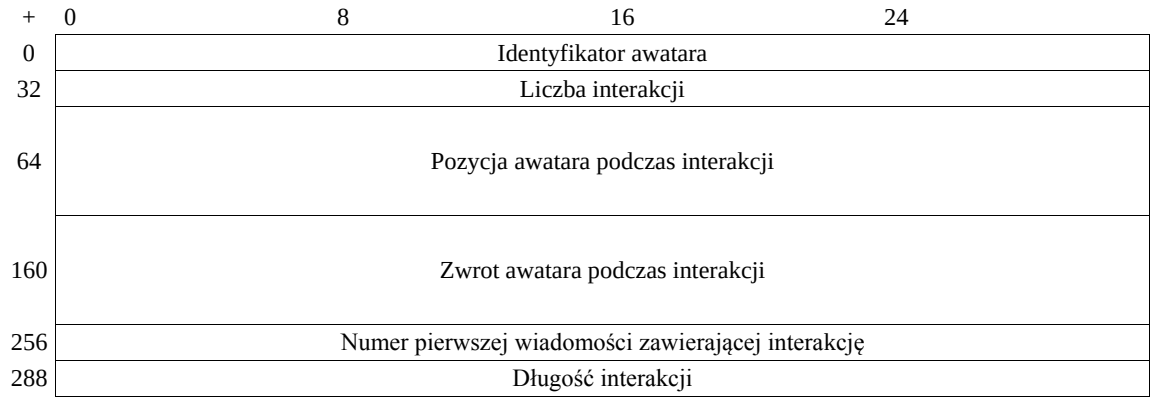
Pełna informacja o uczestnikach rozgrywki przesyłana jest pomiędzy węzłami tylko bezpośrednio po dołączaniu nowego uczestnika. Jest to naturalną konsekwencją jej dużych rozmiarów oraz niezmienności wielu jej elementów. Ze względu na zawodność komunikacji wprowadzono wiadomość 2.5.7, która jest wykorzystywana, gdy węzeł wykryje, że nie posiada informacji o jednym z uczestników.

Węzeł ustala, że nie posiada informacji o jednym z uczestników na podstawie danych dotyczących protokołu (por. wiadomość 2.5.10) zawartych w wiadomości otrzymanej od innego węzła. Następnie wysyła do tego węzła prośbę o retransmisję brakujących danych. W odpowiedzi otrzymuje od niego akceptację prośby o dołączenie do rozgrywki (por. wiadomość 2.5.4).

Prośba o retransmisję danych o uczestniku jest zawsze dołączana na końcu innej wiadomości. Typ takiej wiadomości równy jest koniunkcji logicznej typów wiadomości podstawowej i prośby o retransmisję.

2.5.8. Informacja o interakcjach

Wiadomość zawierająca informacje o interakcjach, w jakie wszedł awatar uczestnika rozgrywki.



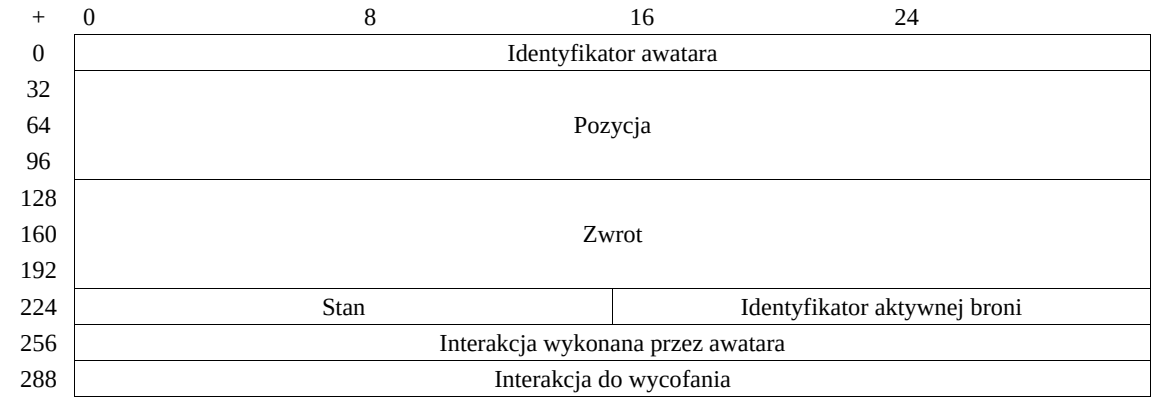
Rysunek 2.8: Informacja o interakcjach

Wiadomość ta wysyłana jest w przypadku utraty pakietu zawierającego dane o istotnej z punktu widzenia rozgrywki interakcji. Najczęściej jest to interakcja, która skutkuje zmianą w wyglądzie wirtualnego świata. Interakcje takie jak oddanie strzału, nie są ponownie przesyłane.

- Długość interakcji oznacza liczbę klatek, w których informacja o niej była wysyłana przez uczestnika rozgrywki.

2.5.9. Informacja o awatarze

Informacja o awatarze jest najczęściej wysyłaną przez uczestników wiadomością.



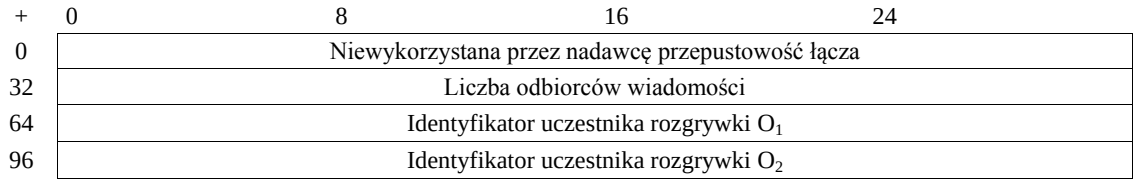
Rysunek 2.9: Informacja o awatarze

Zawiera informacje o nowym stanie awatara oraz wykonanych w ostatniej klatce interakcji. W przypadku wystąpienia konfliktu (por. sekcja 2.7) przesyła również dane o wycofanej

interakcji.

2.5.10. Dane dotyczące protokołu

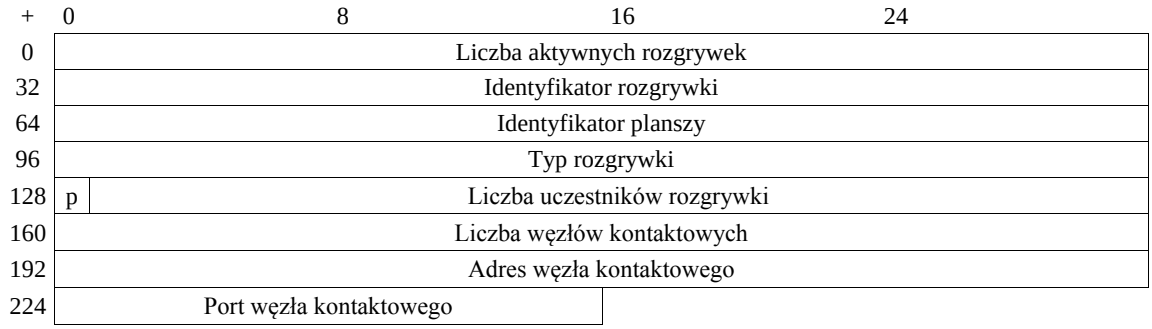
Każda wysyłana wiadomość posiada ogon, który zawiera informacje systemowe o protokole. Zgodnie z ideą algorytmów plotkujących jest to lista innych uczestników rozgrywki – potencjalnych odbiorców kolejnych komunikatów. W protokole zawężamy jednak zbiór uczestników, jacy mogą znaleźć się na liście: są to zawsze węzły, do których dana wiadomość była wysłana. Część wiadomości zawierającą dane dotyczące protokołu będziemy nazywać jej ogonem.



Rysunek 2.10: Dane dotyczące protokołu

2.5.11. Informacja o aktywnych rozgrywkach

Zdecydowano się wprowadzić do protokołu elementy architektury scentralizowanej. Wiadomości 2.5.11 – 2.5.14 są wymieniane pomiędzy uczestnikami rozgrywki a niebiorącym w niej udziału serwerem informacyjnym. Serwer ten ma za zadanie ułatwiać dostęp do informacji o rozgrywkach, a w dalszej perspektywie, umożliwiać zbieranie statystyk o uczestnikach i zwiększać bezpieczeństwo protokołu. Wiadomości 2.5.11 – 2.5.14 wykorzystują standardowy nagłówek protokołu (por. nagłówek 2.5.2), ale nie następują po nich dane dotyczące protokołu (por. wiadomość 2.5.10).



Rysunek 2.11: Informacja o aktywnych rozgrywkach

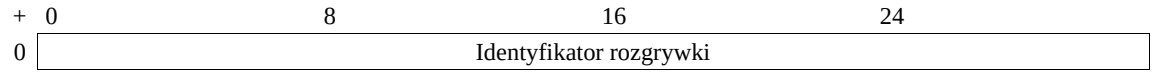
Bit p oznacza wersję protokołu internetowego, z którego korzysta dana rozgrywka. Gdy p jest równy 1, jest to wersja szósta, natomiast 0 oznacza wersję czwartą. Naturalnie pociąga to za sobą zmienną długość powyższej wiadomości.

