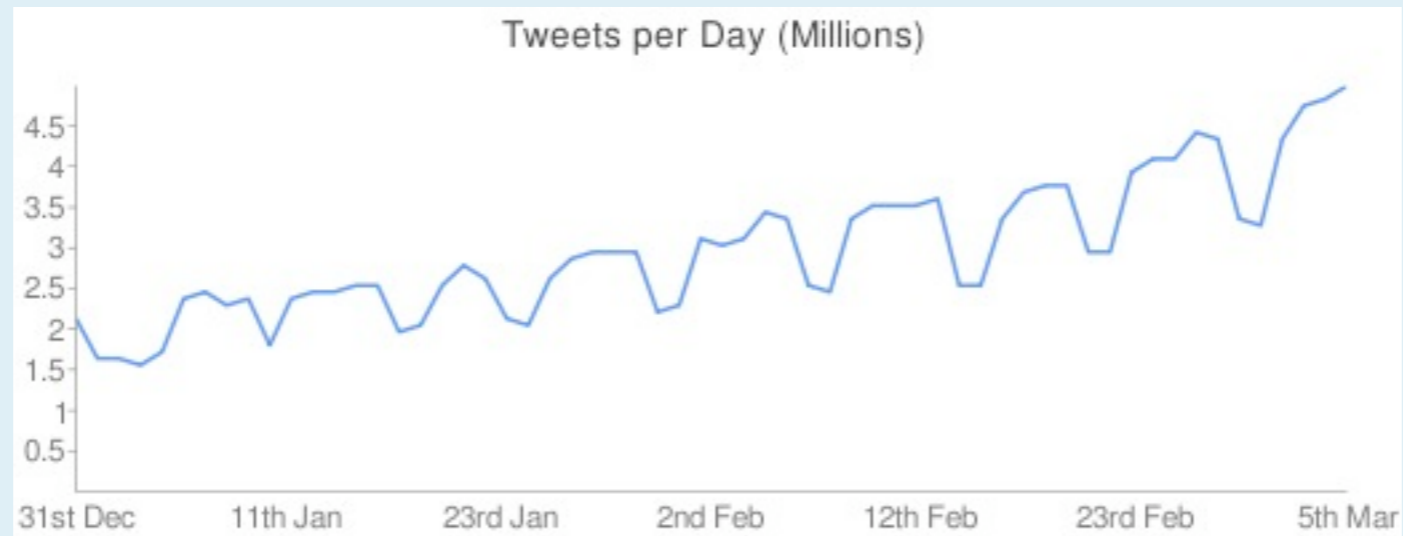
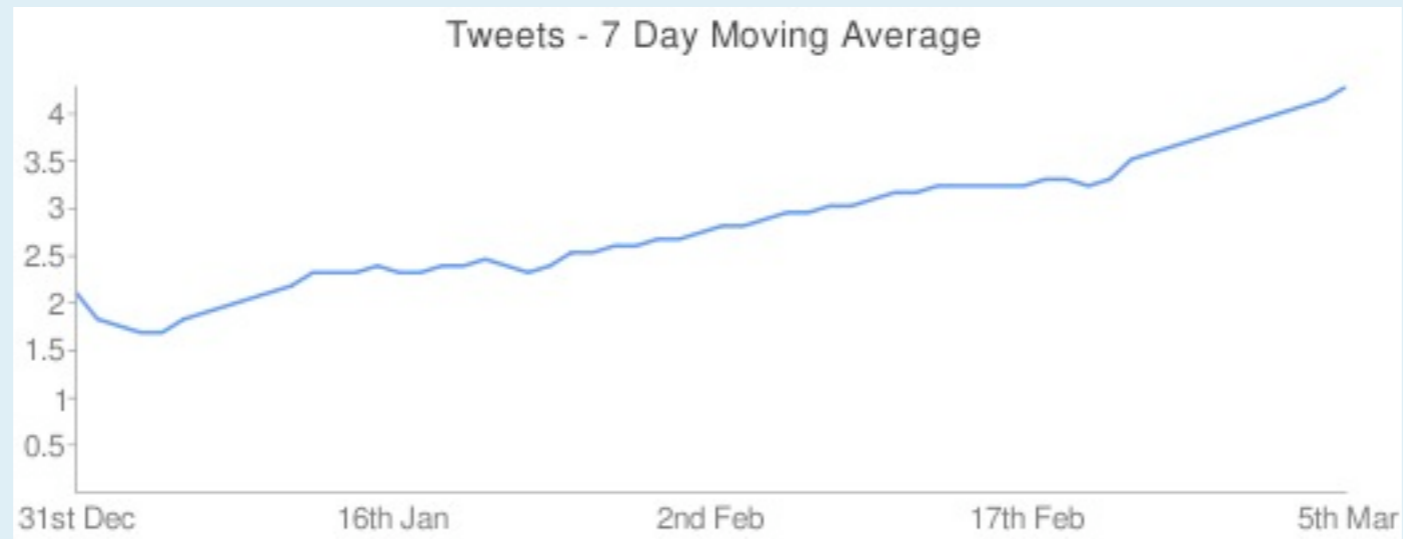




