RIAs with Comet and Critical Updates in Enterprise Environments

Emil Ong
Chief Evangelist
Caucho Technology
emil@caucho.com

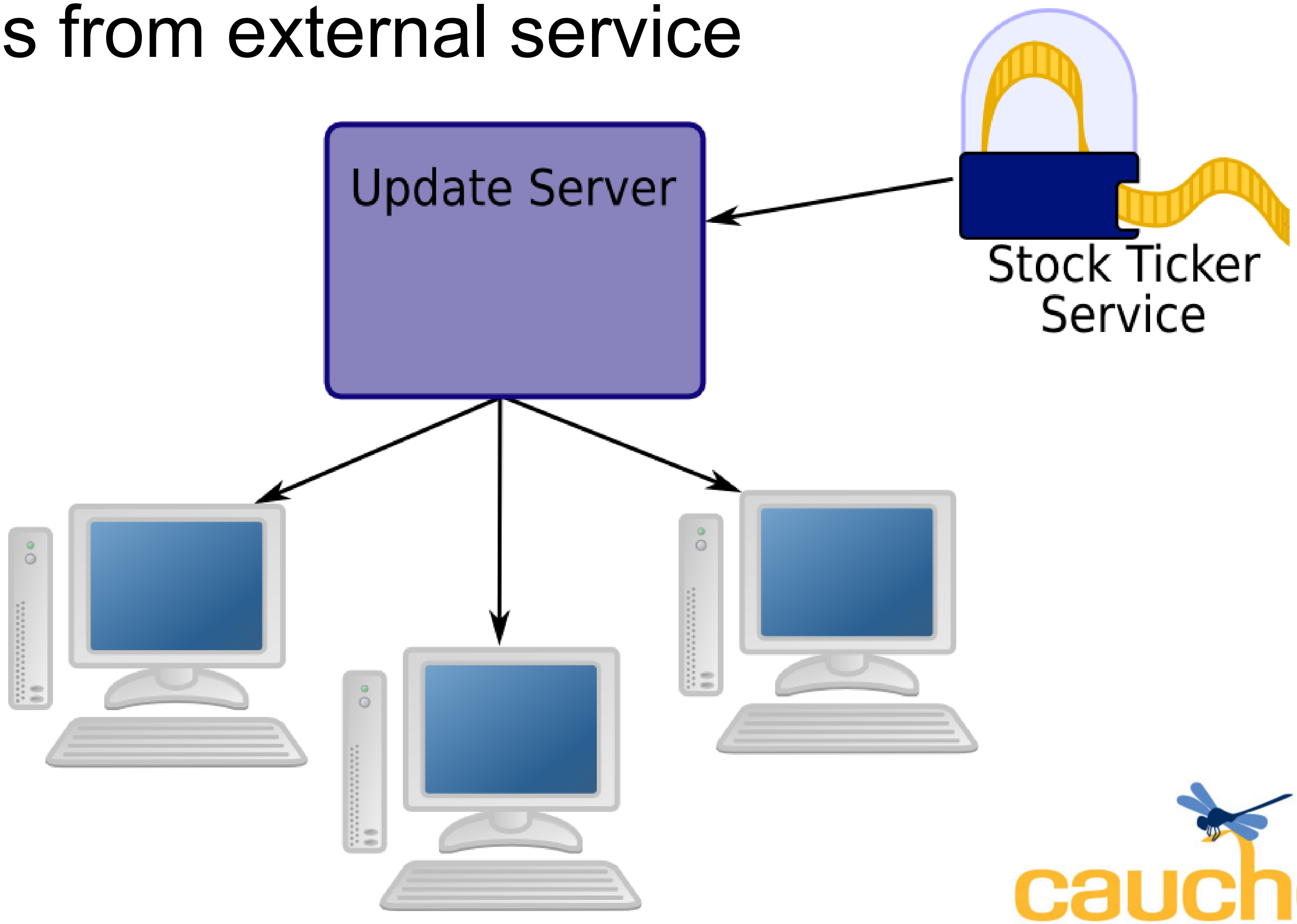
What are critical updates?

- Updates sent from the server to the client
- Time-sensitive updates
- Sources include
  - Updates from a service external to the server
  - Updates from users
  - Updates from users **and** the server
  - Updates from internal network sources

caucho
reliable open source

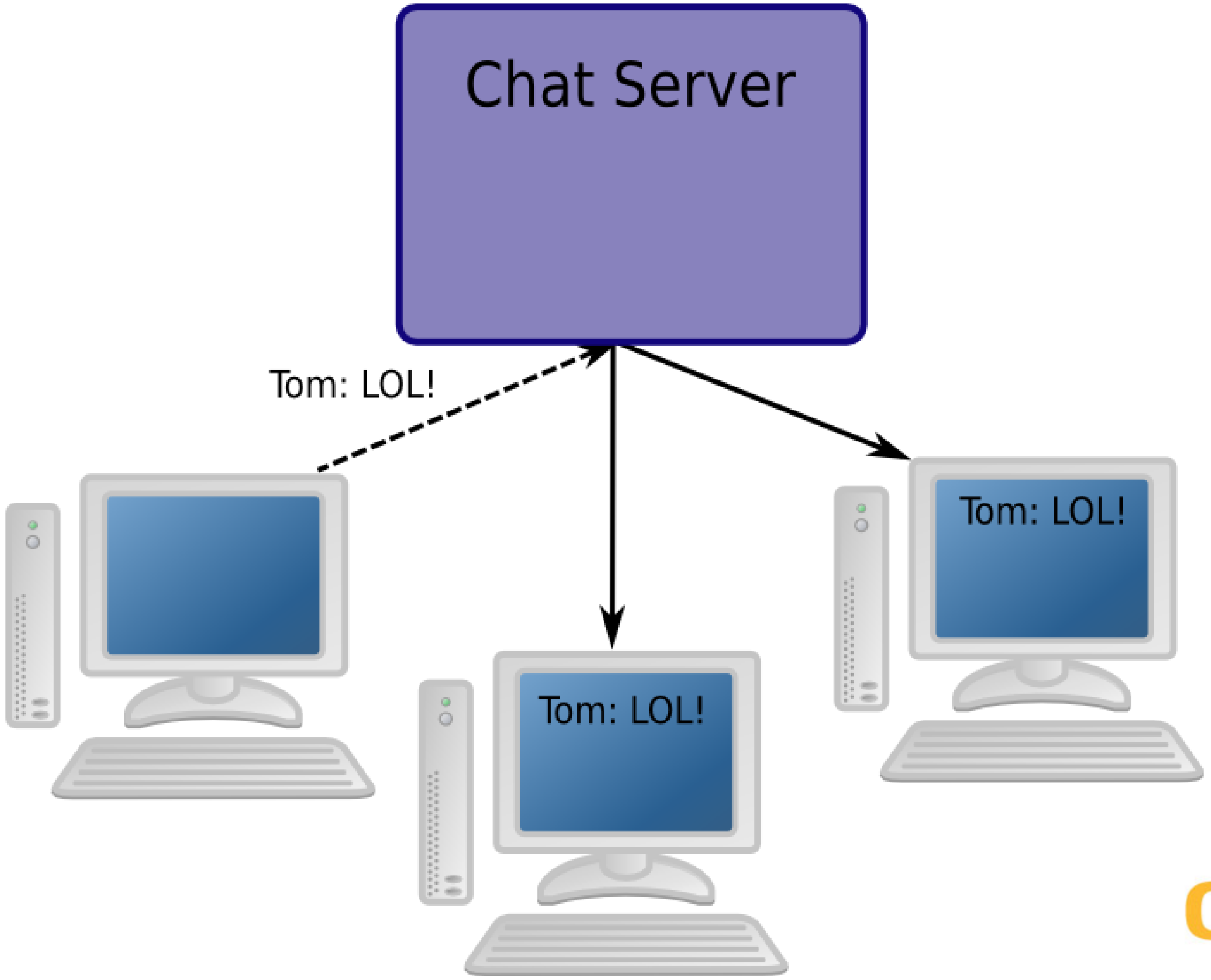
Applications with Critical Updates: Financial