2.5.12. Informacja o aktywnej rozgrywce

Wiadomość ma identyczną strukturę jak informacja o aktywnych rozgrywkach (por. wiadomość 2.5.11), przy czym zawsze ustawia na 1 wartość pola zawierającego liczbę aktywnych rozgrywek.

2.5.13. Prośba o dodatkowe informacje o rozgrywce

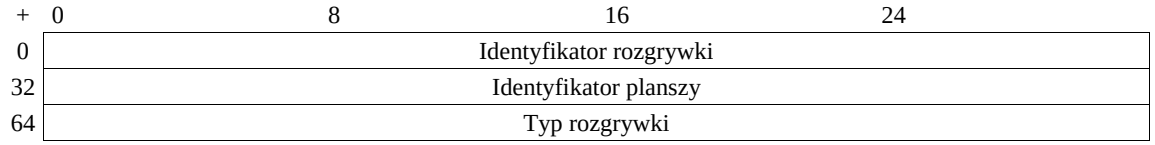
Wiadomość wysyłana przez węzeł, w celu pozyskania informacji o większej liczbie węzłów kontaktowych, co pozwoli na szybsze i płynniejsze dołączenie do rozgrywki.



Rysunek 2.12: Prośba o dodatkowe informacje o rozgrywce

2.5.14. Sygnał życia

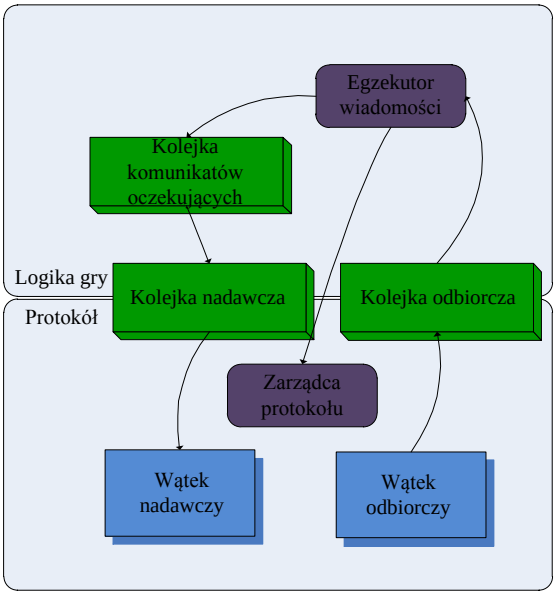
Wiadomość wysyłana przez uczestnika rozgrywki do serwera, informująca, że rozgrywka nadal jest aktywna oraz dany uczestnik nadal jest jej członkiem. Wiadomość zawiera tylko informację o rozgrywce oraz planszy, na której się ona toczy. Wszystkie pozostałe, potrzebne serwerowi informacje znajdują się w nagłówku 2.5.2. Możliwe jest wyłączenie wysyłania sygnału życia w celu zmniejszenia liczby wysyłanych wiadomości.



Rysunek 2.13: Sygnał życia

2.6. Architektura

W ramach pracy zaimplementowano warstwę pośredniczącą pomiędzy logiką gry a przedstawionym w sekcji 2.5 protokołem. Ta warstwa odpowiada za przetwarzanie, przygotowywanie i wysyłanie wiadomości. Wykorzystuje dwa pomocnicze wątki: nadawczy i odbiorczy, z którymi wątek główny, zarządzający logiką rozgrywki, komunikuje się za pomocą dwóch kolejek wiadomości (por. rysunek 2.14). Wiadomości wychodzące są umieszczane w kolejce nadawczej, po czym każda z nich zostaje wysłana do swojego zbioru odbiorców. Wiadomości przychodzące są pobierane z kolejki odbiorczej i przetwarzane przez egzekutora. Egzekutor decyduje o przyjęciu bądź odrzuceniu wiadomości. W przypadku przyjęcia wiadomości, nagłówek oraz informacje o protokole w niej zawarte są przekazywane do zarządcy protokołu. Pozostałe informacje są aplikowane do świata rozgrywki. Sama wiadomość jest umieszczana w kolejce wiadomości oczekujących na powtórne wysłanie.



Rysunek 2.14: Schemat architektury protokołu. Źródło: opracowanie własne.

Wątek nadawczy odpowiada za rozsyłanie wiadomości skonstruowanych w warstwie logiki rozgrywki do znanych odbiorców. Ponadto wątek nadawczy dołącza do wiadomości jej nagłówek oraz ogon (por. wiadomości 2.5.2 i 2.5.10).

Zbiór odbiorców jest pozyskiwany od zarządcy protokołu. Wątek odbiorczy przekazuje otrzymane wiadomości do kolejki odbiorczej. Z powodu ograniczonych zasobów gniazd sieciowych, niezwykle łatwo jest przepełnić ich kolejki i doprowadzić do utraty wszystkich następujących po tym wiadomości, zatem wątek odbiorczy wykonuje tylko rozczłonkowanie wiadomości na nagłówek, dane i ogon, przed przekazaniem ich do wątku głównego.

Kolejki odbiorcza i nadawcza służą do komunikacji pomiędzy wątkiem głównym a wątkami, odpowiednio, odbiorczym i nadawczym. Kolejka wiadomości oczekujących przetrzymuje odebrane i już przetworzone wiadomości, które uczestnik rozgrywki powinien przesłać dalej, do kolejnych uczestników.

Zarządca protokołu przechowuje informacje o pozostałych uczestnikach rozgrywki oraz wybiera spośród nich tych, którzy staną się odbiorcami danej wiadomości.

2.7. Oryginały i repliki

W naturalny sposób każdego z uczestników rozgrywki najbardziej interesuje otoczenie najbliżej ich awatara. Pozyskiwanie informacji o nim w rozproszonym środowisku może odbywać sie aktywnie, jak w przypadku projektu Colyseus, bądź pasywnie, jak w protokole Donnybrook. Oba systemy zakładały, że istnieje tylko jedna kopia główna każdego elementu wirtualnego świata oraz posiada ją dokładnie jeden uczestnik rozgrywki i w żadnym momencie nie zmienia ona swojego właściciela.

Protokół Silac posiada słabsze założenia. Przede wszystkim każdy uczestnik jest właścicielem całego wirtualnego świata i wszystkich elementów, które się w nim znajdują, za wyjątkiem awatarów innych uczestników rozgrywki. W ten sposób wirtualny świat jest współdzielony pomiędzy uczestnikami i każdy z nich odpowiada za utrzymywanie spójności pomiędzy swoim obrazem świata, a obrazami innych uczestników. Uspójnianie obrazów jest wykonywane za

pomocą otrzymywanych informacji o interakcjach awatarów z ich otoczeniem. Oznacza to, że uczestnicy rozgrywki muszą aktywnie rozgłaszać informacje tylko o swoich awatarach, ignorując informacje o innych dynamicznych elementach świata. Informacje o awatarach innych uczestników są rozsyłane tylko w ramach przekazywania informacji kolejnym węzłom. Powyższa strategia oznacza rozproszenie obliczeń dotyczących aktualizacji świata wśród uczestników. Naturalnie powoduje to kilku- bądź nawet kilkunastokrotne obliczanie tych samych wartości na różnych maszynach, ale w przypadku gier komputerowych jest to akceptowalny koszt zwiększenia limitu graczy w grze wieloosobowej. Zysk z takiej strategii wynika ze wspomnianego wcześniej (por. wykres 1.1) rozdźwięku między tempem wzrostu możliwości komputerów i konsol do gier a przepustowością łączy oraz z faktu, że opis skutku pojedynczej interakcji jest nie mniejszy niż opis interacji, która go wywołała.

