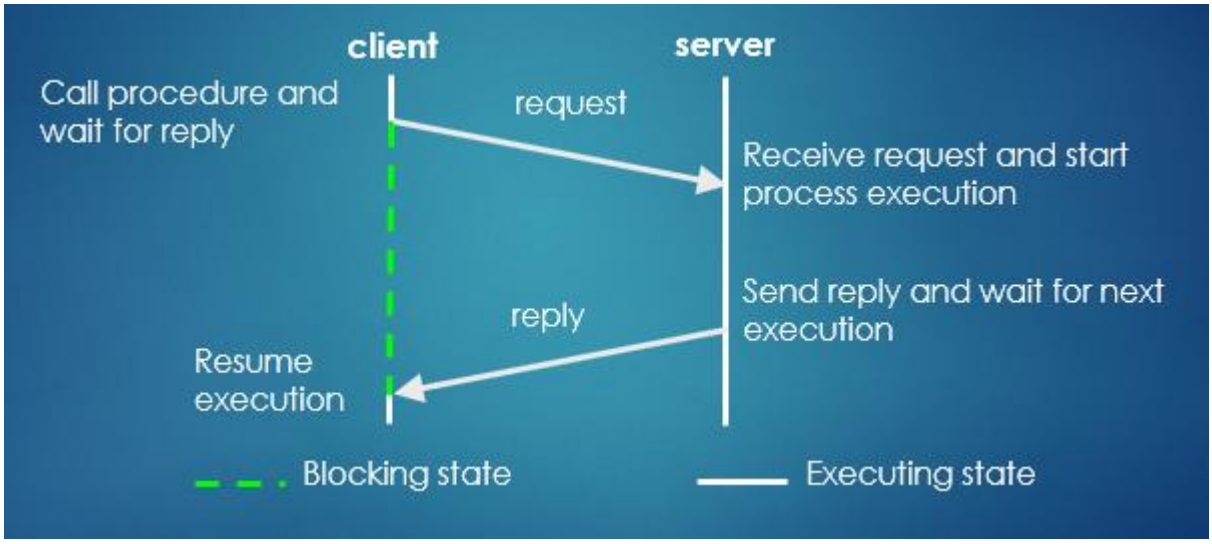
RPC & REST

RPC (อาร์พีซี) ย่อมาจากคำว่า "Remote Procedure Call)" เป็นโปรโตคอลตัวหนึ่งทำหน้าที่เรียกขอใช้บริการและตอบรับคำขอใช้บริการ procedure หรือโปรแกรมที่อยู่บนคอมพิวเตอร์เครื่องอื่นในเครือข่ายเดียวกันรวมถึงยังเป็นโปรโตคอลที่มีรูปแบบโครงสร้างแบบ ลูกข่าย/แม่ข่าย

RPC (อาร์พีซี) คือไลบรารีของ Procedure Procedure อนุญาตให้หนึ่งกระบวนการของไคลเอ็นต์ ส่งตรงไปยังกระบวนการ Server เพื่อรันการเรียก Procedure หากกระบวนการของ Client ได้รันการเรียกในพื้นที่แอดเดรสของตัวเอง เนื่องจาก Client และ Server คือกระบวนการสองขั้นตอนที่แบ่งแยกกัน

- การเรียกขั้นตอน (function) ระยะไกล (RPC) เป็นรูปแบบระดับสูงสำหรับการติดต่อสื่อสารระหว่าง client-server
- มันให้กลไกโปรแกรมเมอร์ที่คุ้นเคยสำหรับการสร้างระบบกระจาย (ระบบขนาดใหญ่)
- ตัวอย่าง: บริการไฟล์ (File Service), บริการตรวจสอบสิทธิ์ (Authenticate Service)

RPC Model



Characteristics (ลักษณะ)

