When it comes to choosing between an event-driven API and a request-response API for a social network, first of all, one needs to consider the interaction nature along with the use cases each of them serves best. Here is a simple example that compares these two approaches, arguing for each:

1. **Event-Driven API**

In general, an event-driven API is based on asynchronous communication. Events (such as notifications, messages, or status updates by users) are pushed to interested parties with no need for the interested parties to constantly poll for updates.

Event-Driven API Arguments

- Real-Time Interaction: Most social media platforms require notifications in real time to let users know when there are new messages, likes, comments, etc. Event-driven APIs provide very good functionality in regard to pushing this information right after an update happens.
- Reduced Server Load: The user generally performs continuous requests to the server to check for changes, but the server will only notify upon the availability of new data. This decreases the load on the server and on the network as well.
- Scalability: Event-driven architecture, while the platform grows, can handle high volume user interactions-thinking message exchanges or live video comments better instead of constant client-server requests.
- User Experience: Because of immediate updates, the interaction of users becomes smoother, and thus the experience of the users is far more interactive.

Where Are Event-Driven APIs for Social Media Used?

Real-time chat/messaging: Notifications and updates are pushed directly to the user every time a new message comes up.

Live notifications: Immediately update if somebody likes or shares your post, or comments on the same.

Streaming and live broadcast: User reaction, comments, and views continue updating in real time during live streams.

2. **Request-Response API**

The client requests the server for some data or an action in a request-response model, and it returns the data accordingly. This is a synchronous form of communication, which is quite customary with REST APIs.

Arguments in favor of Request-Response API:

- Simple and Traditional: Request-response is more comprehensible, easier to develop, and implement for simpler actions that do not need real-time updates.
- Predictability: Since the client is the one to send requests for everything, responses happen strictly when needed. That's because it is always known which request led to which response, and handling and debugging certain requests becomes easier.
- Resource Efficiency in Certain Actions: For those kinds of actions that occur less often-for example, lookup users or uploading a profile picture-this model is much more resource-efficient compared to establishing an event-based flow.

Popular Use Cases of Request-Response APIs in Social Media:

Profile and content updates: User update his/her profile or posts some content, which does not necessarily need to be in real time; hence, it can be synchronous. Retrieving the feed: When a user pulls his feed, he asks for the latest content, which again can be request-response. Retrieve other users: Information about other users-for example, search users, fetching historical messages of chat, or fetching some analytics about posts. Summary and Recommendation

For social media applications, the hybrid approach would be the best.

API Event-driven for real-time features such as notifications, messaging, and live content, while request-response for regular actions that are less time-sensitive, such as profile updates, loading of feeds, and functions of a search nature. This balance ensures the platform is both efficient and responsive in real time where it needs to be and simple where synchronous communication will do.