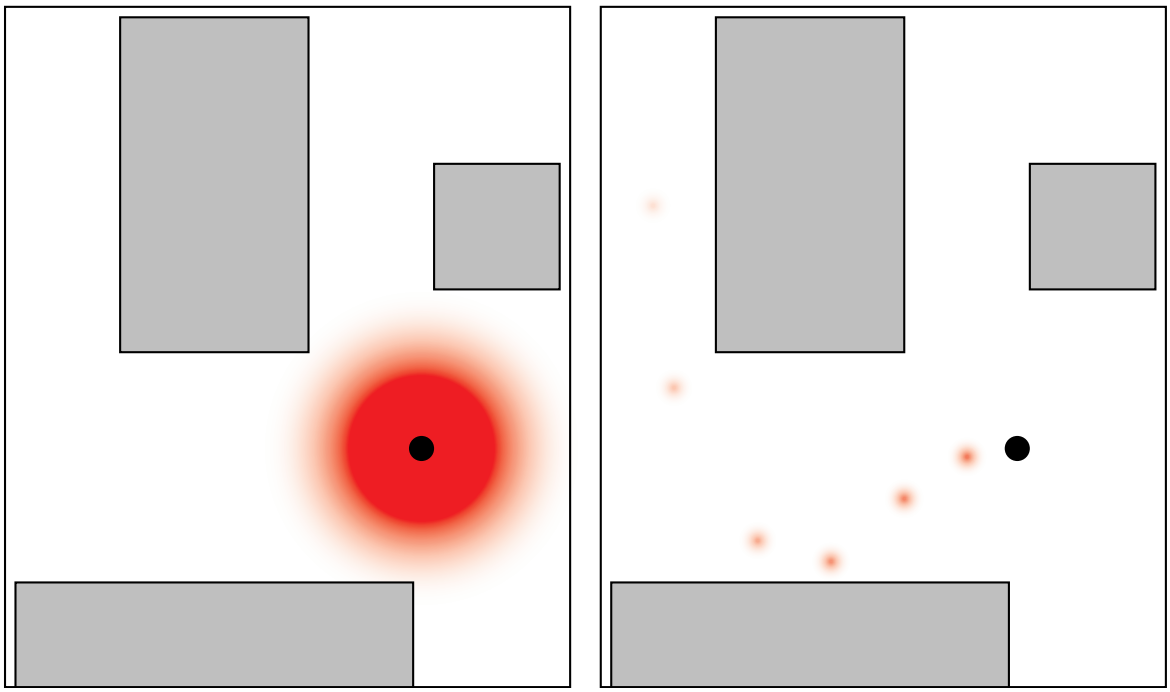
Przykład 1. Awatar zostaje uzbrojony w broń palną i oddaje jeden strzał w kierunku ściany. Interakcją w rozumieniu opracowanego protokołu jest strzał z aktywnej broni, a do jej pełnego opisu potrzebne są: pozycja oraz zwrot awatara, identyfikator broni oraz identyfikator akcji oddania strzału. Skutkiem interakcji jest ślad po pocisku na ścianie, który opisujemy za pomocą pozycji, rozmiaru i kształtu. Pomijamy powszechne już w dzisiejszych grach efekty wizualne takie jak dym unoszący się z lufy broni, czy błysk wystrzału. Jeżeli uwzględnimy fakt, że szczegółowe dane o awatarze muszą zostać rozesłane, bo tylko jeden uczestnik posiada jego kopię główną, opis interacji staje się istotnie mniejszy niż opis jej skutku.

Przykład 2. Awatar włącza ładunki wybuchowe, które doprowadzają do zburzenia budynku. Ponownie interację opisujemy za pomocą położenia i zwrotu awatara oraz identyfikatora akcji. Natomiast skutkiem jest zniknięcie całego budynku, które w terminach logiki gry oznacza zmianę położenia setek elementów wirtualnego świata.

Przyjmując powyższą koncepcję narażamy się na wystąpienie licznych niespójności w obrazach widzianych przez uczestników rozgrywki. Przede wszystkim może dojść do utraty pakietu. Zostały jednak zastosowane metody korygujące efekty jego wystąpienia (przesyłanie pełnych informacji). Jeżeli awatary uczestników rozgrywki znajdują się blisko siebie korekcja wszelkich błędów nastąpi w ciągu wymiany $3 - 4$ wiadomości, co oznacza czas poniżej $1/6$ sekundy. Poważniejsze z punktu widzenia rozgrywki skutki mogą wystąpić wtedy, gdy dwóch lub więcej uczestników rozgrywki dokonuje interacji z tym samym elementem. Każdy uczestnik jest właścicielem tego elementu i posiada jego kopię główną, co może doprowadzić do sytuacji, gdy np. dwa awatary podnoszą jednocześnie ten sam przedmiot. Wtedy następuje wycofanie ostatniej interakcji u wszystkich zainteresowanych uczestników oraz rozesłanie stosownego komunikatu. Z punktu widzenia gracza może wydawać się to słabym rozwiązaniem, ponieważ nagle traci efekt swoich działań. Mimo to posiada ono przewagę czasu wykonywania nad wszelkimi próbami uporządkowania wykonanych interakcji i podjęcia decyzji, który z graczy był pierwszy. Istotnie, obaj gracze są poszkodowani, jednak są poszkodowani natychmiast, podczas gdy w drugim przypadku jeden z nich byłby poszkodowany po upływie pewnego czasu, co mogłoby być dla niego nieoczekiwane i niezrozumiałe, a przez to frustrujące. Należy przy tym podkreślić, że, aby powyższy konflikt wystąpił, gracze muszą rozpocząć interakcję na tym samym przedmiocie w czasie pomiędzy dwoma kolejnymi komunikatami, bo znajdując się blisko siebie, będą prawie nieprzerwanie przesyłać sobie wiadomości. Zatem okno czasowe, którym dysponują wynosi ok. $1/25$ sekundy. W takich warunkach szanse wystąpienia konfliktu są minimalne.

2.8. Obszar zainteresowania i ścieżka zainteresowania

Niniejszy protokół do obszaru zainteresowania dodaje również ścieżkę zainteresowania. Ścieżką zainteresowania nazwiemy zbiór elementów wirtualnego świata, z którymi uczestnik rozgrywki wszedł w jakieś interakcje. Przy czym jest to zbiór z porządkiem liniowym, uporządkowanym względem czasu, jaki minął od zakończenia interakcji. Zatem interacja A jest większa od interakcji B, jeśli została przeprowadzona później. Naturalnie częściej są wysyłane informacje o interakcjach większych, ponieważ są one istotniejsze z punktu widzenia rozgrywki, bo zawierają bardziej aktualne dane. Ponadto informacje o mniejszych interakcjach zostały już wcześniej rozesłane, a w chwili obecnej są rozpowszechniane tylko w celu ewentualnego uspójnienia wirtualnego świata. Należy zaznaczyć, że interakcje ważne z punkty widzenia rozgrywki mogą być traktowane inaczej, to znaczy mogą mieć o wiele dłuższy czas zaniku ścieżki oraz mogą być traktowane jako większe od wszystkich pozostałych interakcji, zapewniając stały przepływ informacji o nich pomiędzy uczestnikami rozgrywki.



(a) (b)

Rysunek 2.15: Obszar i ścieżka zainteresowania. Źródło: opracowanie własne.

Rysunki 2.15a i 2.15b prezentują, odpowiednio, obszar oraz ścieżkę zainteresowania uczestnika rozgrywki. Awatar uczestnika zaznaczony jest na rysunkach na czarno. Zgodnie z wcześniejszą definicją ścieżki zainteresowania, interakcje tracą na ważności wraz z upływem czasu i ostatecznie zanikają.

2.9. Dobór odbiorców

W opisywanym protokole węzły nie poszukują aktywnie potrzebnych informacji, zatem istotną rolę odgrywa sposób rozprzestrzeniania wiadomości. Informacje jako pierwsze powinny otrzymać węzły najbardziej nią zainteresowane. W pracy wprowadzam następujący sposób

doboru odbiorców.

Węzły wybierane są losowo, przy czym przed każdym losowaniem obliczane jest prawdopodobieństwo wyboru każdego węzła. Prawdopodobieństwo wyboru jest ilorazem średniej ważonej trzech czynników oraz sumy wartości wszystkich węzłów. Zatem, gdy wartości węzłów obliczane są przez węzeł j:

$$v_{j,k} = \alpha_1 * d_{j,k} + \alpha_2 * a_{j,k} + \alpha_3 * b_k \tag{2.1}$$

$$p_{j,k} = \frac{v_{j,k}}{\sum_{i=1}^{N} v_{j,i}} \tag{2.2}$$

gdzie:

$d_{j,k}$	obliczone jak w równaniu 2.3,
$a_{j,k}$	obliczone jak w równaniu 2.4,
b_k	obliczone jak w równaniu 2.5.

Przy czym:

$$d_{j,k}, a_{j,k}, b_k \in [0, 1]$$

Po obliczeniu wartości i prawdopodobieństw wszystkich węzłów, wybierany jest ich podzbiór, którego elementy staną się odbiorcami danej wiadomości.

2.9.1. Odległość

Najistotniejsze dla węzła informacje to informacje o elementach świata znajdujących się blisko należącego do niego awatara. Stąd pierwszym czynnikiem składającym się na wartość węzła jest odległość między nim a węzłem wysyłającym wiadomość. Odległość definiuję według formuły 2.3.

$$d_{j,k} = 1 - \frac{\sqrt{(x_j - x_k)^2 + (y_j - y_k)^2 + (z_j - z_k)^2}}{D_{max}} \tag{2.3}$$

gdzie:

D_{max}	maksymalna dopuszczalna przez dany wirtualny świat odległość między jego dwoma elementami.

2.9.2. Funkcja celu

Kolejnym czynnikiem określającym, jak istotny jest dla węzła dany element jest jego pozycja względem zwrotu awatara należącego do węzła. Niech $\beta_{j,k}$ oznacza kąt, wyrażony w radianach, pomiędzy zwrotem awatara uczestnika j a wektorem pomiędzy awatarami uczestników j oraz k, wtedy funkcję celu definiuję jak w równaniu 2.4.

$$a_{j,k} = \exp(-\frac{\beta_{j,k}^2}{2}) \tag{2.4}$$

Potrzeba wprowadzenia funkcji celu wynika z faktu, że chociaż awatary uczestników rozgrywki poruszają się ze znaczną prędkością i rzadko pozostają w miejscu przez dłuższy czas, zazwyczaj koncentrują się tylko na obiektach przed sobą, znajdujących się w niewielkim odchyleniu od ich zwrotu. Ponadto awatary poruszają się zwykle w liniach prostych, zmieniając kierunek i zwrot najczęściej w wyniku napotkania przeszkody. Możliwe jest poprawienie powyższego warunku poprzez uwzględnienie widoczności obiektów. Jeżeli uczestnik rozgrywki

nie widzi danego obiektu, bo ten znajduje się za ścianą, to nie potrzebuje stałego napływu informacji o nim. Niestety, sprawdzanie widoczności obiektów spowodowałoby bardzo silną zależność protokołu od silnika graficznego gry, co ograniczyłoby jego przenośność.

