

# Connect4 - Raport format LNCS

Radu Rareș - Aurelian, 2B5

Universitatea ALEXANDRU IOAN CUZA din Iași, Facultatea de Informatică

## Introducere

Acest raport are rolul de a documenta etapele de implementare ale simulării jocului *Connect Four* într-o aplicație client/server. *Connect Four* este un joc care se joacă în doi jucători, unde fiecare jucător va primi jetoane de culoare *roșu sau galben*. Aceștia își vor alege o culoare (în cazul nostru primul jucător care se conectează va alege primul culoarea, astfel va determina culoarea celui de-al doilea jucător), iar apoi, pe rând, vor alege pe ce coloană vor așeza jetonul într-o tabla de dimensiunea 6x7. Jetonul va cădea pe cea mai joasă poziție valabilă din coloana aleasă. Primul jucător care va avea 4 jetoane de aceeași culoare pe o linie, coloană sau diagonală va câștiga. [1]

## Tehnologii utilizate

Conexiunea dintre server și clienți este asigurată prin folosirea *protocolului TCP*, protocol orientat conexiune care asigură trimiterea datelor în întregime și în ordine către destinatar.[2]

Protocolul UDP nu a fost ales deoarece acesta este un protocol neorientat conexiune, iar acest tip de protocol nu asigură faptul că toate datele trimise vor ajunge la destinație și nu asigură ordinea de trimitere/primire a datelor, iar pentru implementarea simulării jocului *Connect Four* este necesară primirea datelor în întregime, precum și ordinea în care acestea au fost trimise sau primite pentru ca nu cumva un jucător să înceapă înaintea celuilalt sau ca nu cumva să mute de două sau mai multe ori înainte ca celălalt jucător să mute. [2]

Este folosit modelul client/server TCP *concurrent*, concurența este implementată prin crearea unui *thread* pentru fiecare doi jucători conectați, astfel când doi jucători sunt conectați, ei vor avea tabla lor de joc, iar dacă se vor mai conecta mai multe perechi de jucători, aceștia vor avea o tablă diferită de joc față de ceilalți jucători.

De ce *threading*? Thread-urile au memorie partajată deci nu este nevoie de o copie intermediară a memoriei, nu avem un transfer de date în sine și totodată thread-urile au propriile date private. Un alt aspect important ar fi că crearea unui pool de thread-uri este mai eficientă din punctul de vedere al factorului de timp decât crearea unei mulțimi de procese copil. Mai mult, aplicațiile ce folosesc thread-uri sunt în general mai rapide decât dacă utilizează procese și avem nevoie doar de un array care ține evidența thread-urilor și o structură de date care va putea conține anumite informații pe care dorim să le transmitem pentru a evita folosirea de variabile globale.[3]



## Detalii de implementare

Se va folosi o structura pentru a ține evidența id-ului thread-ului precum și clienții pe care îi va servi, care vor fi identificați prin descriptorul de fisier returnat de funcția *accept()* și o structură pentru ținerea evidenței "identității" unui jucător prin culoarea pe care o va avea, precum și scorul său contra jucătorului cu care joacă împotriva.

Procesul server va aștepta conectarea a cel puțin doi jucători pentru a putea începe jocul. Odată conectați, se va crea un thread pentru a-i servi pe aceștia.

Server-ul va ține evidența mutărilor jucătorilor într-un tablou bidimensional de 6 rânduri și 7 coloane, care reprezintă tabla de joc.

Primul client care se va conecta, va primi de la server un mesaj (printr-un apel *write()*) pentru a decide ce culoare va dori să aleagă, *R(ogru)* sau *G(alben)*, al doilea client conectat va aștepta alegerea primului client. Clientul va aștepta mesajul de la server printr-un apel *read()*, acesta va alege, iar în funcție de alegerea sa, cel de-al doilea client va primi culoarea rămasă disponibilă.

Din nou, primul client conectat va primi de la server un mesaj pentru a decide dacă ar dori să înceapă primul sau nu, iar al doilea client va aștepta din nou alegerea primului jucător. După ce alege, informațiile necesare înainte de începerea jocului sunt stabilite.