- Updates from external service



Applications with Critical Updates: Chat

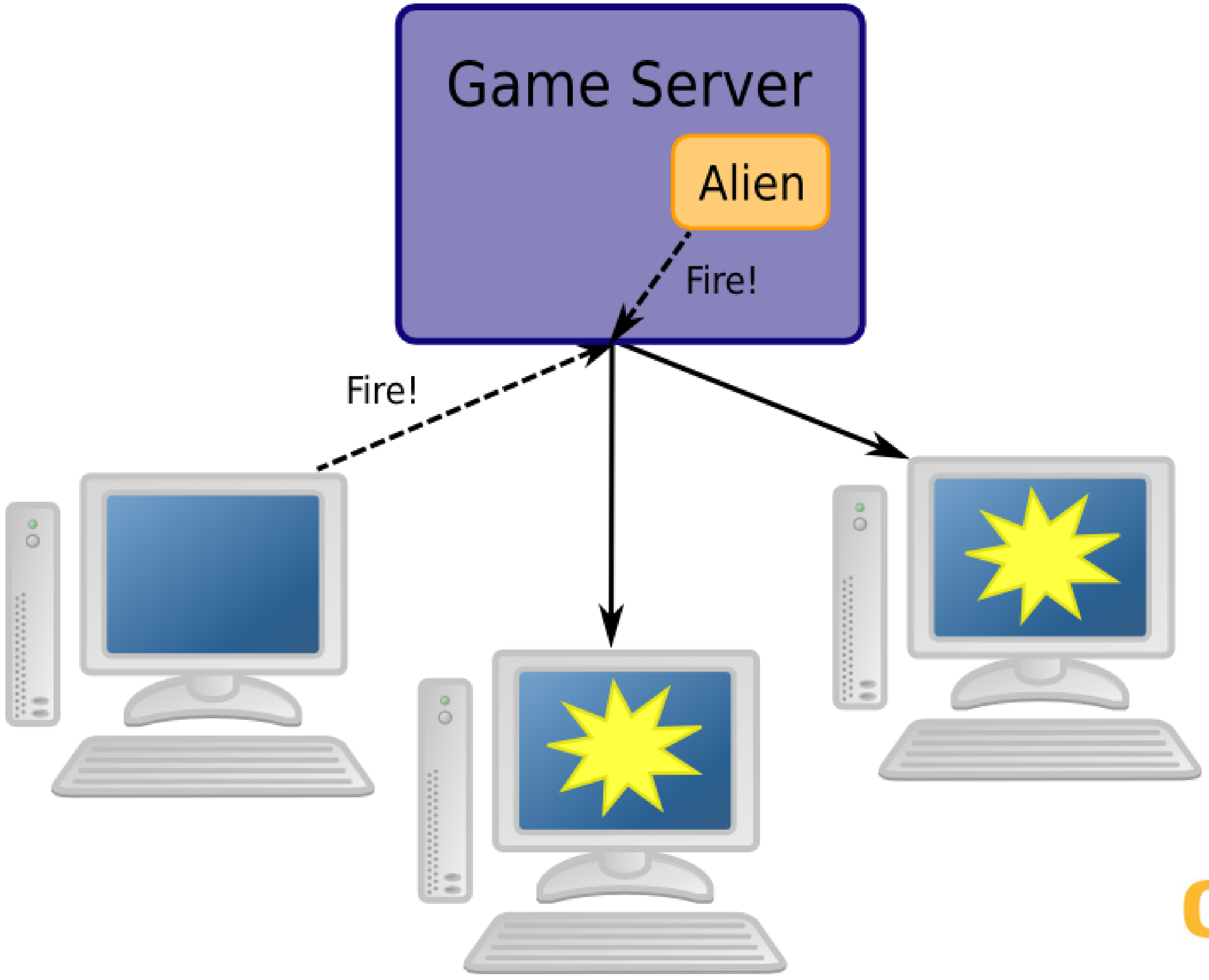
- Updates from other users



caucho
reliable open source

Applications with Critical Updates: Games

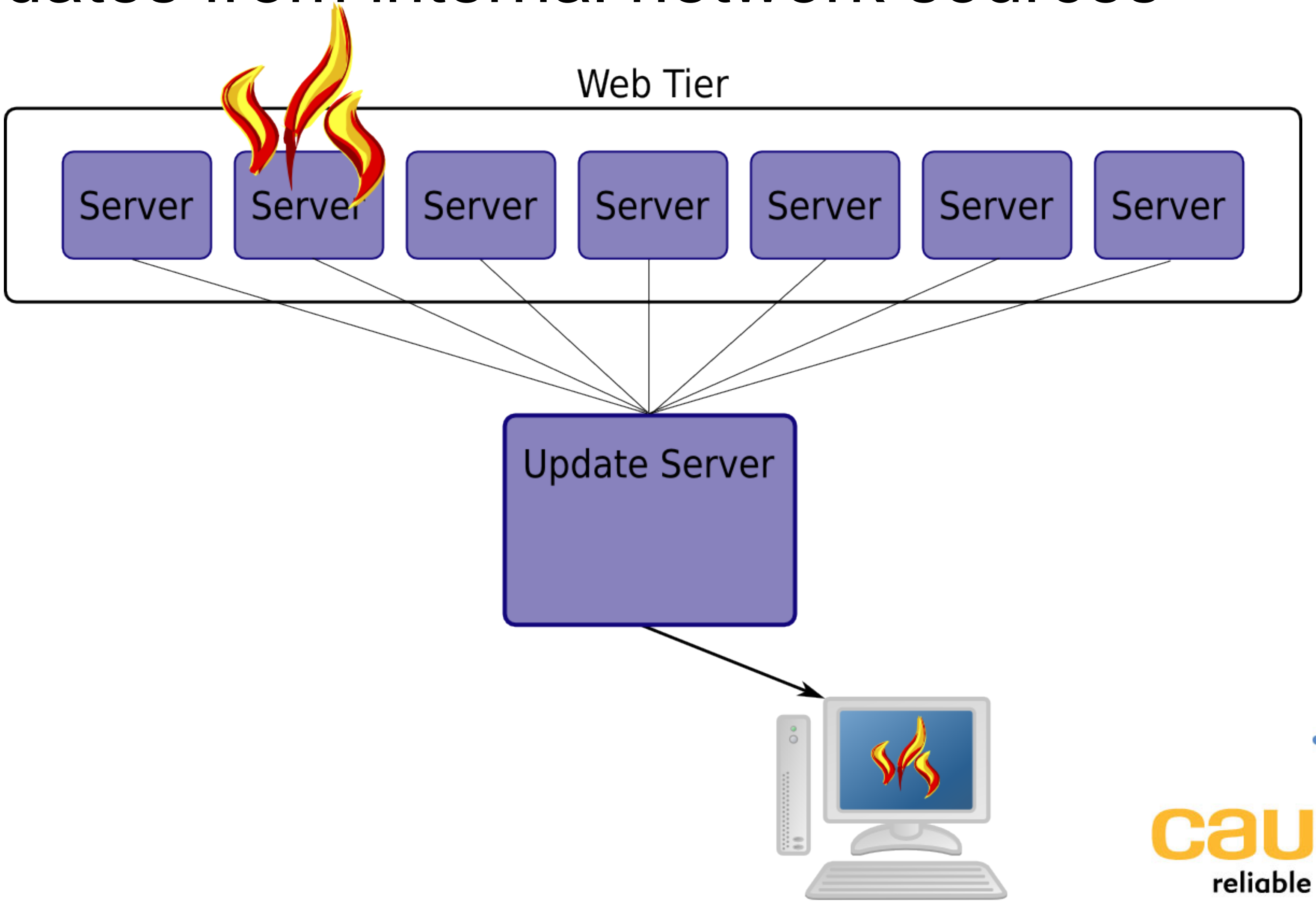
- Updates from other users and server



caucho
reliable open source

Applications with Critical Updates: Network administration

- Updates from internal network sources