2.9.3. Dostępne zasoby sieciowe

Algorytm plotkujący zakłada rozprzestrzenianie się informacji, gdy jedne węzły przekazują ją innym. Z tego powodu, w pierwszej kolejności informacja jest przekazywana węzłom, które posiadają najwięcej wolnych zasobów sieciowych. Wartość b_k jest wprost proporcjonalna do wartości wolnej przepustowości łącza fb_k przekazanej przez węzeł k w ostatnim otrzymanym od niego komunikacie, wyrażonej w bajtach.

$$b_k = \frac{fb_k}{fb_{max}} \tag{2.5}$$

gdzie:

fb_{max} największa wartość wolnej przepustowości łącza przekazana przez wszystkie znane węzły.

2.10. Przekazywanie informacji

Każdy węzeł ponad generowane przez siebie informacje, które rozprzestrzenia po sieci, musi przekazywać wiadomości otrzymane od innych węzłów. Schemat działania mechanizmu przekazywania informacji prezentuje rysunek 2.16.

Odebrawszy wiadomość węzeł umieszcza ją w kolejce oczekujących wiadomości na losowy czas, nie dłuższy niż 4 klatki. Wybór odbywa się zgodnie z rozkładem jednostajnym dyskretnym czteropunktowym (por. równanie 2.7). Po upływie założonego czasu wiadomość zostaje wysłana. W przypadku, gdy węzeł w ciągu tego czasu oczekiwania otrzyma identyczną wiadomość od dwóch różnych węzłów, wiadomość zostaje usunięta z kolejki bez wysyłania. Mechanizm ten ma na celu ograniczyć ilość komunikacji w sieci. Otrzymanie tej samej informacji z dwóch osobnych źródeł oznacza, że w danym otoczeniu wiadomość została już rozprowadzona i wysyłanie jej po raz kolejny nie przyniesie istotnej zmiany. Ponadto w przypadku, gdy węzeł nie otrzymał ponownie danej wiadomości lub otrzymał ją od tylko jednego innego węzła, wysyła ją modyfikując wagi węzłów (por. równanie 2.6).

Należy zaznaczyć, że nie wszystkie wiadomości podlegają ponownemu wysłaniu. Wiadomości 2.5.4 oraz 2.5.7 nie są przekazywane innym węzłom.

$$v'_{j,k} = v_{j,k} - \alpha_4 * s_k \tag{2.6}$$

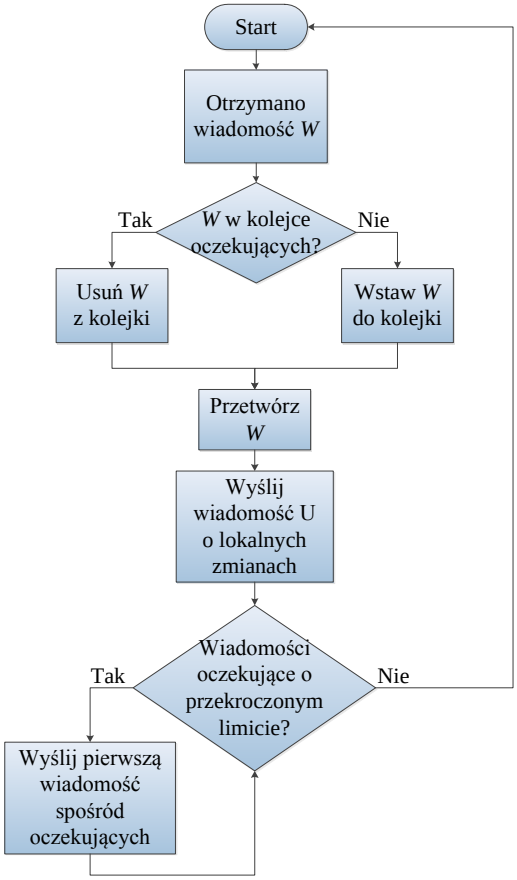
$$\forall i \in \{1, ..., 4\}\, \mathbb{P}(X = i) = 1/4 \tag{2.7}$$

gdzie:

$$s_k = \begin{cases} 1 & k \text{ był wymieniony jako odbiorca w ogonie danej wiadomości} \\ 0 & \text{wpp} \end{cases}$$

2.11. Utrata wiadomości i mechanizm korekcyjny

W ogólnym przypadku węzły nie potwierdzają odebrania wiadomości. Jest to spowodowane charakterem gier, dla których opracowano protokół. W sytuacji, gdy okno komunikacji wynosi



Rysunek 2.16: Schemat przetwarzania wiadomości. Źródło: opracowanie własne.

50 milisekund, oczekiwanie, nawet aktywne, na odpowiedź jest traceniem zasobów. Podobnie, ponieważ istnieje możliwość doprowadzenia do bardzo częstej wymiany wiadomości między dwoma węzłami (rzędu 15 pakietów na sekundę), wysyłanie potwierdzeń, nawet grupowych, spowodowałoby niepotrzebny ruch w sieci.

Nie oznacza to jednak, że węzły nie informują się o efektach swojej komunikacji. Służy do tego pole w nagłówku wiadomości "numer ostatniej otrzymanej od odbiorcy wiadomości" (por. nagłówek 2.5.2). Za pomocą tego pola nadawca wiadomości przekazuje jej przyszłemu odbiorcy, jaki był numer ostatniego komunikatu, który od niego otrzymał. Innymi słowy, jaki jest najnowszy, uspójniony między tymi dwoma węzłami obraz świata. Z punktu widzenia protokołu jest to najistotniejsza informacja. Dzięki niej węzeł wie, względem jakiego stanu wysyłać ewentualne dane uspójniające (por. wiadomość 2.5.8). Ponadto węzeł natychmiastowo potrafi określić, czy utracone zostały dane o istotnej z punktu widzenia wirtualnego świata interakcji, co wymusza dodatkową synchronizację dwóch obrazów świata.

2.12. Równoważenie obciążenia

Protokół wspiera równoważenie obciążenia poprzez wprowadzenie trzeciego czynnika do równania 2.1. Każdy węzeł wysyłając wiadomość, w jej nagłówku umieszcza połowę wartości posiadanej przez siebie wolnej przepustowości. To pozwala radzić sobie z sytuacjami, gdy

nagle węzeł zostanie wybrany do przekazania wiadomości przez wiele innych węzłów. Ponieważ zadeklarował tylko połowę swoich możliwości, najpewniej będzie w stanie obsłużyć nagły skok aktywności. Ponadto podczas następnej wymiany komunikatów przekaże innym o wiele mniejszą wartość, co znowu odciąży jego zasoby.

2.13. Czas życia wiadomości

Wiadomości wykorzystywane przez niniejszy protokół posiadają swoje czasy życia (*ang. time to live*). Jest to konieczny mechanizm zapobiegający kilkunastokrotnemu przysyłaniu jednej wiadomości przez kolejne węzły. W przypadku gier wieloosobowych, a przede wszystkim gier FPS, jest to spowodowane także szybką utratą aktualności, jakiej doświadczają komunikaty. Istotnie, w wirtualnym świecie sytuacja zmienia się średnio co 50 milisekund, zatem wiadomość, która jest wysyłana przez czwartego z kolei nadawcę, odnosi się do stanu gry sprzed co najmniej 200 milisekund. Jeżeli uwzględnimy także opóźnienia wynikające z przesyłania komunikatów przez sieć internetową, wiadomość może zawierać informacje sprzed pół sekundy. Nawet dla bardzo oddalonych od pierwotnego nadawcy uczestników rozgrywki jest to czas zbyt długi.