- ขั้นตอนที่เรียกว่าอยู่ในกระบวนการอื่นซึ่งอาจอยู่ในเครื่องอื่น
- กระบวนการจะไม่แชร์พื้นที่ที่ใช้อยู่ (address space)
- ไม่อนุญาตการส่งผ่านพารามิเตอร์โดยการอ้างอิงและการผ่านตัวชี้ (not support call by reference)
- พารามิเตอร์ถูกส่งผ่านโดยค่า (call by value)
- Procedure แบบรีโมตที่เรียกใช้เรียกใช้งานภายในสภาพแวดล้อมของกระบวนการ Server
- Procedure ที่เรียกไม่สามารถเข้าถึงสภาพแวดล้อมของ Procedure ที่เรียกได้ (Server เข้าถึง Client ไม่ได้)
- ไม่มีข้อความที่ผ่านหรือ I / O เลยโปรแกรมเมอร์จะมองเห็นได้ (Programmer จะไม่เห็นข้อมูลที่วิ่งผ่าน Network)

Features (คุณสมบัติ)

- Simple call syntax (ไวยากรณ์การใช้งานอย่างง่าย)
- Familiar semantics (ความหมายที่คุ้นเคย)
- Well defined interface (interface ที่กำหนดไว้อย่างดี)
- Ease of use (สะดวกในการใช้)
- Efficient (ที่มีประสิทธิภาพ)
- Can communicate between processes on the same machine or different machines (สามารถสื่อสารระหว่างกระบวนการในเครื่องเดียวกันหรือเครื่องที่แตกต่างกัน)

Limitations (ข้อจำกัด)

- พารามิเตอร์ที่ส่งผ่านโดยค่าเท่านั้นและไม่อนุญาตให้ใช้ตัวชี้ (not support call by reference)
- ความเร็ว: เวลาในการเรียกใช้ Procedure แบบรีโมต (และส่งคืน) เวลา (overheads) สามารถมีความสำคัญ (1 - 3 ออเดอร์ของขนาด) ช้ากว่าสำหรับโพรซีเดอร์ในพื้นที่
- ความล้มเหลว: RPC มีความเสี่ยงต่อความล้มเหลวได้มากขึ้น (เนื่องจากเกี่ยวข้องกับระบบสื่อสารเครื่องอื่นและกระบวนการอื่น)
- โปรแกรมเมอร์ควรตระหนักถึงความหมายของการโทรนั่นคือโปรแกรมที่ใช้ RPC จะต้องมีความสามารถในการจัดการข้อผิดพลาดที่ไม่สามารถเกิดขึ้นได้ในการเรียก Procedure ในเครื่อง

Design Issues (ปัญหา)

- (Exception handling) การจัดการข้อยกเว้น
- จำเป็นเนื่องจากความเป็นไปได้ของเครือข่ายและจากโหนดที่ล้มเหลว
- RPC ใช้ค่าส่งคืนเพื่อระบุข้อผิดพลาด (ตรวจสอบ error จากค่าที่ return กลับมา)
- (Transparency) ความโปร่งใส
- Syntactic สามารถทำได้โดยใช้ไวยากรณ์เดียวกับการเรียกโปรซีเจอร์ในพื้นที่
- Semantic ความหมายเป็นไปไม่ได้เนื่องจากข้อ จำกัด RPC: ความล้มเหลว (คล้ายกัน แต่ไม่เหมือนกัน)
- (Delivery guarantees) รับประกันการจัดส่ง
- ลองใหม่อีกครั้งข้อความคำขอ: ว่าจะส่งอีกครั้งข้อความคำขอจนกว่าจะตอบกลับหรือ Server จะถือว่าล้มเหลว;
- การกรองซ้ำ: เมื่อมีการใช้การส่งสัญญาณซ้ำจะกรองการทำซ้ำที่ Server หรือไม่
- การส่งใหม่การตอบกลับ: ว่าจะเก็บประวัติของข้อความตอบกลับเพื่อให้การตอบกลับที่หายไปถูกส่งใหม่โดยไม่ต้องดำเนินการ Server อีกครั้ง

