รายงานการออกแบบโปรโตคอลสำหรับการพัฒนา Network Application

ชื่อโปรโตคอล: Home-Server Telemetry Protocol (HSTP)

1. วัตถุประสงค์ของโปรแกรม

โปรแกรมนี้เป็น Network Application ที่ทำหน้าที่เป็นระบบดึงข้อมูลทางไกล (Telemetry System) ใช้สำหรับตรวจสอบสถานะและบริหารจัดการเซิร์ฟเวอร์ขนาดเล็ก

- **โปรแกรมนี้คืออะไร:** เป็นโปรแกรมแบบ Client-Server Architecture ที่ออกแบบมาเพื่อการดึงข้อมูลทรัพยากรระบบ (System Resources) และจัดการ Service ต่างๆ
- **ใช้สำหรับทำอะไร:** ใช้ส่งคำสั่งจากเครื่อง Client เพื่อร้องขอข้อมูล เช่น สถานะการทำงานของ CPU/RAM, ตรวจสอบสถานะการทำงานของ Service (เช่น Docker, Plex, Pi-hole) และรองรับคำสั่งควบคุมเบื้องต้น เช่น การสั่ง Restart Service

2. คุณลักษณะของโปรแกรม (Application Characteristics)

- **Lightweight & Low Overhead:** ตัวโปรแกรมและโปรโตคอลถูกออกแบบมาให้มีการใช้แบนด์วิดท์ที่ต่ำมาก โดยส่งข้อมูลเป็น Plain Text ขนาดเล็ก (ระดับ Bytes) แทนที่จะใช้โครงสร้างที่ซับซ้อนอย่าง JSON หรือ XML
- **Low Latency:** ตัด Overhead ของ HTTP Header ทิ้งไป ทำให้การตอบสนองต่อ Request รวดเร็วในระดับเสี้ยววินาที (Millisecond)

- **Concurrent Processing:** ฝั่ง Server รองรับการจัดการผู้ใช้งานหลายคน (Multi-client) ในเวลาเดียวกัน ผ่านเทคนิค Multi-threading

3. การเลือกใช้ Transport Layer

โปรแกรมนี้เลือกใช้ Service Model ของ Transport Layer แบบ **TCP (Transmission Control Protocol)**

- **เหตุผล:** ระบบ Telemetry และการสั่งการบริหารจัดการเซิร์ฟเวอร์ ต้องการความน่าเชื่อถือของข้อมูล (Reliability) สูงสุด คำสั่งอย่างการตรวจสอบสถานะระบบ หรือคำสั่งสั่ง Restart Service จะต้องถูกส่งไปถึง Server อย่างครบถ้วนสมบูรณ์และเรียงลำดับอย่างถูกต้อง หากใช้ UDP ข้อมูลอาจสูญหายระหว่างทาง ซึ่งจะทำให้ฝั่ง Client ได้รับข้อมูลสถานะที่ไม่ตรงกับความเป็นจริง หรือคำสั่งควบคุมระบบอาจทำงานผิดพลาดได้

4. การออกแบบ Application-Layer Protocol

โปรโตคอล **HSTP** มีรูปแบบโครงสร้างข้อความ (Message Format) เป็น Plain text (UTF-8) โดยทั้งฝั่ง Request และ Response จะใช้สัญลักษณ์ \n (Newline) เป็นตัวบอกจุดสิ้นสุดของข้อความ (Delimiter)

4.1 รูปแบบ Request Message (ฝั่ง Client)

โครงสร้าง: <COMMAND> [ARGUMENTS]\n

คำสั่ง (Command)	**พารามิเตอร์ (Arguments)**	**คำอธิบาย**
REQ_SYS	(ไม่มี)	ร้องขอข้อมูลการใช้ทรัพยากรพื้นฐาน เช่น CPU และ RAM
REQ_SVC	<service_name>	ร้องขอสถานะการทำงานของ Service ที่ระบุ
CMD_RESTART	<service_name>	ส่งคำสั่งขอ Restart Service ที่ระบุ
EXIT	(ไม่มี)	แจ้ง Server เพื่อขอยกเลิกการเชื่อมต่อ Socket

4.2 รูปแบบ Response Message (ฝั่ง Server)

โครงสร้าง: <STATUS_CODE> <STATUS_PHRASE> [DATA]\n

Status Code	**Status Phrase**	**ความหมาย**	**ตัวอย่าง Response ที่ส่งกลับ**
200	OK_STATUS	คำสั่งทำงานสำเร็จและแนบข้อมูลกลับ	200 OK_STATUS CPU:12%|RAM:4GB
200	OK_BYE	ตอบรับการตัดการเชื่อมต่อจาก Client	200 OK_BYE
202	ACCEPTED	รับคำสั่งแล้ว กำลังดำเนินการทำงาน	202 ACCEPTED Restarting docker...
400	BAD_REQUEST	คำสั่งผิดรูปแบบ หรือไม่รู้จักคำสั่งนี้	400 BAD_REQUEST Command not supported
404	NOT_FOUND	ไม่พบ Service หรือ Resource ที่ร้องขอ	404 NOT_FOUND Service 'apache' unknown