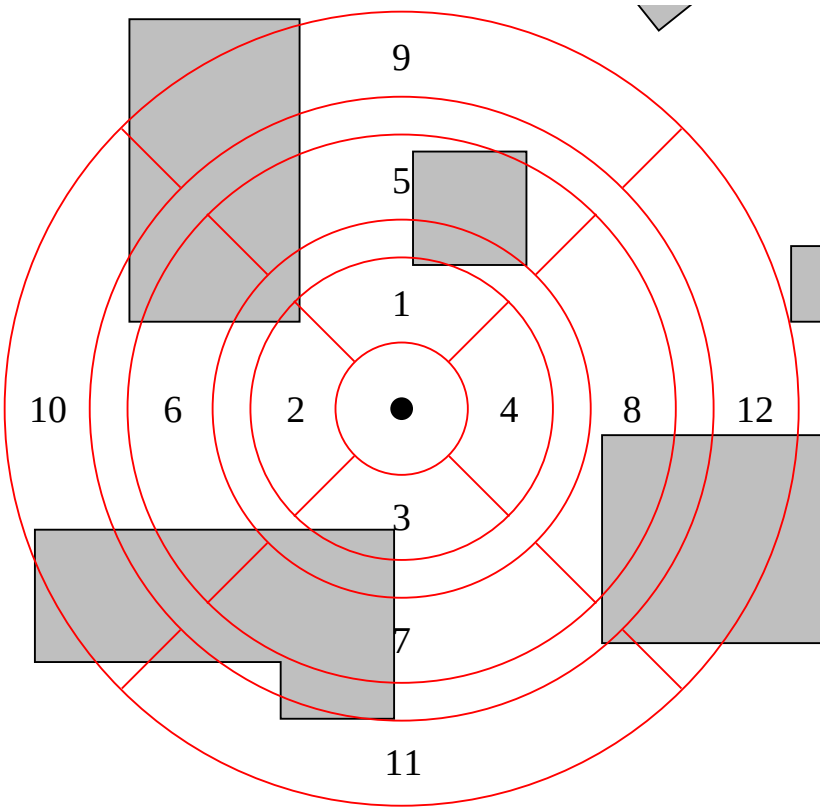
W celu wyeliminowania takich sytuacji wprowadzono do nagłówka wiadomości pole skok (por. nagłówek 2.5.2). Każda wiadomość może zostać wysłana tylko przez trzech kolejnych nadawców, co oznacza, że otrzymawszy wiadomość o wartości pola skok równej 2, węzeł już jej nie wyśle do kolejnych odbiorców. Dzięki temu ograniczamy ilość danych przesyłanych w sieci oraz nie pozwalamy węzłom zużywać swoich mocy na obsługiwanie zdezaktualizowanych komunikatów.

Rozdział 3

Weryfikacja poprawności i wydajności protokołu

3.1. Wyznaczanie parametrów protokołu

Przed przystąpieniem do właściwych testów protokołu Silac, przeprowadziłem cykl doświadczeń w celu wyznaczenia parametrów α_1, α_2, α_3 z równania 2.1 oraz α_4 z równania 2.6. Te cztery parametry determinują prawdopodobieństwo wyboru węzła jako odbiorcy wiadomości, a tym samym definiują sposób rozprzestrzeniania się danych po sieci. Intuicyjnie ich wartości powinny odzwierciedlać jak istotne są cztery cechy węzła: odległość i kąt między awatarami, wolna przepustowość łącza oraz wybór węzła podczas poprzedniej komunikacji.



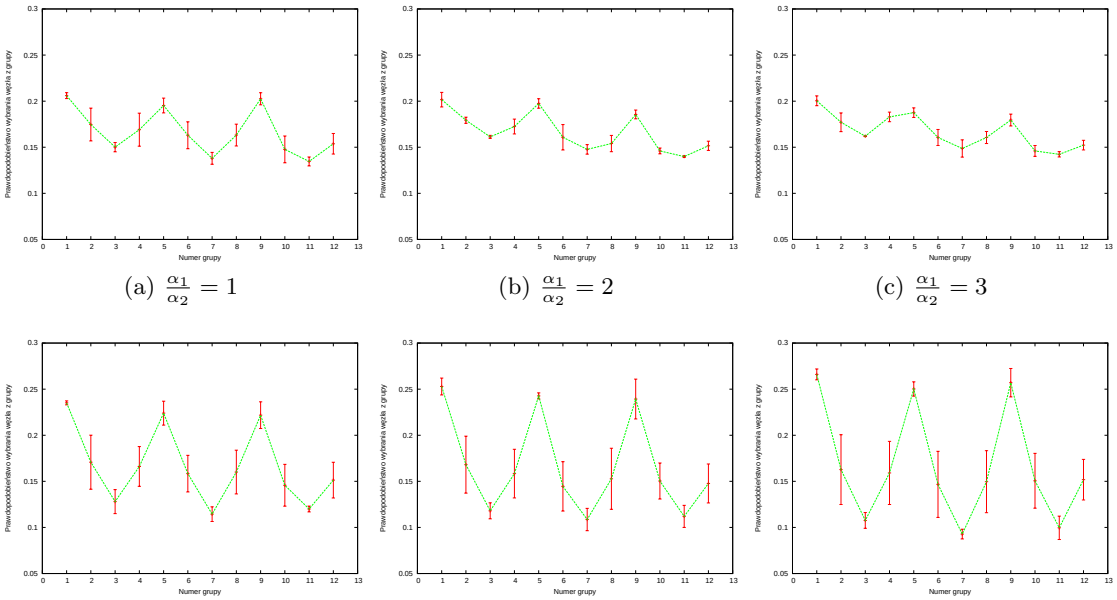
Rysunek 3.1: Rozmieszczenie grup węzłów. Źródło: opracowanie własne.

Eksperymenty 3.1.1 – 3.1.3 miały na celu zbadanie zależności pomiędzy wartościami parametrów a prawdopodobieństwem wyboru danego węzła. Znając je wybrałem ten zestaw parametrów, który najbardziej odpowiadał wymogom stawianym protokołowi.

W eksperymentach użyto siedemdziesięciu dwóch węzłów, podzielonych na dwanaście grup (por. rysunek 3.1). W każdej grupie znajdowało się siedem węzłów, rozmieszczonych równomiernie na całym jej obszarze. Ponadto istniał jeden wyróżniony węzeł, który będę nazywać **nadawcą**. Badano prawdopodobieństwo wybrania przez nadawcę jakiegokolwiek węzła z danej grupy przy ustalonym stosunku dwóch wybranych parametrów. Awatary należące do węzłów pozostawały nieruchome, aby prawdopodobieństwo ich wyboru było niezmienne.

3.1.1. Odległość i funkcja celu

W pierwszym etapie weryfikacji przeprowadziłem badanie zależności pomiędzy odległością a kątem między awatarami. Wraz ze wzrostem wartości stosunku $\frac{\alpha_1}{\alpha_2}$, odległość zaczyna dominować nad kątem między awatarami (por. wykresy 3.2b – 3.2c). Powoduje to, że węzły o podobnej odległości od nadawcy, ale istotnie różnym kącie, a zatem także widoczności, były wybierane równie często.



(a) $\frac{\alpha_1}{\alpha_2} = 1$ (b) $\frac{\alpha_1}{\alpha_2} = 2$ (c) $\frac{\alpha_1}{\alpha_2} = 3$

(d) $\frac{\alpha_1}{\alpha_2} = \frac{1}{2}$ (e) $\frac{\alpha_1}{\alpha_2} = \frac{1}{3}$ (f) $\frac{\alpha_1}{\alpha_2} = \frac{1}{4}$

Rysunek 3.2: Wyniki eksperymentu. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Podobnie wraz ze spadkiem wartości stosunku $\frac{\alpha_1}{\alpha_2}$, kąt zaczynał dominować nad odległością między awatarami (por. wykresy 3.2d – 3.2f). Wraz z tym węzły o podobnym kącie lecz różnych odległościach stawały się równie dobrymi odbiorcami wiadomości. Ponadto zaobserwowano znaczny wzrost wartości odchylenia standardowego dla węzłów o średnich kątach między swoimi awatarami a awatarem nadawcy. Jest to spowodowane wyborem funkcji wykładniczej w równaniu 2.4 i oznacza, że prawdopodobieństwo wyboru węzła z grup $2, 4, 6, 8, 10, 12$ przyjmuje wartości z bardzo dużego przedziału.

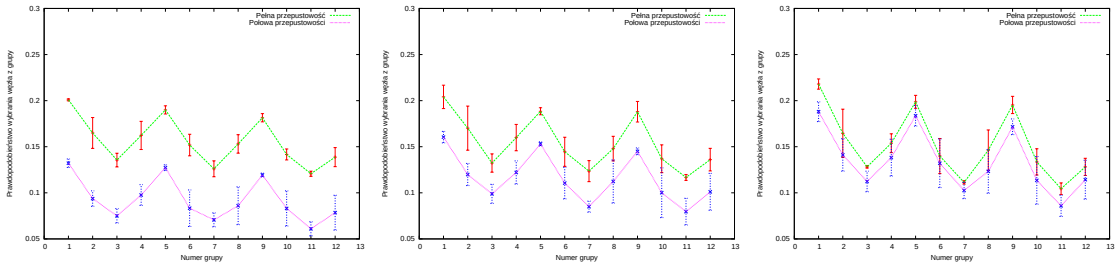
Przeprowadzone badania wykazały, że najbardziej potrzebom protokołu odpowiada stosunek $\frac{\alpha_1}{\alpha_2} = 1$. Różnicuje on węzły ze względu na odległość i kąt między ich awatarami, jednak

żadna z cech nie dominuje nad drugą. Prawdopodobieństwo wyboru węzła z grup naprzeciw nadawcy jest największe, a odchylenia standardowe są wystarczające małe, aby protokół zachowywał się w sposób przewidywalny.

3.1.2. Odległość i przepustowość sieci

W następnej kolejności przeprowadziłem badanie zależności pomiędzy odległością węzłów a deklarowaną przez nie wolną przepustowością sieci.