The Next 10 Years

Information growth is exponential

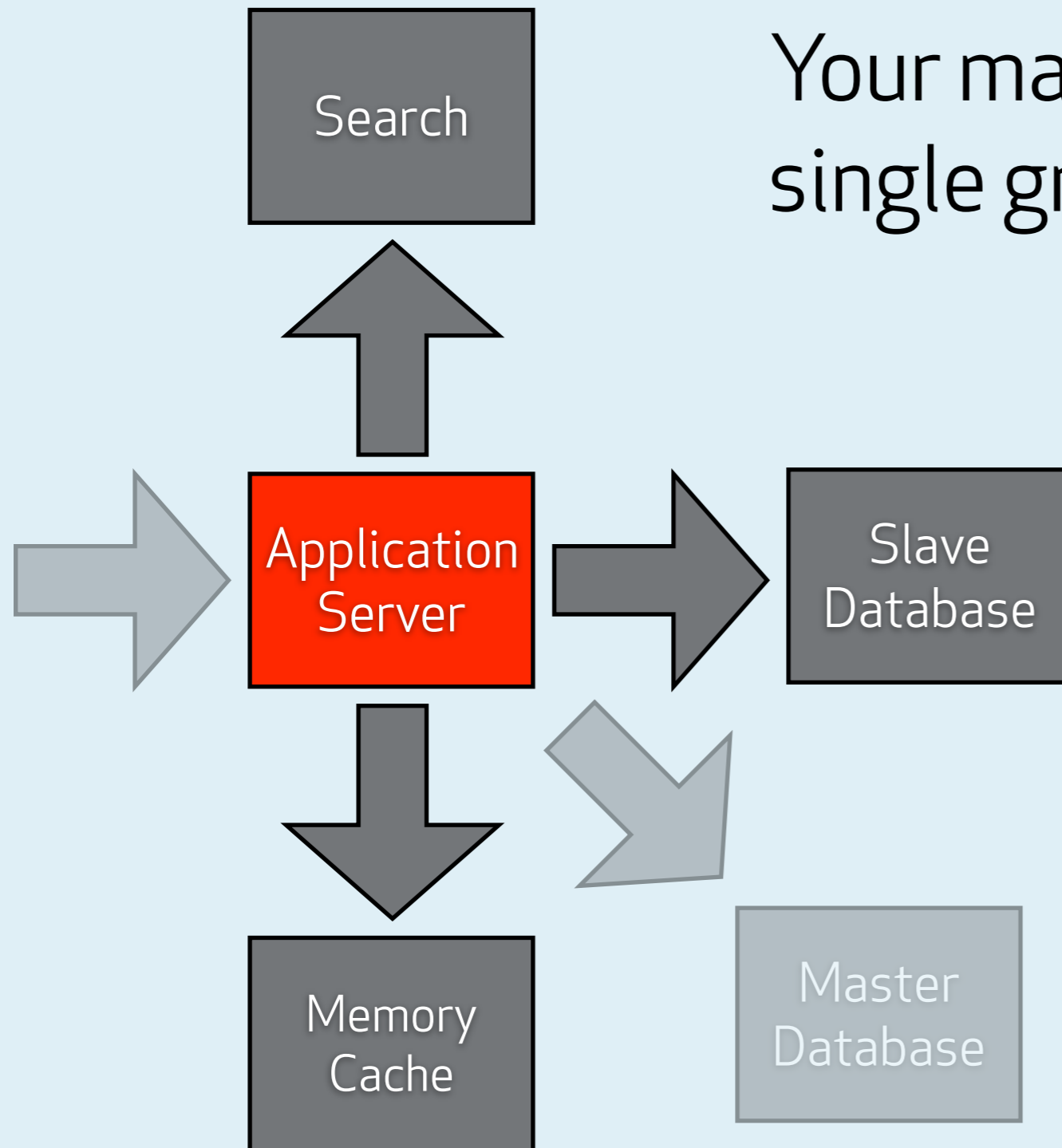


You may remember the following
slide from my cluster
management presentation.



Offload from the master database

Your master database is the single greatest limitation on scalability.