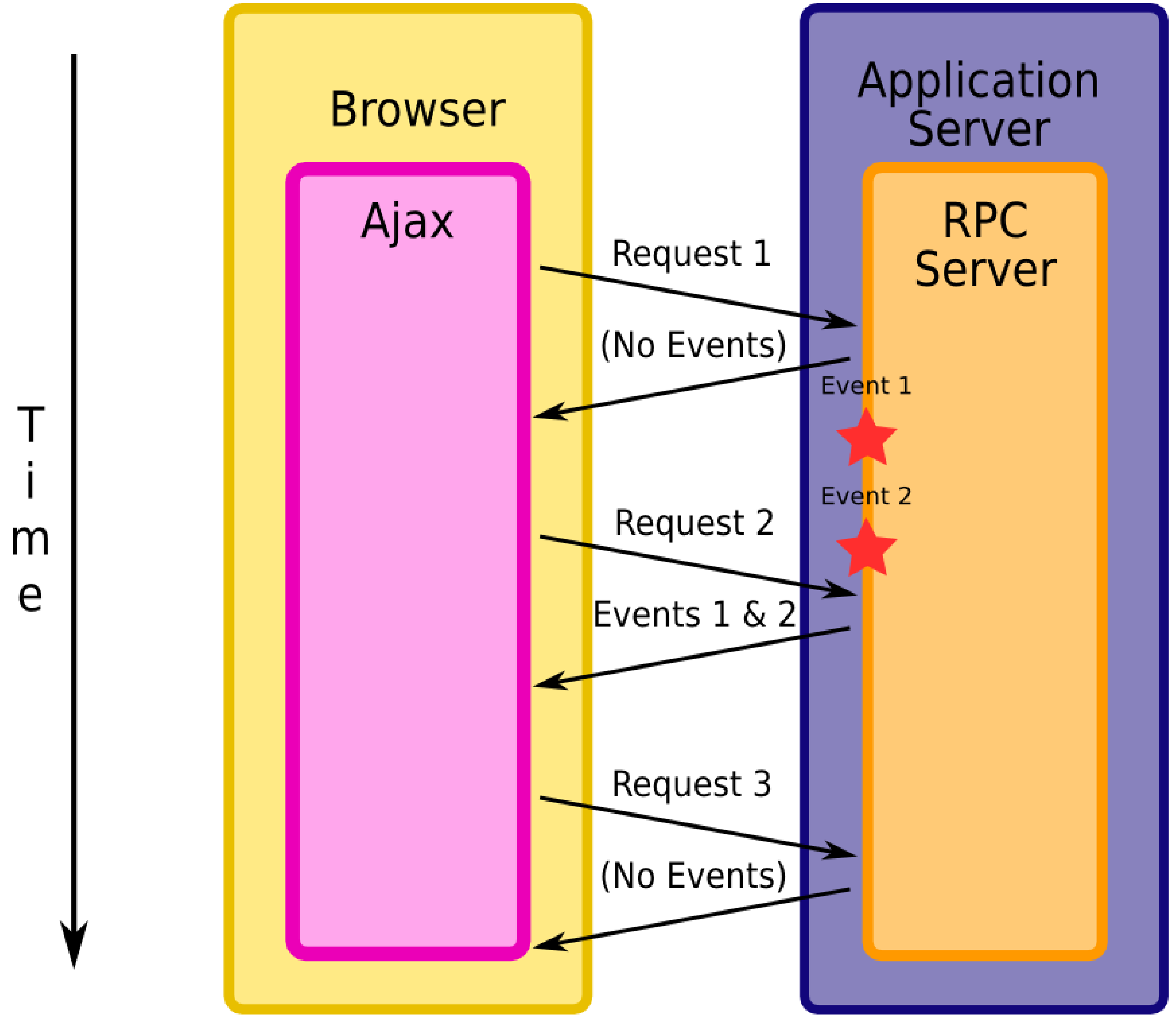
caucho
reliable open source

Approaches to Critical Updates: Polling

- Polling
  - Client periodically checks with server for new updates using RPC or web services
  - **Poll too little:** May get updates too late, may get event "clumping"
  - **Poll too much:** May use server resources unnecessarily when there are no new updates
  - **Too much or too little depends on the events – the client has no way of knowing!**