Podczas eksperymentu wprowadzono trzynastą grupę, liczącą, tak jak pozostałe, siedem węzłów. Obszar trzynastej grupy pokrywał się z obszarem jednej z pozostałych dwunastu z tym, że węzły trzynastej grupy deklarowały przepustowość o połowę mniejszą niż wszystkie pozostałe. Dla każdej wartości stosunku parametrów α_1 i α_3 przeprowadzono dwanaście rozgrywek, w każdej trzynasta grupa pokrywała się z inną z dwunastu podstawowych grup.



(a) $\frac{\alpha_1}{\alpha_3} = 1$ (b) $\frac{\alpha_1}{\alpha_3} = 2$ (c) $\frac{\alpha_1}{\alpha_3} = 4$

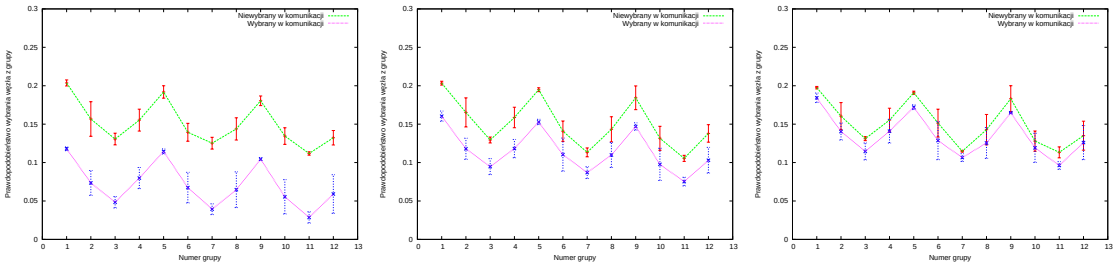
Rysunek 3.3: Wyniki eksperymentu. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Gdy parametr α_3 był duży w porównaniu z parametrem α_1 zaobserwowano znaczne zmniejszenie prawdopodobieństwa wyboru węzła (por. wykres 3.3a), tak że najważniejsze z punktu widzenia rozgrywki węzły z grupy pierwszej stawały się najmniej istotne. Uznano to za niepożądane, bo informacja powinna w pierwszej kolejności docierać do węzłów najbliżej nadawcy. Z drugiej strony, gdy parametr α_3 był mały, nie wprowadzał żadnej istotnej zmiany (por. wykres 3.3c). W takim przypadku informacja o wolnej przepustowości sieci stawała się zbędna. Wobec czego zdecydowano się wybrać parametry, takie że $\frac{\alpha_1}{\alpha_3} = 2$.

3.1.3. Ponowny wybór węzła do komunikacji

Idealnym rozwiązaniem byłoby, gdyby czwarty parametr α_4 sprowadzał prawdopodobieństwo wyboru węzła do wartości prawdopodobieństwa utraty pakietu do niego adresowanego. Możliwe jest wyliczanie odsetka pakietów, które dotarły do danego węzła, ale ze względu na heterogeniczność sieci ta informacja będzie poprawna tylko dla jednego określonego nadawcy, wymuszając wymianę jej pomiędzy węzłami podczas komunikacji. Zdecydowano się zastosować mechanizm mniej dokładny, lecz prostszy i nie wymagający przesyłania dodatkowych danych. Prawdopodobieństwo wyboru węzła, który był wymieniony w ogonie wiadomości (por. wiadomość 2.5.10) jest zmniejszane o wartość α_4.

Przeprowadzony eksperyment opierał się na tych samych założeniach, co eksperyment 3.1.2. Przy czym węzły dodatkowej, trzynastej grupy nie deklarowały zmniejszonej przepustowości, tylko były uznawane za węzły, które już zostały wybrane do otrzymania danej wiadomości. Ponownie, trzynasta grupa w dwunastu kolejnych eksperymentach pokrywała kolejne z dwunastu podstawowych grup.



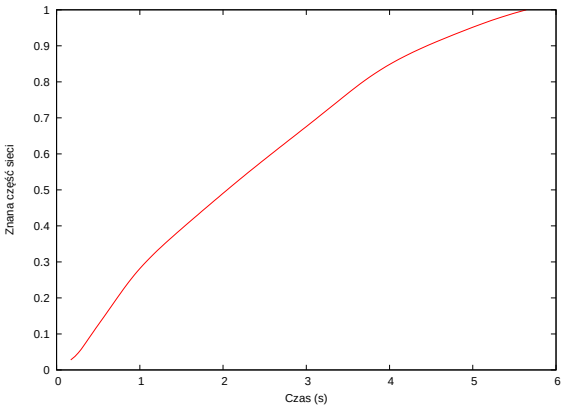
(a) $\frac{\alpha_1}{\alpha_4} = 1$ (b) $\frac{\alpha_1}{\alpha_4} = 2$ (c) $\frac{\alpha_1}{\alpha_4} = 4$

Rysunek 3.4: Wyniki eksperymentu. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

W przypadku parametry α_4 zdecydowano, że jego małe wartości nie wprowadzają pożądanych zmian kolejności, w jakiej wybierane są węzły (por. wykresy 3.4b i 3.4c). Oznacza to, że wybrano taki parametr α_4, że $\frac{\alpha_1}{\alpha_4} = 1$, gdyż wtedy prawdopodobieństwo wyboru węzła do ponownej komunikacji jest mniejsze niż prawdopodobieństwo wyboru węzła, który danej wiadomości jeszcze nie otrzymał.

3.2. Szybkość rozprzestrzeniania się danych

Wyznaczywszy wartości parametrów protokołu, przeprowadziłem badania cech protokołu za pomocą testów wydajnościowych. Pierwszym testem była szybkość rozprzestrzeniania się danych po sieci. W eksperymencie wykorzystano sieć składającą się ze stu węzłów, z których każdy uczestniczył w komunikacji przez co najmniej minutę. Dzięki temu każdy węzeł znał całą lub przynajmniej większość sieci. Następnie dołączono do sieci nowy węzeł, który znał tylko jeden węzeł należący do sieci i tylko z nim się kontaktował. Wykres 3.6 prezentuje procent znanej nowemu węzłowi sieci w zależności od czasu, przez jaki uczestniczył w komunikacji. Przy czym znajomość węzłów jest relacją symetryczną: jeśli nowy węzeł zna 50% węzłów z sieci, to te węzły znają jego.

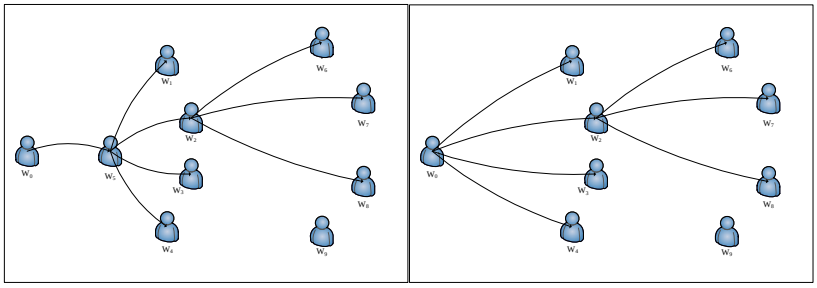


Rysunek 3.5: Zależność rozmiaru znanej części sieci od czasu. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Wyniki eksperymentu są zadowalające, bo średnio w ciągu 5.5 sekundy informacja o nowym węźle docierała do wszystkich pozostałych. Jeśli wzięty pod uwagę będzie fakt, że plan-

sza, na której toczy się rozgrywka musi zostać załadowana do pamięci podręcznej, co, nawet w komercyjnych produktach, trwa pewien czas, okazuje się, że nowy uczestnik przystępuje do rozgrywki wieloosobowej znając już wszystkich pozostałych.

Warto zauważyć, że zwiększenie liczby znanych nowemu węzłowi węzłów z sieci nie zmniejsza czasu rozprzestrzeniania się danych. Spowodowane jest to faktem, że większa liczba znanych na początku węzłów pozwala zaoszczędzić tylko jeden skok wiadomości, czyli ok. 70 – 100 milisekund (por. rysunek 3.6).

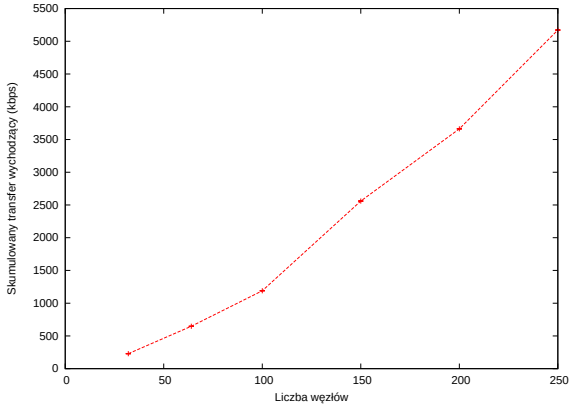


Rysunek 3.6: Schemat przekazywania informacji podczas dołączania do rozgrywki. Źródło: opracowanie własne.