Serverul le va trimite jucătorilor tabla de joc printr-un sir de caractere care va reprezenta tabla de joc, iar clienții vor procesa acel sir de caractere pentru a se afișa în o tabla de joc de dimensiunea 6x7.

Jucătorul care începe primul va alege o coloană, un număr din intervalul [1,7], pe care dorește să pună jetonul. Dacă este plină coloana, atunci va trebui să aleagă altă coloană. Dacă coloana nu este plină, jetonul va fi pus pe cea mai de jos linie disponibilă din acea coloană. Jucătorul va trimite alegerea sa server-ului, server-ul va procesa tabla de joc nouă și va verifica dacă, după acea alegere, jucătorul a câștigat sau dacă este egalitate, în caz contrar le trimite ambilor jucători din nou tabla nou updatată.

În acest timp, cel de-al doilea jucător așteaptă mutarea celuilalt jucător precum și noua tablă de joc de la server. Când îi vine rândul, acesta va repeta pașii primului jucător, va alege pe ce coloană vrea să pună jetonul, se verifică dacă alegerea este validă, îi trimite server-ului alegerea, server-ul procesează noua tablă, verifică dacă jucătorul a câștigat sau dacă este egalitate, în cazul în care nu este egalitate sau nu a câștigat, trimite ambilor jucători tabla nou updatată.

Se repetă acești pași până când unul din jucători câștiga sau este egalitate. Un jucător poate câștiga dacă reușește să aibă 4 jetoane de aceeași culoare, una lângă alta, pe o linie, o coloană sau pe o diagonală, în acest caz, se va incrementa cu 1 scorul jucătorului care a câștigat, scorul jucătorului care a pierdut va rămâne același, iar ambii jucători vor primi de la server mesajul aferent statutului, dacă au câștigat sau nu. Egalitate este în cazul în care toată tabla este plină și niciun jucător nu a câștigat, în acest caz, scorul ambilor jucători va fi incrementat cu 1 și vor primi un mesaj de la server cum că este remiză.

Odată ajuns în stadiul de câștig sau de egalitate, server-ul le va trimite jucătorilor un mesaj dacă vor mai dori să continue să joace sau nu. Clienții vor

răspunde cu "Da" sau "Nu", dacă răspunsul ambilor jucătorilor este afirmativ, atunci se va începe o nouă partidă, reluându-se toți pașii de mai sus, în cazul în care ambele răspunsuri sunt negative, sau doar unul este afirmativ, nu se va începe o nouă partidă, astfel server-ul va închide conexiune cu cei doi jucători.

## Concluzii

În concluzie, pentru a îmbunătăți experiența utilizatorilor care vor dori să încerce această implementare a jocului *Connect Four* se poate adăuga o interfață grafică, se poate adăuga o variantă pentru care un jucător să aibă ca oponent calculatorul, se poate implementa un mod de alegere a oponentului de către un jucător, precum și posibilitatea de a crea un cont pentru a urmări eventualele statistici împotriva altor jucători cu care a jucat înainte.

## References

1. Informații despre jocul Connect Four: [https://en.wikipedia.org/wiki/Connect\\_Four](https://en.wikipedia.org/wiki/Connect_Four)
2. Informații despre protocolul TCP și protocolul UDP: [https://profs.info.uaic.ro/~computernetworks/files/6rc\\_ProgramareaInReteaII\\_Ro.pdf](https://profs.info.uaic.ro/~computernetworks/files/6rc_ProgramareaInReteaII_Ro.pdf) și <https://profs.info.uaic.ro/~ioana/Week7>
3. Informații despre Thread-uri în POSIX: [https://profs.info.uaic.ro/~computernetworks/files/7rc\\_ProgramareaInReteaIII\\_Ro.pdf](https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf) și <https://computing.llnl.gov/tutorials/pthreads/>
4. Informații despre Socket-uri BSD: [https://profs.info.uaic.ro/~computernetworks/files/5rc\\_ProgramareaInReteaI\\_ro.pdf](https://profs.info.uaic.ro/~computernetworks/files/5rc_ProgramareaInReteaI_ro.pdf)