caucho
reliable open source

Problems with Polling



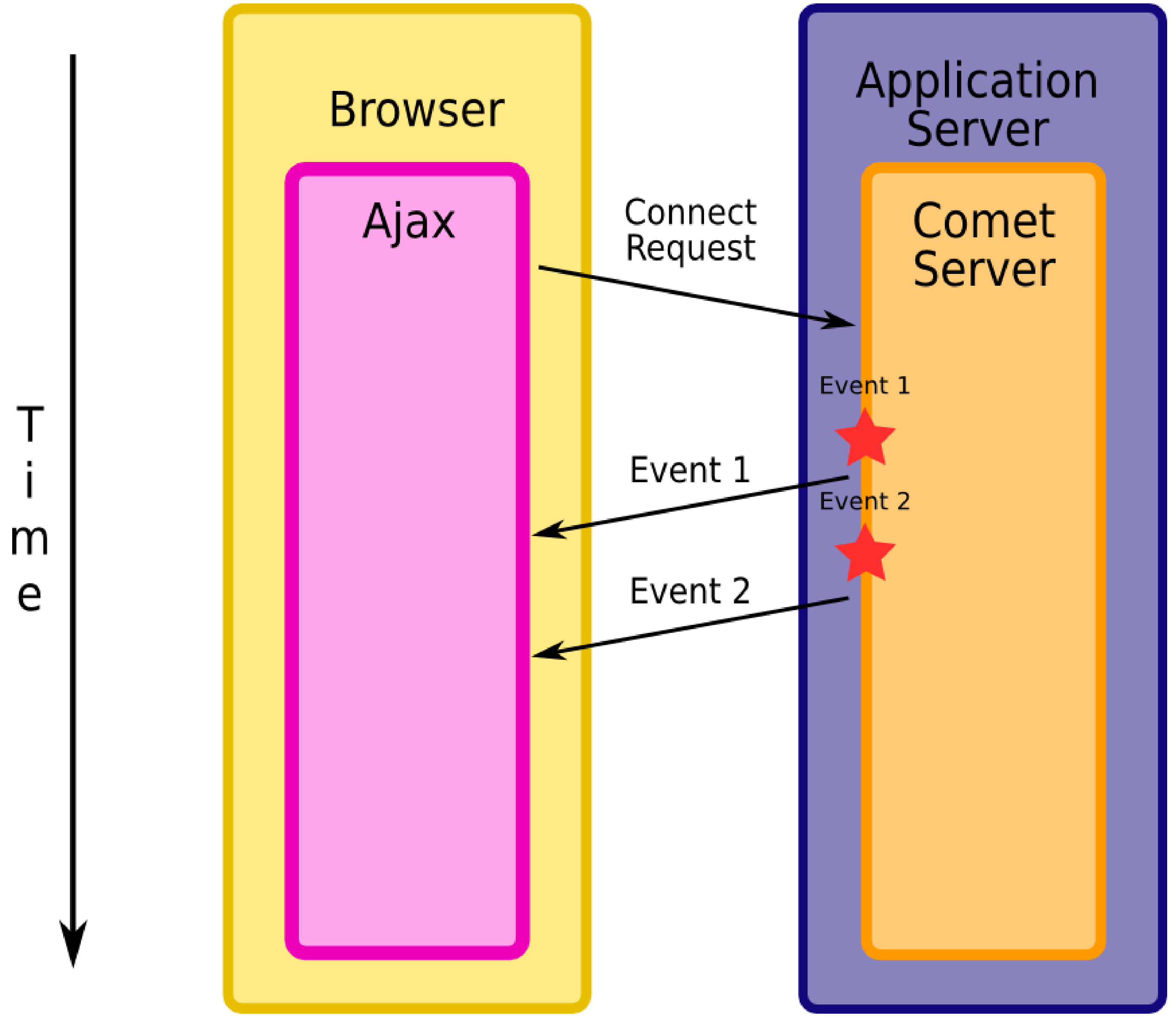
caucho
reliable open source

Approaches to Critical Updates: Comet

- Comet (a.k.a server-push, reverse Ajax)
  - Client makes initial registration request
  - Server sends updates to client
  - Updates get to client as they happen
  - Requires persistent connection

caucho
reliable open source

Comet in action



caucho
reliable open source

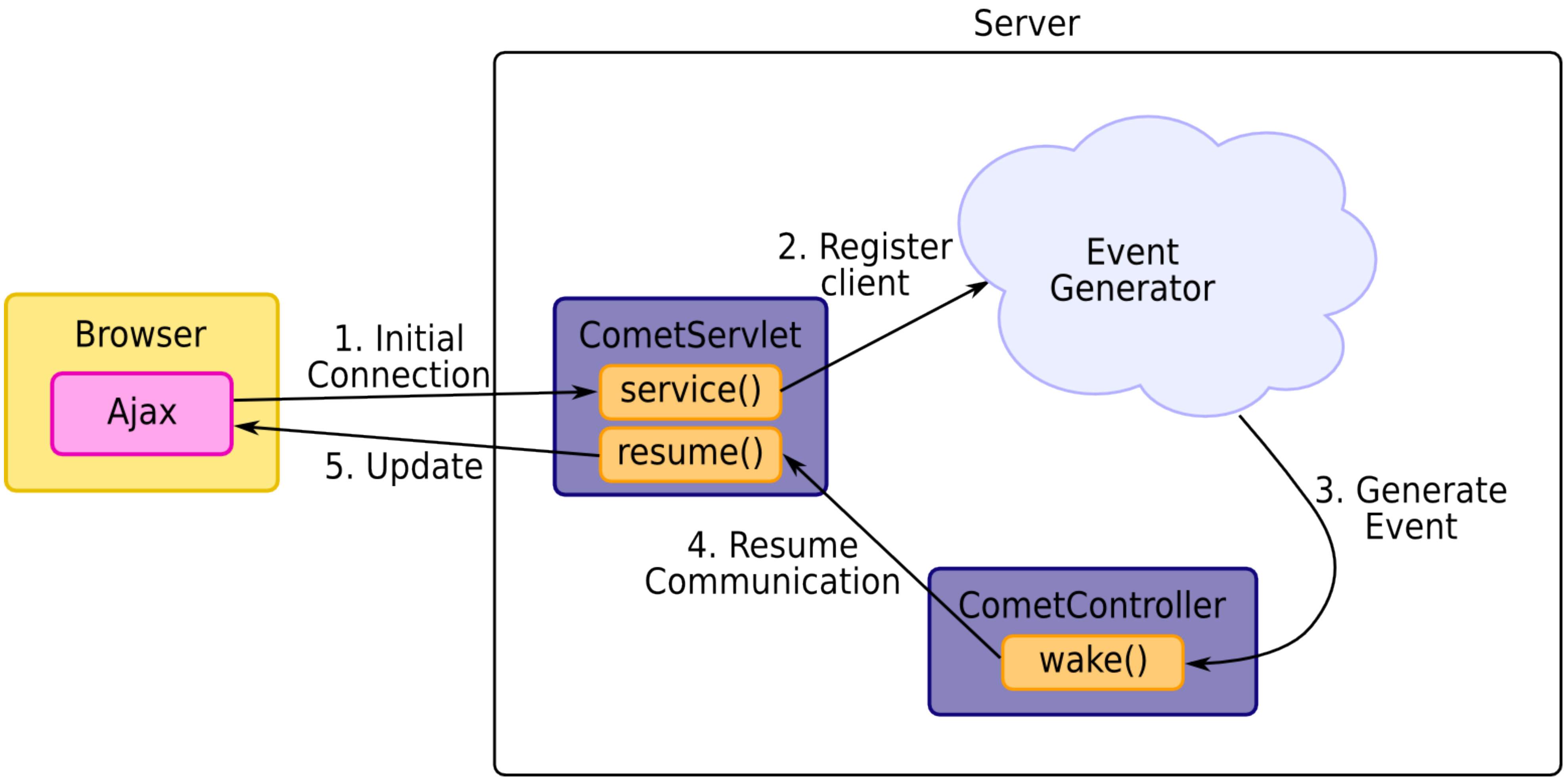
Developer-focused Comet

- Comet is a new way of thinking for many application developers
  - Resin Comet makes the transition easier with an evolutionary API based on Java Servlets
- With a traditional server, you may get excessive threads or have to manage thread pool manually
  - Resin Comet handles the thread management for you

Resin's Comet API and Infrastructure

- Similar to Java Servlet API
- Resource management is automatic:
  - Threads are pooled in the background
  - Developers can worry about client state instead of threads
- Two main API classes:
  - `CometServlet` – handles communication with client
  - `CometController` – encapsulates per-client state

