University Of Science and Technology of Hanoi
ICT Department

Distributed System Report

Lecturer: Le Nhu Chu Hiep

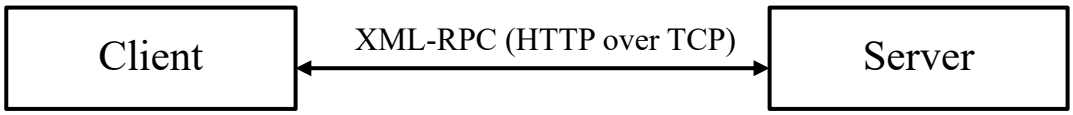
Practical Work 2: RPC File transfer

Student: Nguyễn Thị Vàng Anh - 23BI14032

1. RPC Service Design

The system uses XML-RPC to expose file operations as remote procedures.
Instead of manually sending text commands over raw TCP, the client calls functions on a ServerProxy object, and the XML-RPC library handles serialization and transport over HTTP/TCP.
RPC methods:

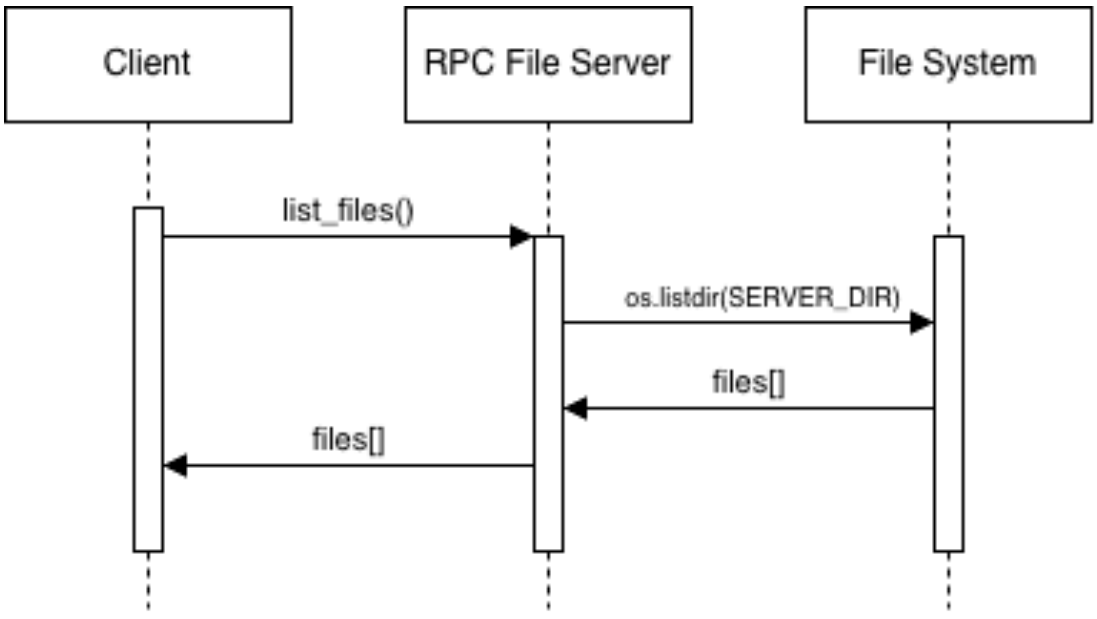
- list_files(): return list of files stored on the server
- upload_file(filename, data): upload a file from client to server (data is binary content)
- download_file(filename): return binary content of a file from server to client



2. System Organization

The system consists of two main programs and two storage folders:

- Client side
  - client_rpc.py: menu, user interface, RPC calls
  - client_files/: folder to store downloaded files
- Server side
  - server_rpc.py: XML-RPC server, file operations, chat functions
  - server_files/: folder to store uploaded files
- For example, RPC service design (list_files):