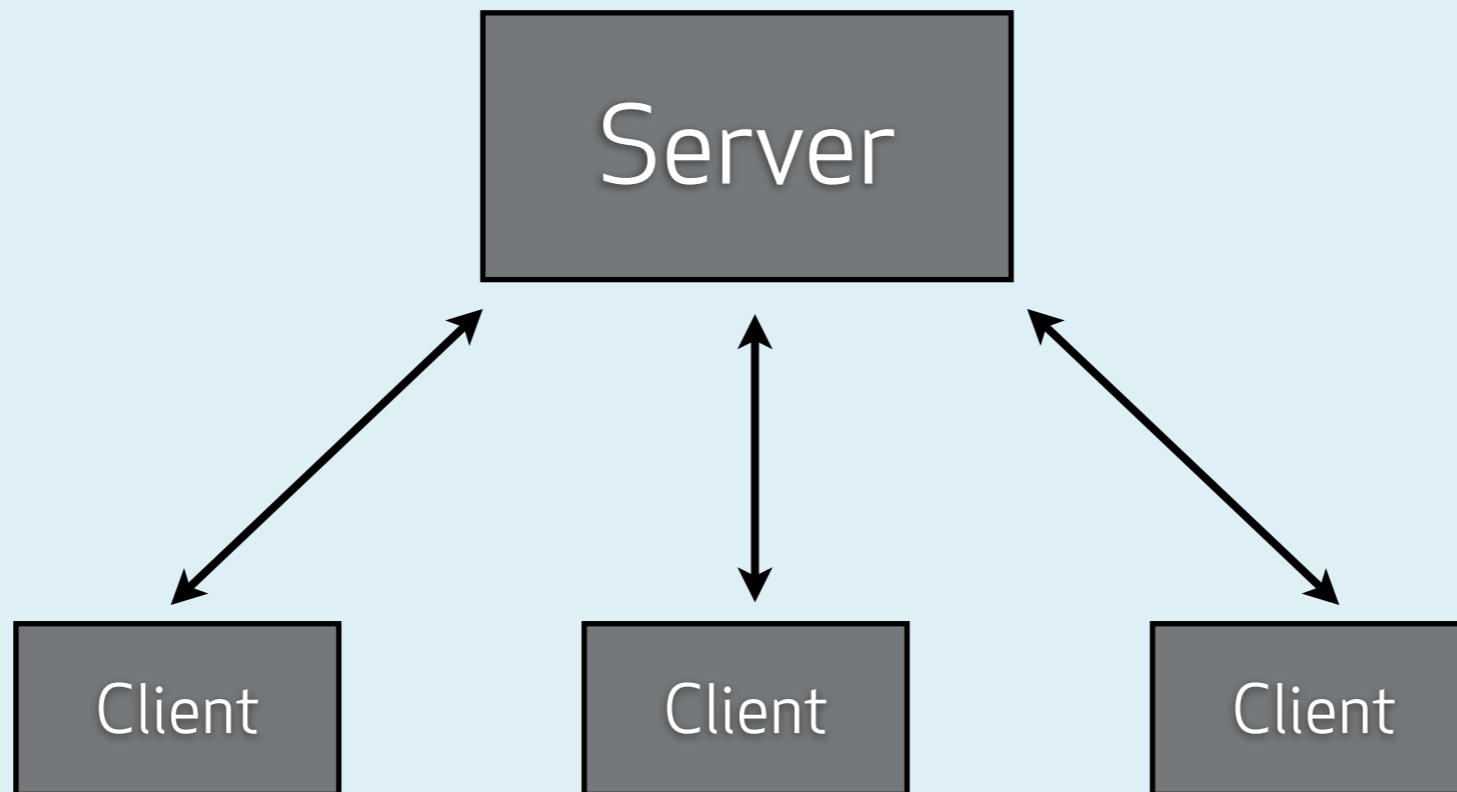


I'll say it again...

Your master database is the
single greatest limitation on
scalability.



Now: clients and servers



It's hard to scale

Our consistency models create **contention**.

Disconnected operation is **broken** or impossible.

Synchronization models create **duplicates** and **lose data**.

Data models **punish** work to partition and distribute data.

Databases have fostered the **myth** that applications should remain ignorant of conflicts and inconsistency.



The problem

With an **explosion** of devices, we can't afford bottlenecks.

Yet, our **assumptions** of consistency create bottlenecks.

We need to **change** our assumptions.