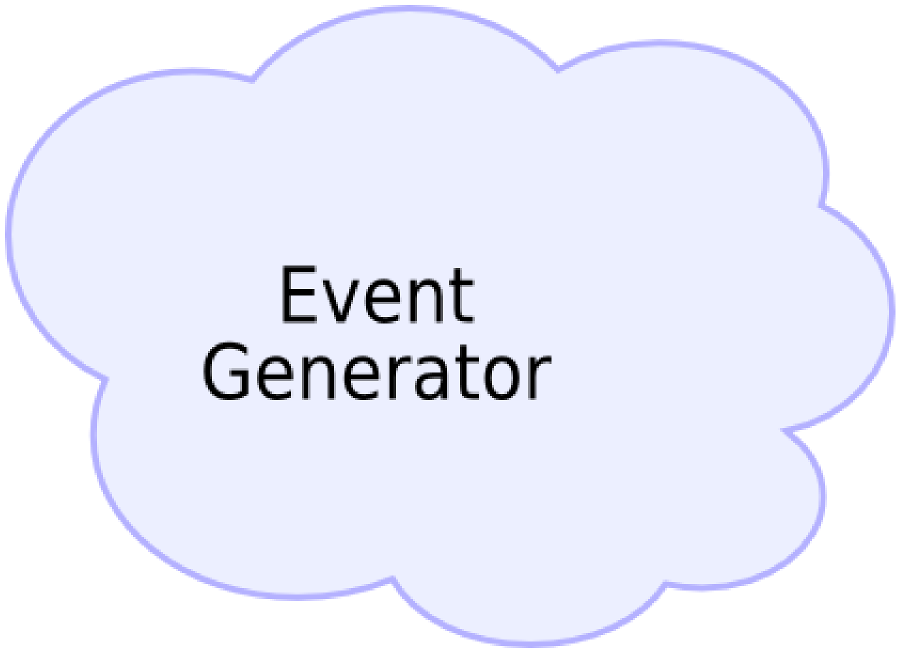
Basic Comet Architecture



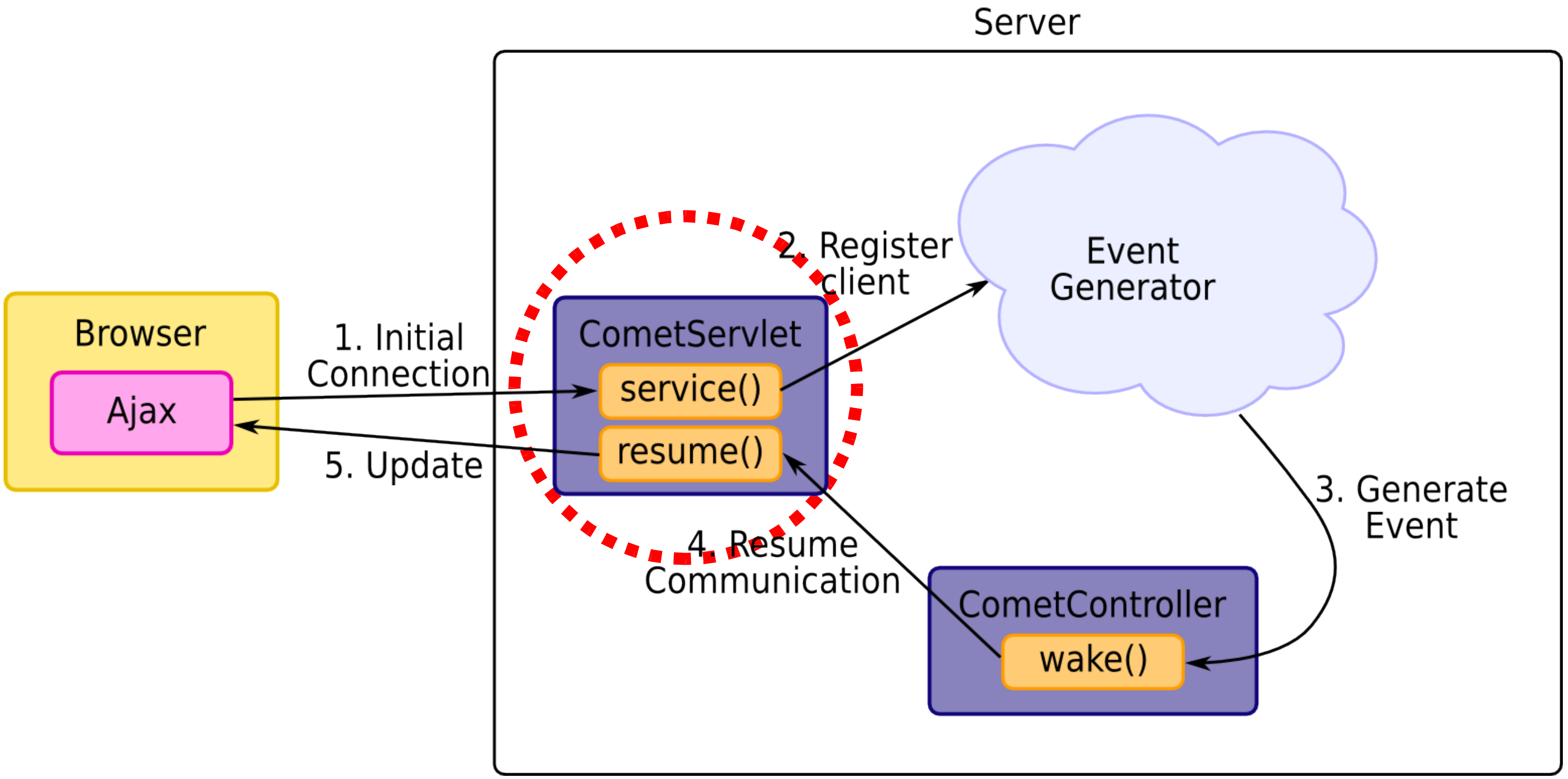
caucho
reliable open source

Event Generator: Implementation Examples

- JMS messages
- SEDA pipeline
  - Mule
- Network monitors
  - SNMP
  - Firewall
- Web services
  - SOAP



Basic Comet Architecture



caucho
reliable open source

CometServlet

- `CometServlet` – client communication
- Explicit code separation between initial and subsequent communication
  - Handle the initial connection