Marshalling

- สถาปัตยกรรมที่แตกต่างกันใช้วิธีต่าง ๆ ในการแสดงข้อมูล
- กระบวนการเรียกใช้ (และใช้งาน) ใช้วิธีก
- ารจัดเก็บข้อมูลในรูปแบบของแพลตฟอร์ม
- Middleware มีการแสดงข้อมูลทั่วไป (CDR) ซึ่งไม่ขึ้นอยู่กับแพลตฟอร์ม
- กระบวนการผู้โทรจะแปลง argument เป็นรูปแบบ CDR
- เรียกว่า“ Marshalling”
- กระบวนการ Callee แยก argument จากข้อความเป็นรูปแบบขึ้นอยู่กับแพลตฟอร์มของตนเอง
- เรียกว่า "Unmarshalling"
- ค่าส่งคืนจะถูกบันทึกไว้ในกระบวนการ callee และ unmarshalled ที่กระบวนการของการเรียกใช้

Representational state transfer (REST) คือ การสร้าง Webservice ชนิดหนึ่งที่ใช้สื่อสารกันบน Internet ใช้หลักการแบบ stateless คือไม่มี session ซึ่งต่างจาก webservice แบบอื่นเช่น WSDL และ SOAP การทำงานของ RESTful Webservice จะอาศัย URI/URL ของ request เพื่อค้นหาและประมวลผลแล้วตอบกลับไปในรูป XML, HTML, JSON โดย response ที่ตอบกลับจะเป็นการยืนยันผลของคำสั่งที่ส่งมา และสามารถพัฒนาด้วยภาษา programming ได้หลากหลาย คำสั่งก็จะมีตาม HTTP verbs ซึ่งก็คือ

- GET ทำกการดึงข้อมูลภายใน URI ที่กำหนด
- POST สำหรับสร้างข้อมูล
- PUT ใช้แก้ไขข้อมูล
- DELETE สำหรับลบข้อมูล

ใช้ REST เมื่อต้องการลดขนาดของข้อมูล และ จำนวน bandwidth ที่ใช้งาน ต้องการเมื่อทำงานอยู่บนระบบ web และ mobile ตัวอย่าง เช่น Social media service, Web Chat service

ข้อดี

- ทำการอยู่บน HTTP และทำตามมาตรฐานของ HTTP จึงทำให้พัฒนาได้ง่าย
- สนับสนุนรูปแบบข้อมูลมากมาย เช่น XML, JSON, Plain Text และอื่น ๆ อีกมากมาย
- รองรับการขยายระบบได้ง่าย
- มีประสิทธิภาพการทำงานที่ดี
- รองรับเรื่อง caching ข้อมูล

ข้อเสีย

- ทำงานได้เฉพาะ HTTP protocol เท่านั้น
- ไม่มีเรื่องของ security และ reliability มาให้ในตัว ดังนั้นต้องทำเอง
- รูปแบบข้อมูลที่ส่งไปมาระหว่าง client-server ไม่มีข้อจำกัดอะไรเลย

Simple Object Access Protocol (SOAP)

ใช้ SOAP เมื่อต้องการจัดการ transaction เมื่อต้องทำงานกับหลาย ๆ ระบบ ต้องการความเข้มงวดในการเชื่อมต่อระหว่าง client/server ตัวอย่าง เช่น Finacial service และ Telecommunication service

ข้อดี

- สามารถทำงานอยู่บน protocol ใด ๆ ก็ได้
- อธิบาย service ด้วย WDSL (Web Service Description Language)
- มีความน่าเชื่อถือ เมื่อเกิดปัญหาสามารถทำการ retry ได้
- สนับสนุนเรื่อง security อยู่แล้ว ทั้ง authentication, authorization และ การเข้ารหัสข้อมูล

ข้อเสีย

- ยากต่อการพัฒนา ทำให้ไม่เป็นที่นิยมสำหรับระบบ web และ mobile
- สนับสนุนรูปแบบข้อมูล XML เพียงอย่างเดียว
- เนื่องจากมันเป็น standard ทำให้มีข้อจำกัดเยอะ
- เนื่องจากโครงสร้างมันมีหลายส่วนทำให้มี overhead สูง หรือ ต้องใช้ bandwidth สูงกว่า REST