The myth of needing consistency

Traditionally, applications are designed to run on top of an **ACID-compliant** database.

This sort of consistency is really just an **abstraction**.

Consistency should be a **question**, not an assumption.



When do we need immediate consistency?

Pretty infrequently, it turns out.

Event	For You	For Others
You post a comment.	✓	✗
You change your password.	✓	✗
You post a project.	✓	✗
You promote a post to the home page.	✓	✗
You view recent posts in the Tracker.	✓	✗



Understanding consistency

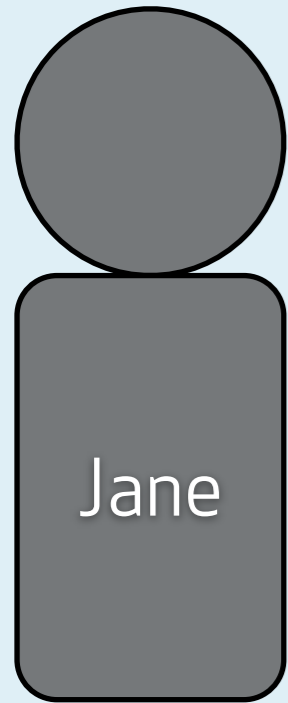
We don't need **immediate** consistency for everyone.

We need **eventual** consistency.

We also need immediate consistency for a user's **own actions**.