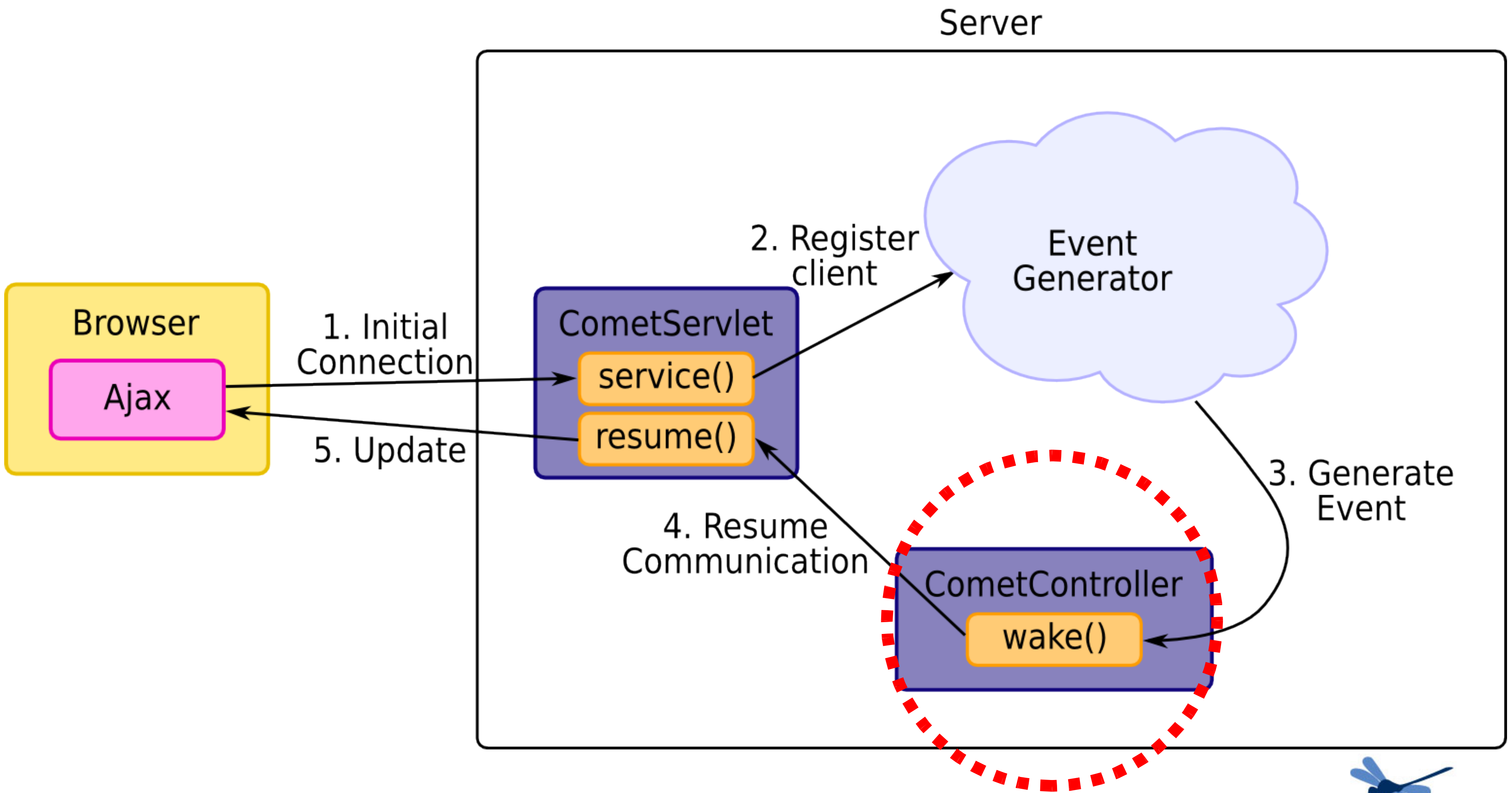
    -
      ```
      service(ServletRequest,ServletResponse,
              CometController)
      ```
  - Send updates to the client

    -
      ```
      resume(ServletRequest,ServletResponse,
             CometController)
      ```

CometServlet

- Evolutionary API approach:
  - Uses Servlet as a foundation
  - Standard Servlet idioms still apply
  - Matching Filter API (adds `doResume()`)
- Idea of continuing communication built into API
  - Have formalized `resume()` rather than simply holding open the client stream

Basic Comet Architecture



caucho
reliable open source

CometController

- `CometController` - manages per-client state
  - State maintenance
    - `getAttribute(String)`
    - `setAttribute(String, Object)`
  - Send updates
    - `wake()`

CometController

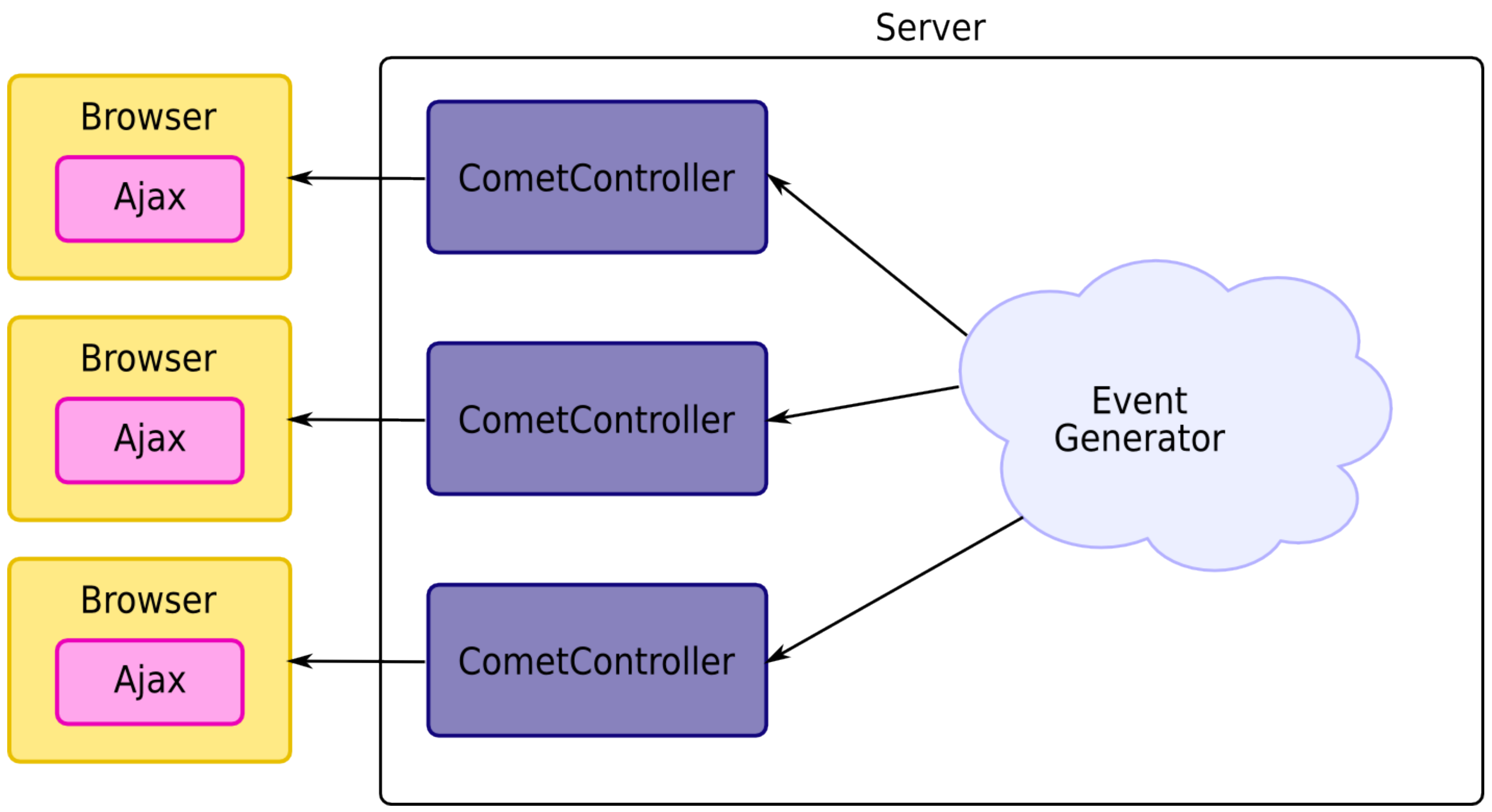
- New concurrency primitive
- Essentially creates a blocking queue to the client
- Think: `java.util.concurrent`
- Gives a handle to the client directly to the service
- Client can be a “stage” in a SEDA

CometController offers Flexibility

- Allows complex interactions with clients
  - Broadcast
  - Unicast
  - Subscription-based

