REST VS Graphql VS gRPC

Aspect	REST	GraphQL	gRPC
How it talks	HTTP/1.1	HTTP/1.1 / 2	HTTP/2 only
Data format	JSON / XML	JSON	Protocol Buffers (binary, tiny)
Endpoints	Many (/users, /posts)	One (/graphql)	Service methods (GetUser())
Fetching data	Can over/under-fetch	Ask exactly what you need	Very efficient
Speed	Okay	Usually faster (less round trips)	Super fast, low latency
Errors	HTTP status codes	Errors in response body	Status + rich metadata
Caching	Native HTTP caching	Harder, custom needed	Mostly app-level
Versioning	New endpoints (v1, v2)	Schema changes, rarely versioned	Version via proto files
Streaming	Workarounds (SSE, WebSocket)	Subscriptions	Built-in streaming (uni + bi)
Security	HTTPS, OAuth, JWT	Same as REST	TLS, mTLS, JWT
Schema/ Contract	Optional (Swagger/ OpenAPI)	Strongly typed schema	Required proto file
Typing	Loose	Strong	Strong

		(schema-driven)	(compiled contracts)
Tooling	Mature everywhere	Growing, IDE support	Strong codegen tools
Learning curve	Easy	Medium	Steep
Mobile use	Often heavy payloads	Lighter (query what's needed)	Great (compact payloads)
Binary data	Base64 hacks	Not natural	Native support
Community	Huge, standard for APIs	Big with frontend-heavy apps	Growing in microservices
Docs needed	Often a must	Schema is self-documented	Proto is self-documenting
Error visibility	Clear HTTP codes	Sometimes hidden in response	Very structured
Real-time	Not native	Subscriptions	Native bidirectional streams
Maturity	Decades old, stable	2015+, still evolving	2016+, rapidly growing
Scalability	Easy horizontal scaling	Works well, but complex infra	Built for large distributed systems
Setup	Simple, no special tools	Needs GraphQL server	Needs proto + codegen
Best for	CRUD APIs, public endpoints	Complex UIs, mobile/web	High-perf microservices, IoT, internal APIs