Consistency by perspective



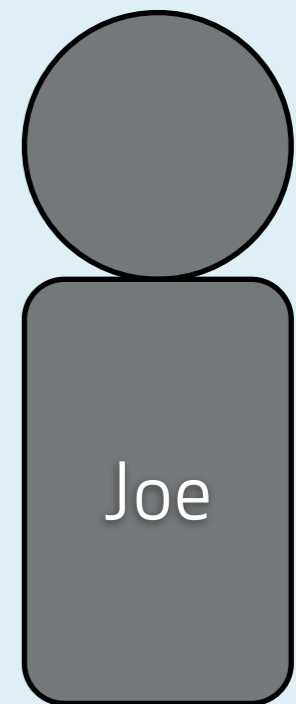
1:00pm

1:02pm

1:01pm

1:03pm

1:02pm



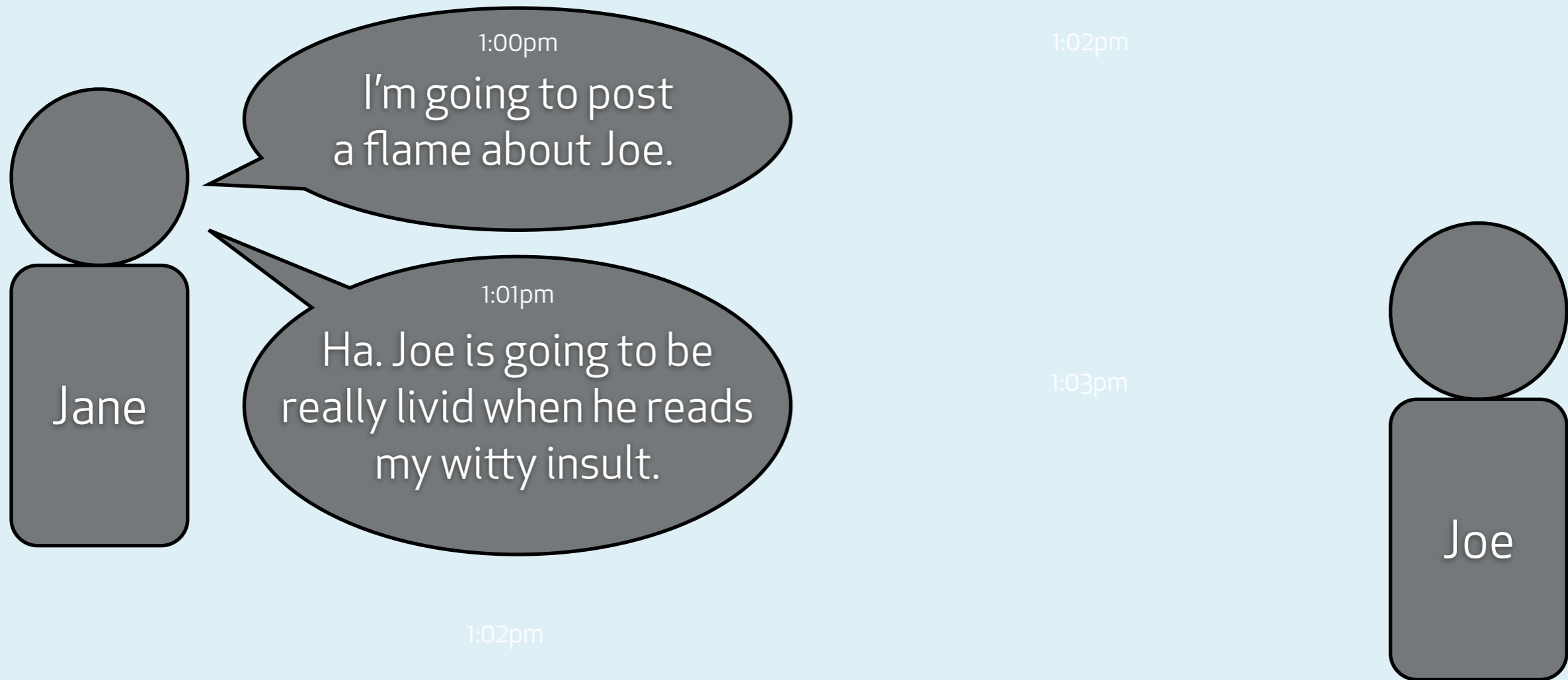
