

סיכום שיעור 7

פרוטוקול MCP וליגת סוכנים

מרצה: ד"ר יורם סגל

קורס: סוכני בינה מלאכותית

מילות מפתח:

- MCP • Client-Server • פרוטוקול תקשורת • סוכנים • ליגה • כלים (Tools)
- משאבים (Resources) • ניהול ששנים • JSON • ארכיטקטורה מבוזרת

תוכן העניינים

1	מבוא לפרוטוקול MCP	2
1.1	רקע והתפתחות	2
2.1	הבעיה שהפרוטוקול פותר	2
2	ארכיטקטורת Client-Server ב-MCP	2
1.2	הגדרות בסיסיות	2
2.2	מחזור חיי החיבור	3
3	רכיבי הפרוטוקול	3
1.3	כלים (Tools)	3
2.3	משאבים (Resources)	3
3.3	הנחיות (Prompts)	3
4	ארכיטקטורת הקליינט	3
1.4	שכבות הקליינט	3
2.4	מודולים הכרחיים	4
1.2.4	מנהל הסשנים (Session Manager)	4
2.2.4	רישום כלים (Tool Registry)	4
3.2.4	ניהול תור הודעות (Message Queue)	4
3.4	טיפול בשגיאות	4
5	עבודה עם מספר שרתים	5
1.5	טופולוגיית כוכב	5
2.5	איזון עומסים (Load Balancing)	5
6	אבטחת מידע	5
1.6	איומים נפוצים	5
2.6	הגנות מומלצות	5
7	מטלה: ליגת סוכנים	5
1.7	תיאור המשימה	5
2.7	חוקי המשחק	6
3.7	עקרונות ארכיטקטוניים	6
4.7	מבנה הודעה בפרוטוקול	6
8	סיכום	6
1.8	נקודות מפתח	6
2.8	הכנה לפרויקט גמר	7

1 מבוא לפרוטוקול MCP

1.1 רקע והתפתחות

פרוטוקול MCP (Model Context Protocol) הוא תקן תקשורת שפותח על ידי Anthropic לאפשר תקשורת אחידה בין מודלי שפה (LLM) לבין שירותים חיצוניים. הפרוטוקול מהווה מהפכה בתחום סוכני הבינה המלאכותית.

התפתחות היסטורית:

- עיתונים – דרך תקשורת ראשונית
- HTTP – פרוטוקול האינטרנט
- USB – סטנדרט להעברת נתונים פיזית
- MCP – פרוטוקול לתקשורת בין סוכני AI

2.1 הבעיה שהפרוטוקול פותר

לפני MCP, כל סוכן נדרש לאינטגרציה ייחודית לכל שירות. מצב זה יצר:

- מורכבות של $O(n \times m)$ – למשל, 5 סוכנים כפול 5 שרתים = 52 אינטגרציות
- קושי בתחזוקה והרחבה
- בעיות סקלבליות

עם MCP: המורכבות יורדת ל- $O(n + m)$ – למשל, 5 סוכנים + 5 שרתים = 01 חיבורים בלבד.

2 ארכיטקטורת Client-Server ב-MCP

1.2 הגדרות בסיסיות

קליינט (Client):

- יוזם את הפנייה לשרת
- מכיל את מודל השפה (LLM)
- מנהל סשנים, כלים, הודעות ומשאבים
- הוא "המוח" של המערכת

שרת (Server):

- פסיבי – ממתין לבקשות
- מספק כלים (Tools) ומשאבים (Resources)
- הוא "הידים והרגליים" של המערכת
- מבצע פעולות דטרמיניסטיות

חשוב להבין: ב-MCP, הקליינט הוא לא באמת קליינט טהור והשרת הוא לא באמת שרת טהור – שניהם יכולים גם ליזום וגם לקבל בקשות.

2.2 מחזור חיי החיבור

1. שלב האתחול (Initialize): פתיחת חיבור בין הקליינט לשרת
2. שלב הפעולה (Operation): הקליינט שולח בקשות, השרת מחזיר רשימת יכולות וכלים
3. שלב הסיום (Close): סגירה מסודרת ושחרור משאבים

3 רכיבי הפרוטוקול

1.3 כלים (Tools)

כלים הם פונקציות שהמודל יכול להפעיל לביצוע פעולות או חישובים. מבנה הגדרת כלי:

- name – שם ייחודי לזיהוי
 - description – הסבר למודל מה הכלי עושה
 - input_schema – הגדרת פרמטרים בפורמט JSON Schema
- דוגמה: מחשבון – מקבל שני אופרנדים ופעולה, מחזיר תוצאה.

2.3 משאבים (Resources)

משאבים הם מקורות מידע שהמודל יכול לקרוא אך לא לשנות (Read-Only). דוגמאות:

- בסיסי נתונים
- דפי אינטרנט
- קבצי PDF
- תמונות

3.3 הנחיות (Prompts)

תבניות מוכנות להנחיית המודל לביצוע משימות ספציפיות.