Eksperyment pomógł równocześnie ustalić limit na czas braku komunikacji pomiędzy węzłami. Jeśli węzeł nie otrzyma żadnych informacji o innym przez co najmniej 10 sekund, to wymaże go ze swojej pamięci.

3.3. Ilość wysyłanych danych

W sekcji 1.5 ustalono, że architektura scentralizowana wymaga znacznego transferu danych pomiędzy serwerem a klientami. Architektura rozproszona potrzebuje mniejszego transferu danych w przeliczeniu na węzeł. Przy czym sumaryczny transfer danych jest znacznie większy.



Rysunek 3.7: Zależność średniego transferu danych od liczby węzłów. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Wykres 3.7 prezentuje średni transfer danych wychodzący z jednego węzła wraz z odchyleniem standardowym w zależności od liczby węzłów biorących udział w rozgrywce. Odchylenie standardowe jest bardzo małe (rzędu 0.1%), co jest spowodowane faktem, że znaczna część

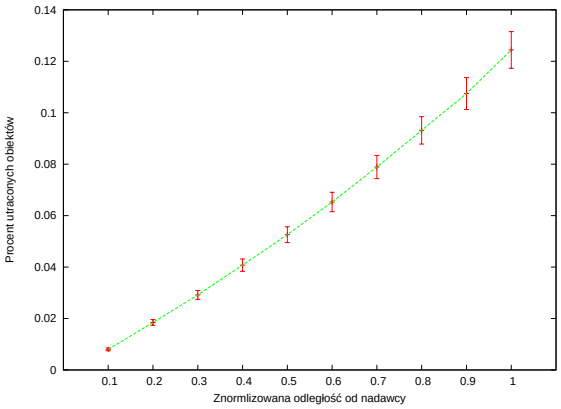
(ponad 90%) wychodzących wiadomości to wiadomości o stanie awatara. Protokół Silac oferuje liniową zależność ilości wysyłanych danych od liczby uczestników rozgrywki, co oznacza, że jest lepszy niż jakiekolwiek rozwiązanie scentralizowane.

Dla każdej badanej liczby uczestników rozgrywki przeprowadzono 5 rozgrywek. Każda z rozgrywek trwała 12 minut, przy czym pierwszych dwóch minut nie brano pod uwagę podczas obliczania skumulowanego transferu wychodzącego. Dzięki temu faza wymiany pierwszych komunikatów między węzłami: próśb o dołączenie do rozgrywki i ich akceptacji, kiedy węzły nie posiadają pełnej informacji o świecie nie wpływała na ostateczny wynik.

3.4. Ilość zgubionych informacji

W ocenie jakości protokołu sieciowego gry, niemniej ważną od ilości wysyłanych danych jest informacja, jaka część tych danych dociera na czas. Wykres 3.8 prezentuje procentowy udział obiektów, o których informacje badany węzeł utracił w ciągu całej rozgrywki, w zależności od ich odległości od niego.

Badanie przeprowadzono na stu węzłach. Obiektami w rozumieniu niniejszego badania, były wszystkie interakcje awatarów z wirtualnym światem.



Rysunek 3.8: Dystrybuanta liczby utraconych obiektów w zależności od odległości od nadawcy. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

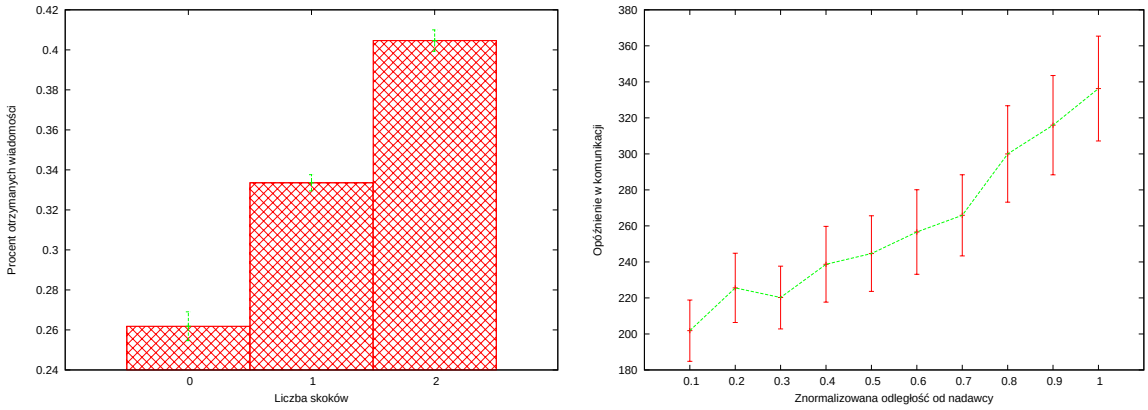
Wykres 3.8 pokazuje, że w najbliższym sąsiedztwie wybranego węzła (do $\frac{1}{5}$ rozmiaru planszy utracono nie więcej jak 2% informacji. Ponadto zaobserwowano niewielkie odchylenie standardowe (ok. 5%).

3.5. Liczba skoków wiadomości

Wiedząc, że protokół doprowadził do utraty informacji tylko o niewielkiej części obiektów w wirtualnym świecie, zbadano średnie opóźnienie z jakim węzeł otrzymywał informacje.

W eksperymencie badano ile skoków wykonała wiadomość zanim została odebrana przez badany węzeł oraz ile czasu średnio zajmowało jej dotarcie do niego w zależności od odległości awatarów nadawcy i odbiorcy. Ponownie w eksperymencie brało udział sto węzłów.

Wykres 3.9a pokazuje, że najwięcej odebranych wiadomości odbyło już dwa skoki i w ten sposób zakończyło swoją podróż w badanych węźle. Jedna trzecia wiadomości miała za sobą jeden skok. Zaobserwowano również niewielkie odchylenie standardowe (poniżej 5%).



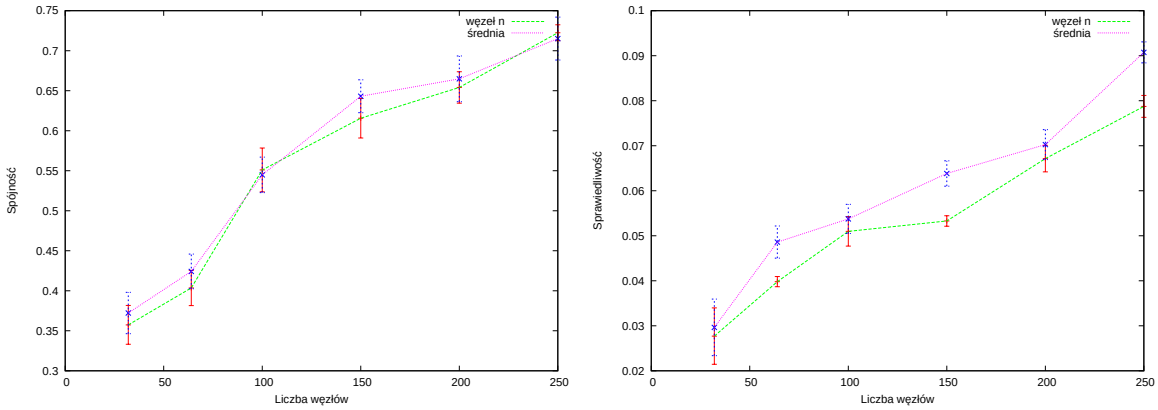
(a) (b)

Rysunek 3.9: Wartość opóźnień otrzymanych wiadomości. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Znaczna liczba skoków, jakie wykonała wiadomość przed odebranie przekłada się na spore opóźnienia dla obiektów znajdujących się w dużej odległości od badanego węzła (por. wykres 3.9b). Opóźnienie rzędu 300 milisekund byłoby nieakceptowalne w grach FPS [3], jednak, skoro dotyczy ono obiektów znajdujących się po drugiej stronie planszy, których uczestnik rozgrywki nie widzi i z którymi nie wchodzi w interakcje, można je uznać za zadowalające. Także opóźnienia dla niewielkich odległości są wystarczająco małe, aby uczestnik rozgrywki ich nie zauważał.

3.6. Spójność i sprawiedliwość

W sekcji 1.4 zdefiniowano dwie metody oceny jakościowej protokołów sieciowych stworzonych dla gier wieloosobowych. Wykresy 3.10a oraz 3.10b prezentują wyniki dla pojedynczego węzła oraz średni wynik dla wszystkich węzłów.



(a) (b)

Rysunek 3.10: Zależność spójności i sprawiedliwości od liczby węzłów. Źródło: opracowanie własne na podstawie danych empirycznych przeprowadzonego eksperymentu.

Należy zwrócić uwagę na niewielkie odchylenia standardowe, które świadczą o tym, że protokół Silac nie faworyzuje żadnego węzła.

Wykres 3.10a pokazuje, że w ciągu każdej sekundy rozgrywki, mniej niż jeden obiekt będący bezpośrednio przy awatarze badanego węzła był w niewłaściwym stanie.

Wykres 3.10b prezentuje, że podczas rozgrywki między dwoma węzłami, których awatary znajdowały się blisko siebie, występowały identyczne opóźnienia, a zatem żaden z uczestników rozgrywki nie posiadał przewagi nad drugim.