1:05pm



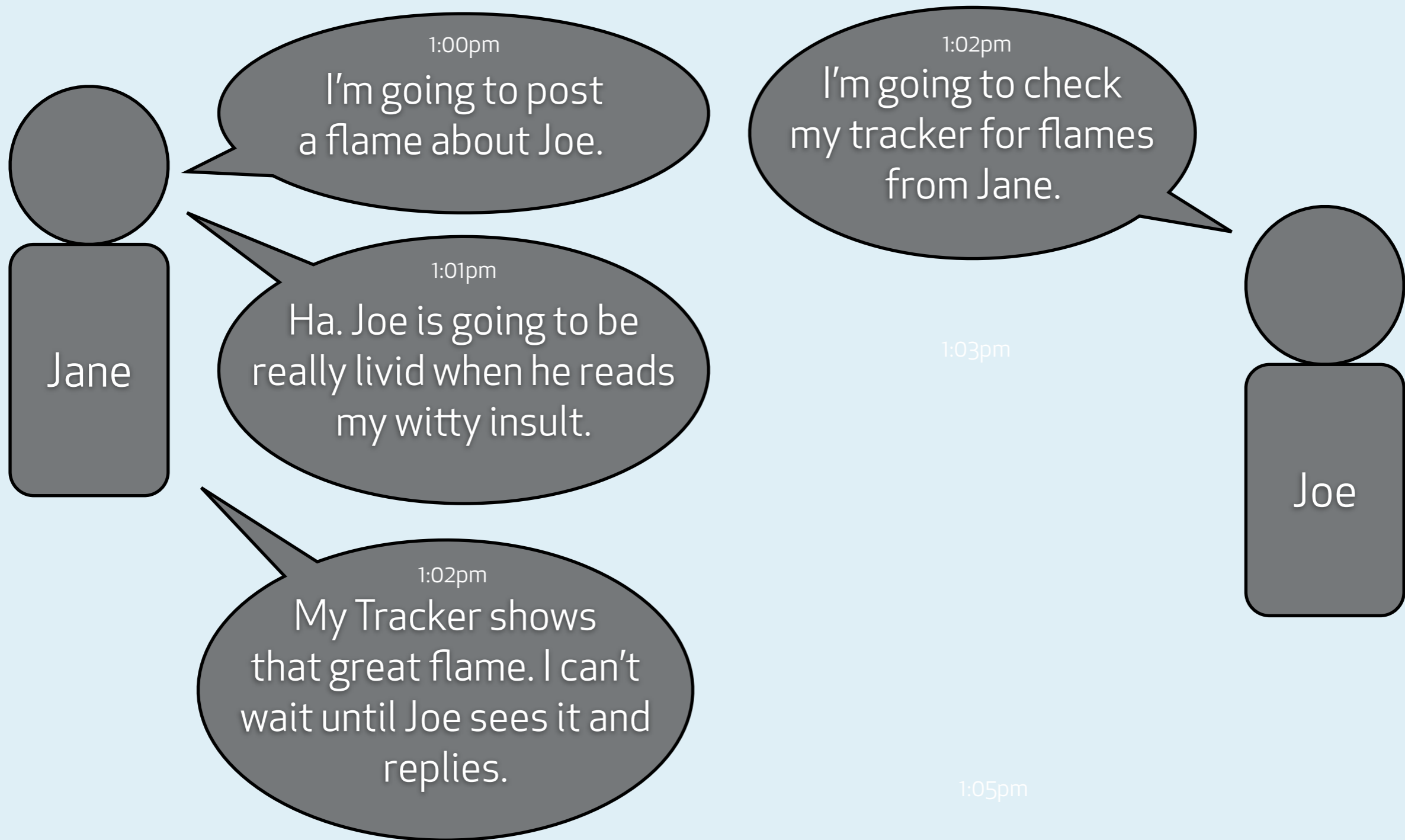
Consistency by perspective



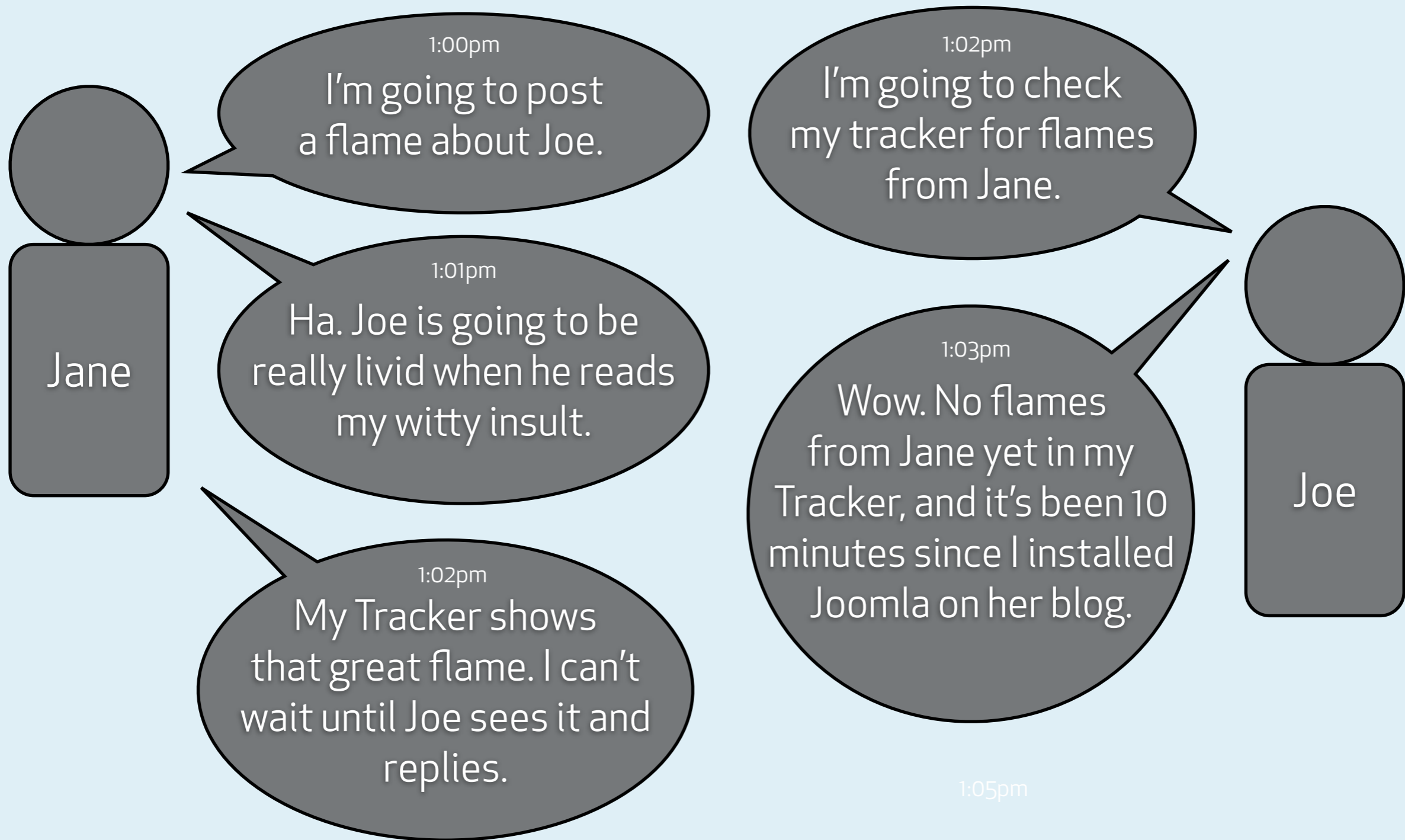
Consistency by perspective