CometController Flexibility: Broadcast

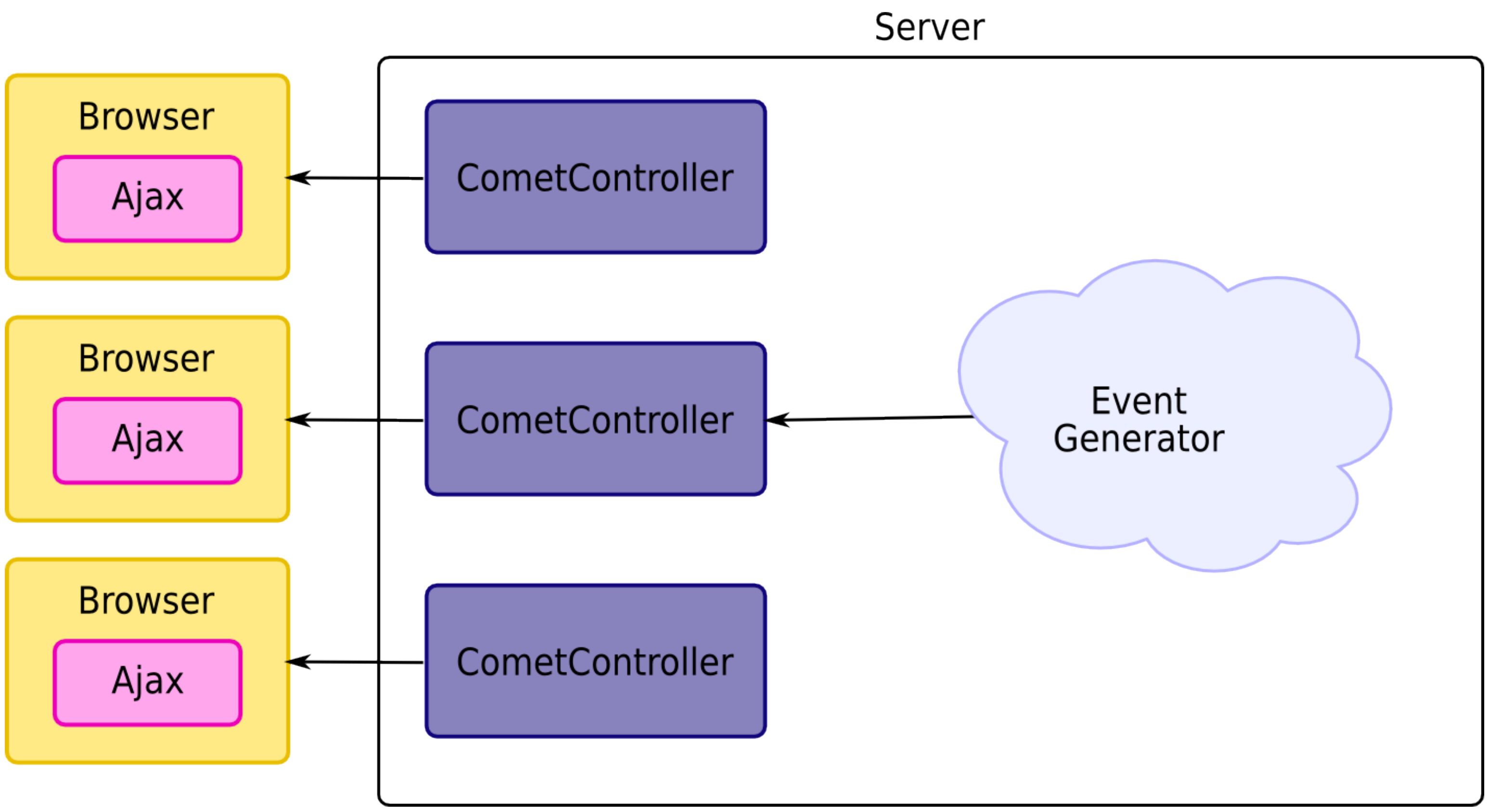
- Chat room



reliable open source

CometController Flexibility: Unicast

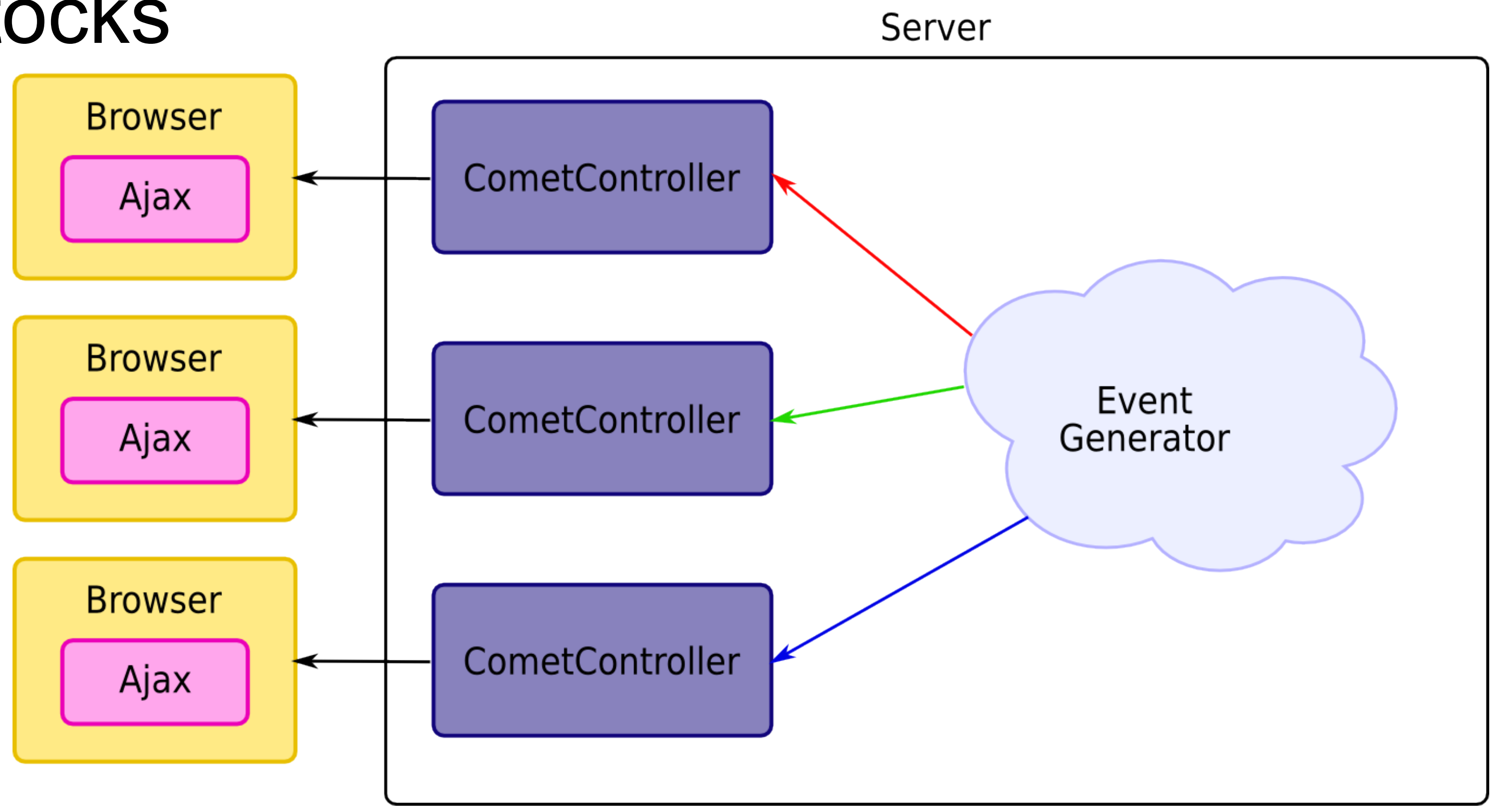
- Instant messaging



caucho
reliable open source

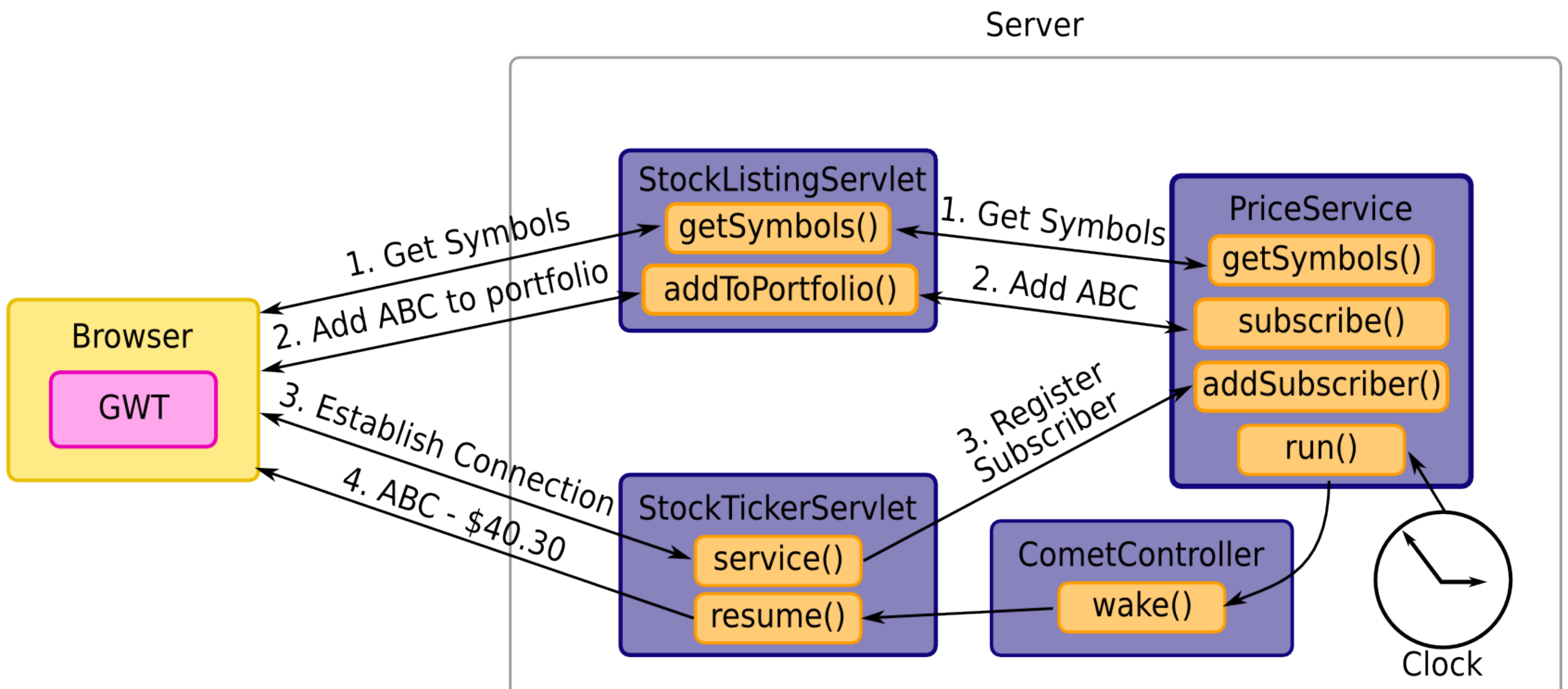
CometController Flexibility: Subscription-based

- News
- Stocks



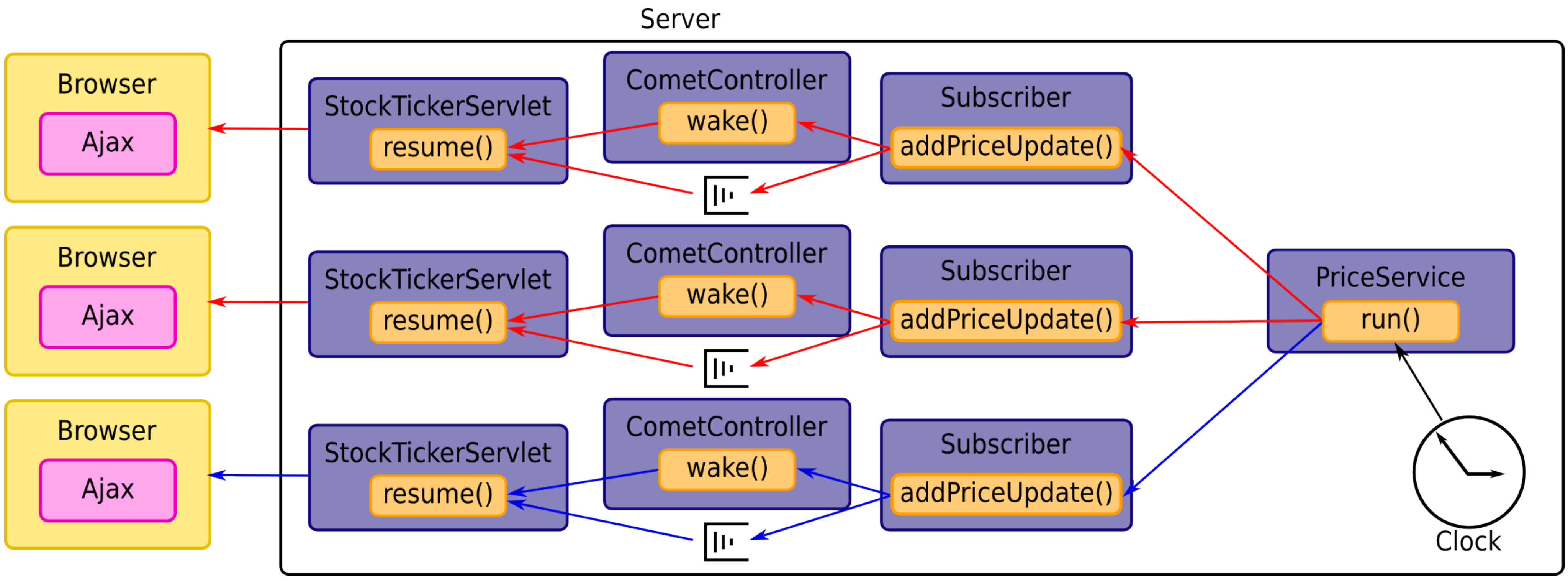
Demo

Demo Architecture



caucho
reliable open source

Demo Architecture (cont.)



caucho
reliable open source

Conclusion

- Comet approaches are becoming necessary for certain classes of applications
- Resin's Comet API and infrastructure:
  - Are familiar to Java Servlet developers
  - Offer a concurrency primitive that is based on client connections
  - Remove the need to worry about threads at development time

Where to find Resin

http://www.caucho.com/

Questions?

caucho
reliable open source

Thank you!

caucho

reliable open source