4 ארכיטקטורת הקליינט

1.4 שכבות הקליינט

1. מודל השפה (LLM): מקבל החלטות ויוזם פעולות
2. ממשק הלקוח (Client Interface): ה-API שהמודל עובד מולו
3. ליבת המערכת: מנהל סשנים ורישום כלים
4. עיבוד הודעות: המרה ל-JSON
5. שכבת תקשורת: תרגום ל-HTTP או STDIO

2.4 מודולים הכרחיים

1.2.4 מנהל הסשנים (Session Manager)

- מנהל מחזור חיים של חיבורים
- מבצע Handshake – אימות שהחיבור הצליח
- מנהל Heartbeat – בדיקות דופק תקופתיות
- Retry Logic – מנגנון אוטומטי לחידוש קשר

2.2.4 רישום כלים (Tool Registry)

- מחזיק רשימת כלים זמינים מכל שרת
- מרכז את המידע לשימוש ה-LLM
- מטפל בהתנגשות שמות בין שרתים שונים

3.2.4 ניהול תור הודעות (Message Queue)

- מנהל תור של הודעות נכנסות ויוצאות
- מטפל בעדיפויות
- מונע הצפה

3.4 טיפול בשגיאות

סוגי שגיאות:

- Transient: שגיאות זמניות (רשת, עומס) – שווה לנסות שוב
- Permanent: שגיאות קבועות (קובץ חסר, הרשאה) – אין טעם לנסות
- Timeout: חריגה בזמן – ניתן להגדיל את זמן ההמתנה

אסטרטגיית Exponential Backoff:

זמן ההמתנה בין ניסיונות גדל באופן אקספוננציאלי:

- ניסיון 1: המתנה קצרה
- ניסיון 2: המתנה כפולה
- ניסיון 3: המתנה פי 4
- ...וכן הלאה...

חשוב: יש להוסיף Jitter (רעש אקראי) כדי למנוע מספר תהליכים שמתעוררים בו-זמנית.

5 עבודה עם מספר שרתים

1.5 טופולוגיית כוכב

קליינט אחד מרכז את התקשורת מול שרתים רבים. הקליינט:

- מציג ל-LLM רשימה מאוחדת של כלים
- מנהל מרחבי שמות למניעת התנגשות
- מנתב בקשות לשרת המתאים

2.5 איזון עומסים (Load Balancing)

אסטרטגיות נפוצות:

- Round Robin: חלוקה מעגלית שווה
- Least Connections: הפניה לשרת הפנוי ביותר
- Weighted: עדיפות לשרתים חזקים יותר

6 אבטחת מידע

1.6 איומים נפוצים

התקפות Prompt Injection דרך תוכן באינטרנט:

- האקרים מכניסים הוראות זדוניות לדפי אינטרנט תמימים
- ה-LLM קורא את התוכן ומבצע את ההוראות
- אין כיום פתרון מושלם לבעיה זו

2.6 הגנות מומלצות

- עבודה בתוך Docker או סביבות וירטואליות (Sandbox)
- הגבלת הרשאות הגישה לתיקיות
- הגדרת גבולות נזק מראש
- תיעוד (Logging) מלא של כל הפעולות

7 מטלה: ליגת סוכנים

1.7 תיאור המשימה

בניית ליגה של סוכני AI שמשחקים במשחק "זוג או פרד".

רכיבי המערכת:

1. **מנהלת הליגה:** מגדירה מחזורי משחקים, מצוותת שחקנים, מנהלת דירוג
2. **שופט:** מנהל משחק בודד בין שני שחקנים
3. **שחקנים:** ארבעה סוכנים שמשחקים זה נגד זה

2.7 חוקי המשחק

- כל שחקן בוחר "זוגי" או "אי-זוגי"
- השופט מגריל מספר
- אם שניהם צדקו או שניהם טעו – תיקו
- אם אחד צדק והשני טעה – המנצח הוא מי שצדק
- 03 שניות לתגובה, 3 ניסיונות לפני פסילה

3.7 עקרונות ארכיטקטוניים

1. הפרדת אחריות: שכבת הליגה והשיפוט אינן תלויות במשחק הספציפי
2. מודולריות: ניתן להחליף את המשחק בקלות
3. פרוטוקול אחיד: כל ההודעות ב-JSON עם מבנה קבוע

4.7 מבנה הודעה בפרוטוקול

כל הודעה חייבת לכלול:

- protocol – גרסת הפרוטוקול
- message_type – סוג ההודעה
- league_id – מזהה הליגה
- round – מספר המחזור
- match_id – מזהה המשחק
- timestamp – חותמת זמן

8 סיכום

1.8 נקודות מפתח

1. MCP מאפשר תקשורת אחידה בין סוכני AI לשירותים חיצוניים
2. הקליינט הוא "המוח" (עם ה-LLM), השרת הוא "הידיים והרגליים"
3. יש להקפיד על ניהול ששנים, כלים, והודעות
4. מנגנוני התאוששות ואבטחה הם קריטיים
5. הארכיטקטורה חייבת להיות מודולרית וסקלבילית

2.8 הכנה לפרויקט גמר

המטלה הנוכחית היא הבסיס לפרויקט הגמר:

- בפרויקט הגמר – ליגה של כל הכיתה
- המשחק יהיה מורכב יותר (אולי "איקס עיגול גדול")
- הסוכנים ישתמשו באסטרטגיות AI אמיתיות

כל הזכויות שמורות לד"ר יורם סגל (c)