Service-Oriented Communication Technologies

A Comparative Study: SOAP · REST · GraphQL · gRPC

Salim LAKHAL — Nossa IYAMU

CSC8603 — TELECOM SudParis — February 2026

AGENDA

01 Introduction to SOA

02 SOAP / WSDL

03 REST

04 GraphQL

05 gRPC

06 Comparative Analysis

SOAP / WSDL

Contract-first: WSDL defines service before code

XML messaging: verbose but self-describing

WS-* extensions: security, transactions, reliability

Strongly typed: XSD validation at protocol level

```
<Envelope>
   <Body>
      <CalculateRequest>
         <operation>ADD</operation>
         <a>15.5</a>
         <b>24.3</b>
      </CalculateRequest>
   </Body>
</Envelope>
```

Best for: banking, healthcare, government, ERP integration

REST

Resource-oriented: URLs identify resources

HTTP semantics: GET / POST / PUT / DELETE

Stateless: no server-side session

HTTP caching: CDN-friendly, free scalability

```
POST /calculate HTTP/1.1
Content-Type: application/json

{"operation":"add","a":15.5,"b":24.3}

-- 200 OK --
{"result": 39.8}
```

Best for: public APIs, mobile backends, CRUD services

GraphQL

Single endpoint: POST /graphql for everything

Client controls fields: no over-fetching

Strongly typed schema: SDL, introspectable

Mutations for writes, **subscriptions** for real-time

```
query {
   calculate(a: 15.5, b: 24.3,
   operation: "add") {
      result
      operation
   }
}
-- response --
{ "data": { "calculate": {
"result": 39.8,
"operation": "ADD" }}}
```

Best for: mobile BFF, dashboards, developer platforms

gRPC

Binary protocol: Protobuf, ~3x smaller than JSON

HTTP/2: multiplexing, bidirectional streaming

Contract-first: .proto defines the API

4 modes: unary, server-stream, client-stream, bidi

```
service CalculatorService {
   rpc Calculate(Request)
      returns (Response);
   rpc CalculateStream(Request)
      returns (stream Response);
}
```

Best for: internal microservices, IoT, real-time pipelines

COMPARATIVE ANALYSIS

Protocol & Format

Technology	Transport	Format	Schema
SOAP	HTTP (any)	XML	WSDL mandatory
REST	HTTP	JSON	OpenAPI optional
GraphQL	HTTP	JSON	SDL mandatory
gRPC	HTTP/2	Protobuf binary	.proto mandatory

COMPARATIVE ANALYSIS

Performance & Security

Technology	Performance	Security	Caching
SOAP	Slow (XML)	WS-Security, SAML	No native
REST	Good (JSON)	TLS, OAuth2, JWT	HTTP cache / CDN
GraphQL	Good	OAuth2 + directives	Complex
gRPC	Best (binary)	mTLS standard	No native

WHEN TO USE WHAT

SOAP

Regulated industries: banking, healthcare, legacy ERP. When WS-Security or non-repudiation is legally required.

REST

Public APIs, mobile apps, CDN-cached content, third-party integrations.

GraphQL

Diverse clients needing different data shapes. Mobile BFF, developer platforms, complex data graphs.

gRPC

Internal service-to-service calls. Performance-critical paths, streaming data, polyglot microservices.

LIVE EXAMPLES — THIS PROJECT

SOAP / WSDL

Python + zeep library

```
python3 examples/soap/client.py
```

REST

Python + Flask framework

```
python3 examples/rest/client.py
```

GraphQL

Node.js + Express + graphql-http

```
node examples/graphql/client.js
```

gRPC

Python + grpcio library

```
python3 examples/grpc/client.py
```

KEY FINDINGS

1. **There is no best technology**
   Each solves a different set of constraints.

2. **SOAP is not obsolete**
   WS-* compliance is legally required in banking and healthcare.

3. **gRPC is surprisingly fast**
   Protobuf encoding is 3-5x smaller than equivalent JSON.

4. **GraphQL shifts complexity**
   Less over-fetching but requires DataLoader to avoid N+1 queries.

Questions?

CSC8603 — TELECOM SudParis

Salim LAKHAL — Nossa IYAMU — February 2026