Rozdział 4

Podsumowanie

W pracy zaprojektowano i zaimplementowano protokół wymiany danych między uczestnikami gry wieloosobowej o rozproszonej architekturze. Zdecydowano się oprzeć protokół na algorytmie plotkującym, który zapewniał losowe rozprzestrzenianie danych w sieci. Następnie przeprowadzono analizę środowiska, które protokół obsługuje. W wyniku tej analizy przyjęto następujące założenia:

- dane wysyłane są często (kilkanaście razy na sekundę),
- dane szybko stają się nieaktualne,
- utrata krótkich (kilkuelementowych) sekwencji komunikatów jest akceptowalna,
- najważniejsze dla gracza są dane o obiektach blisko niego.

Główne cele projektowanego protokołu zostały zdefiniowane jak następuje:

- opóźnienia w przesyłaniu danych dotyczących obiektów widocznych dla gracza powinny być minimalne,
- wszyscy gracze powinni być traktowani jednakowo z dokładnością do przepustowości ich łącz i występujących na nich opóźnieniach,
- sieć musi być odporna na nagłą utratę węzła lub komunikatu.

W celu lepszego dopasowania protokółu do potrzeb gry wieloosobowej, opracowano autorski model wyboru węzłów, do których wysyłane są wiadomości. Zakładał on, że pierwszeństwo w otrzymaniu informacji jest funkcją odległości i kąta między awatarami należącymi do węzłów, czyli pośrednio ich widoczności dla gracza, oraz deklarowanej przez nie wolnej przepustowości sieci oraz ostatniej wymiany komunikatów.

Idea zastosowania algorytmu plotkującego w grze wieloosobowej nie jest nowym pomysłem [10, 28], jednakże niniejsza praca jest pierwszą znaną autorowi, w której przeprowadzono dogłębną analizę takiego rozwiązania.

Protokół został napisany w języku C/C++ dla środowiska systemu operacyjnego Linux. Ponieważ projekt wykorzystuje bibliotekę operacji wejścia-wyjścia oferowaną przez Boost.org, jest niezależny od systemowej implementacji gniazd internetowych i można z powodzeniem zastosować go w środowisku Microsoft Windows.

Protokół został poddany weryfikacji poprawności i wydajności. Przeprowadzona weryfikacja wykazała, że ilość danych wysyłanych przez pojedynczy węzeł rośnie liniowo wraz

z rozmiarem sieci. Opóźnienia i utrata informacji, będące skutkiem zastosowania rozproszonej architektury oraz ograniczeń na liczbę wysyłanych komunikatów, są akceptowalne nawet dla dużej liczby węzłów. Ponadto protokół nie rozróżnia węzłów z dokładnością do przepustowości ich łącz, zatem nie daje żadnemu z nich przewagi nad innymi.

Dalszym kierunkiem badań nad protokołem powinno być zastosowanie kompresji wysyłanych danych (np. kodowania Huffmana). Ponieważ wysyłane dane nie są danymi losowymi, kodowanie Huffmana może znacznie ograniczyć ich ilość i pozwolić na równoczesną grę większej liczby graczy. Możliwe jest również zastosowanie silniejszych warunków w funkcji doboru węzłów do komunikacji, przede wszystkim podczas wyboru węzłów, które już mogły daną wiadomość otrzymać. Takie rozwiązanie zmniejszałoby losowość rozprzestrzeniania danych po sieci, lecz zakładając, że ruch awatarów uczestników rozgrywki jest celowy i w jakimś stopniu przewidywalny, deterministyczny algorytm mógłby okazać się lepszy. Protokół wymaga również pogłębionej weryfikacji, zwłaszcza w celu określenia maksymalnej liczby graczy, dla których może prowadzić spójną i sprawiedliwą rozgrywkę.

W niniejszej pracy badano możliwości zastosowania algorytmów plotkujących w grach wieloosobowych. Eksperyment dał wynik pozytywny. Wraz z zakończeniem pracy kod źródłowy zostanie opublikowany w sieci na zasadach otwartego oprogramowania. Ponieważ protokół Silac został napisany dla logiki gry Quake 3 Arena, może, przy niewielkich zmianach, z powodzeniem zastąpić protokół sieciowy każdej gry pochodzącej z rodziny Quake.

Dodatek A

Załączniki na płycie CD

- Elektroniczna wersja niniejszej pracy.
- Kod źródłowy protokołu Silac.
- Dokumentacja protokołu Silac wykonana za pomocą Doxygen.
- Skrypty testowe protokołu Silac.

Bibliografia

[1] André Allavena, Alan Demers, John E. Hopcroft, *Correctness of a gossip based membership protocol*, PODC '05 Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing, Las Vegas, NV, USA, 2005.

[2] Rajesh Krishna Balan, Maria Ebling, Paul Castro, Archan Misra, *Matrix: Adaptive Middleware for Distributed Multiplayer Games*, Middleware '05 Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware, Grenoble, France, 2005.

[3] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, *The Effects of Loss and Latency on User Performance in Unreal Tournament 2003*, NetGames August 2004.

[4] Bethesda Softworks. http://www.bethsoft.com.

[5] Ashwin Bharambe, Jeffrey Pang, Srinivasan Seshan, *Colyseus: A Distributed Architecture for Online Multiplayer Games*, NSDI'06 Proceedings of the 3rd conference on Networked Systems Design & Implementation, Volume 3, San Jose, CA, USA, 2006.

[6] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, *Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games*, SIGCOMM '08 Proceedings of the ACM SIGCOMM 2008 conference on Data communication, Seattle, WA, USA, 2008.

[7] Blizzard Entertainment. http://us.blizzard.com/en-us/company/press/pressreleases.html?101007.

[8] Kuan-Ta Chen, Chin-Laung Lei, *Network Game Design: Hints and Implications of Player Interaction*, NetGames '06 Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, Singapore, 2006.

[9] Danny Chihdo, *Reinventing Oblivion*. http://web.archive.org/web/20071223022131/http://www.xbox.com/en-US/games/t/theelderscrollsIVoblivion/20051110-fe.htm.

[10] Gabriele D'Angelo, Stefano Ferretti, Moreno Marzolla, *Adaptive Event Dissemination for Peer-to-Peer Multiplayer Online Games*, ICST/CREATE-NET DISIO '11 Proceedings of the 2nd Workshop on Distributed Simulation and Online Gaming, Barcelona, Spain, 2011.

[11] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars, *Computational Geometry: Algorithms and Applications, Second Edition*, 2000.

[12] Niels Drost, Elth Ogston, Rob V. van Nieuwpoort, Henri E. Bal, *ARRG: Real-World Gossiping*, HPDC '07 Proceedings of the 16th International Symposium on High Performance Distributed Computing, Monterey, CA, USA, 2007.

[13] Entertainment Software Association, *2010 Essential Facts About the Computer and Video Game Industry.*

[14] Entertainment Software Association, *Video Games in the 21st Century: the 2010 Report.*

[15] Game Environments Internet Utilisation Study, Swinburne University of Technology. http://caia.swin.edu.au.

[16] Piers Harding-Rolls, *Subscription MMOGs: Life Beyond World of Warcraft*, Screen Digest, 2009.

[17] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, Subhash Suri, *Towards Realistic Mobility Models For Mobile Ad hoc Networks*, MobiCom '03 Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, San Diego, CA, USA, 2003.

[18] Emmanuel Lety, Laurent Gautier, Christophe Diot *MiMaze, a 3D Multi-Player Game on the Internet*, Proceedings of the 4th International Conference on Virtual System and Multimedia, Gifu, Japan, 1998.

[19] Nancy Lynch, Seth Gilbert, *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*, ACM SIGACT News, Volume 33 Issue 2 (2002), 51–59.

[20] Gordon E. Moore, *Cramming more components onto integrated circuits*, Electronics, volume 38, number 8, 19 April 1965.

[21] Gordon E. Moore, *Excerpts from "A conversation with Gordon Moore: Moore's Law"*, Intel Corporation 2005.

[22] Soraia Raupp Musse, Daniel Thalmann, *A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis*, Workshop Computer Animation and Simulation of Eurographics (1997) 39–52.

[23] Nielsen's Law of Internet Bandwidth. http://www.useit.com/alertbox/980405.html

[24] NPD Group Inc. http://www.npd.com.

[25] PricewaterhouseCoopers, *Global Entertainment and Media Outlook 2010-2014.*

[26] Quake II. http://www.idsoftware.com/games/quake/quake2.

[27] Quake III Arena. http://www.idsoftware.com/games/quake/quake3-arena.

[28] Christian Seeger, Patric Kabus, Bettina Kemme, *Area-Based Gossip Multicast*, NetGames '08 Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, Worcester, MA, USA, 2008

[29] Swee Ann Tan, William Lau, Allan Loh, *Networked Game Mobility Model for First-Person Shooter Games*, NetGames '05 Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games, Hawthorne, NY, USA, 2005.

[30] United States Patent and Trademark Office. U.S. Patent #2,455,992.

[31] Spyros Voulgaris, Daniela Gavidia, Maarten Van Steen, *CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays*, Journal of Network and Systems Management, Vol. 13, No. 2. (1 June 2005), 197–217.