3. File Transfer Implementation

3.1. Server side - implement RPC functions

```
from xmlrpc.server import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from xmlrpc.client import Binary, Fault
import os

SERVER_DIR = "server_files"
os.makedirs(SERVER_DIR, exist_ok=True)

class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ("/RPC2",)

def list_files():
    return os.listdir(SERVER_DIR)

def upload_file(filename, data):
    try:
        safe_name = os.path.basename(filename)
        path = os.path.join(SERVER_DIR, safe_name)
        with open(path, "wb") as f:
            f.write(data.data)
        print(f"[INFO] Uploaded file: {path}")
        return True
    except Exception as e:
        print(f"[ERROR] upload_file: {e}")
        return False

def download_file(filename):
    safe_name = os.path.basename(filename)
    path = os.path.join(SERVER_DIR, safe_name)

    if not os.path.exists(path):
        raise Fault(1, f"File not found: {filename}")

    with open(path, "rb") as f:
        data = f.read()
    print(f"[INFO] Downloaded file: {path}")
    return Binary(data)

if __name__ == "__main__":
    host = "0.0.0.0"
    port = 9000

    with SimpleXMLRPCServer(
        (host, port),
        requestHandler=RequestHandler,
        allow_none=True,
```

```
        logRequests=True,
    ) as server:
        print(f"[INFO] RPC file server listening on {host}:{port}")

        # Register functions
        server.register_function(list_files, "list_files")
        server.register_function(upload_file, "upload_file")
        server.register_function(download_file, "download_file")

        # Start serving (blocking)
        server.serve_forever()
```

3.2. Client side - call RPC and handle local files

```
from xmlrpc.client import ServerProxy, Binary, Fault
import os

CLIENT_DIR = "client_files"
os.makedirs(CLIENT_DIR, exist_ok=True)

def menu():
    print("\n=== RPC File Transfer Client ===")
    print("1. List files on server")
    print("2. Upload file to server")
    print("3. Download file from server")
    print("0. Exit")

def list_files(proxy):
    try:
        files = proxy.list_files()
        if not files:
            print("No files on server.")
        else:
            print("Files on server:")
            for name in files:
                print(" -", name)
    except Exception as e:
        print("Error listing files:", e)

def upload(proxy):
    filepath = input("Enter local file path to upload: ").strip()

    if not os.path.exists(filepath):
        print("File does not exist.")
        return

    filename = os.path.basename(filepath)
```

```
    try:
        with open(filepath, "rb") as f:
            data = f.read()

        ok = proxy.upload_file(filename, Binary(data))
        if ok:
            print("Upload successful.")
        else:
            print("Upload failed (server returned False).")
    except Exception as e:
        print("Error uploading file:", e)

def download(proxy):
    filename = input("Enter server file name to download: ").strip()

    try:
        binary_data = proxy.download_file(filename)
        # Save to client_files directory
        save_path = os.path.join(CLIENT_DIR, filename)
        with open(save_path, "wb") as f:
            f.write(binary_data.data)
        print(f"Downloaded and saved to {save_path}")
    except Fault as fault:
        print(f"Server error ({fault.faultCode}): {fault.faultString}")
    except Exception as e:
        print("Error downloading file:", e)

if __name__ == "__main__":
    server_host = input("Enter server IP/hostname (default: localhost): ").strip() or
"localhost"
    server_port = input("Enter server port (default: 9000): ").strip() or "9000"

    url = f"http://{server_host}:{server_port}"
    proxy = ServerProxy(url, allow_none=True)

    while True:
        menu()
        choice = input("Choose: ").strip()

        if choice == "1":
            list_files(proxy)
        elif choice == "2":
            upload(proxy)
        elif choice == "3":
            download(proxy)
        elif choice == "0":
            print("Bye!")
```

```
            break
        else:
            print("Invalid choice.")
```

4. Conclusion

The RPC file transfer system offers a simple and reliable way for clients to interact with the server through remote procedures. With clear functions for listing, uploading, and downloading files, the design is easy to implement, maintain, and extend for future use.