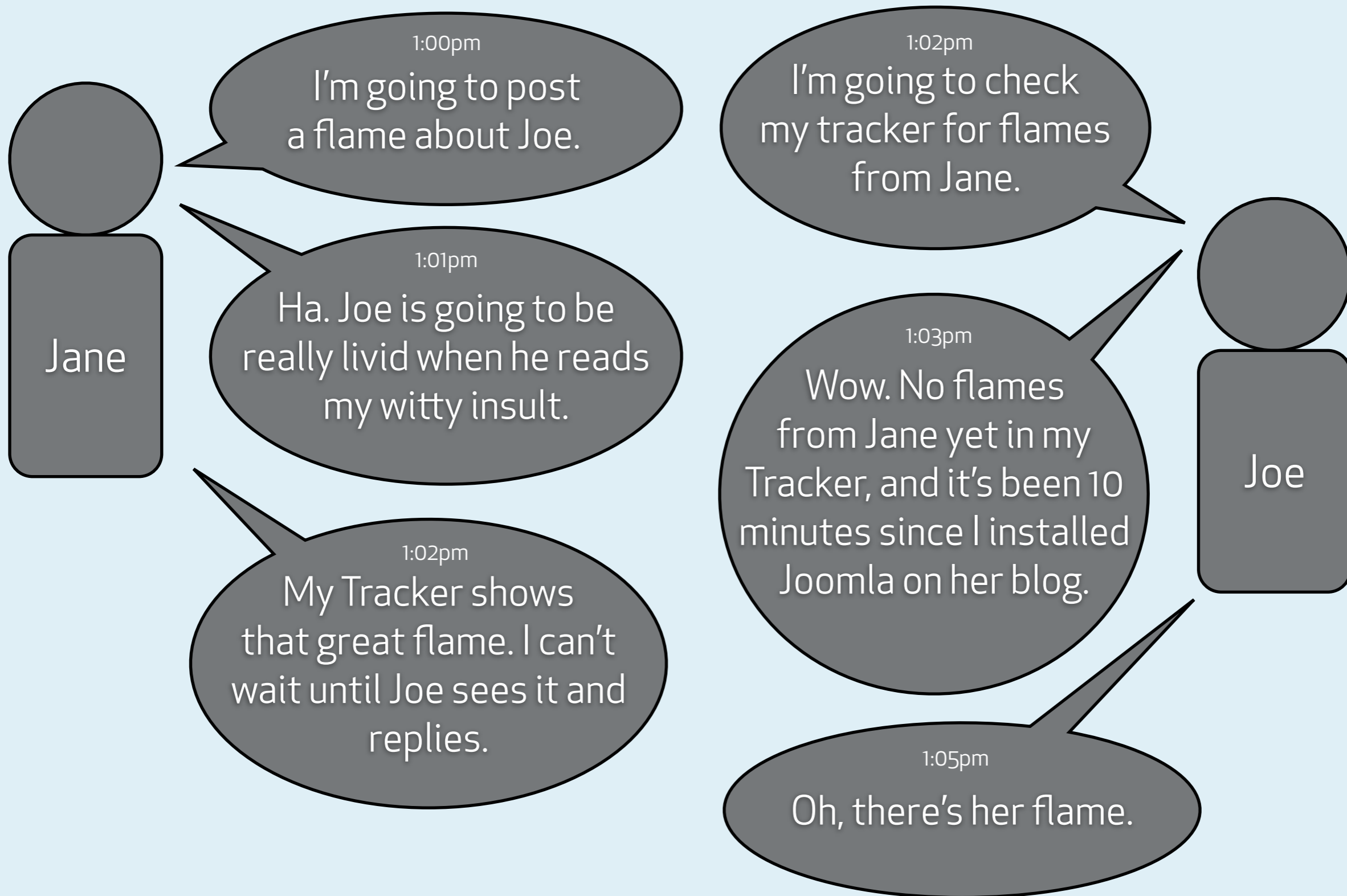
Consistency by perspective



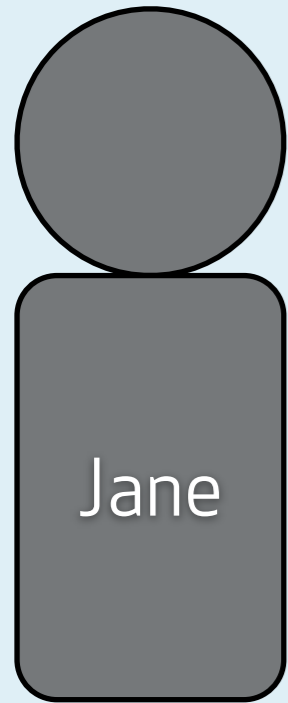
Consistency by perspective



Consistency by perspective



The problems of inconsistency



1:00pm

1:01pm

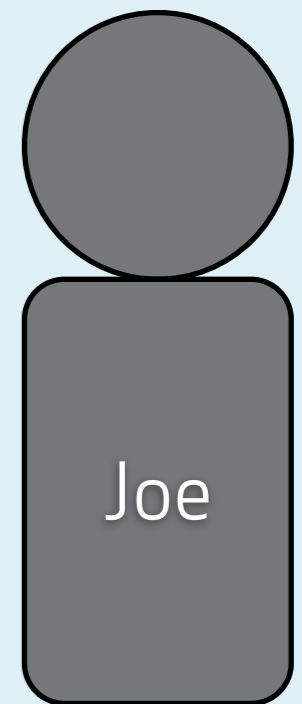
1:02pm

1:03pm

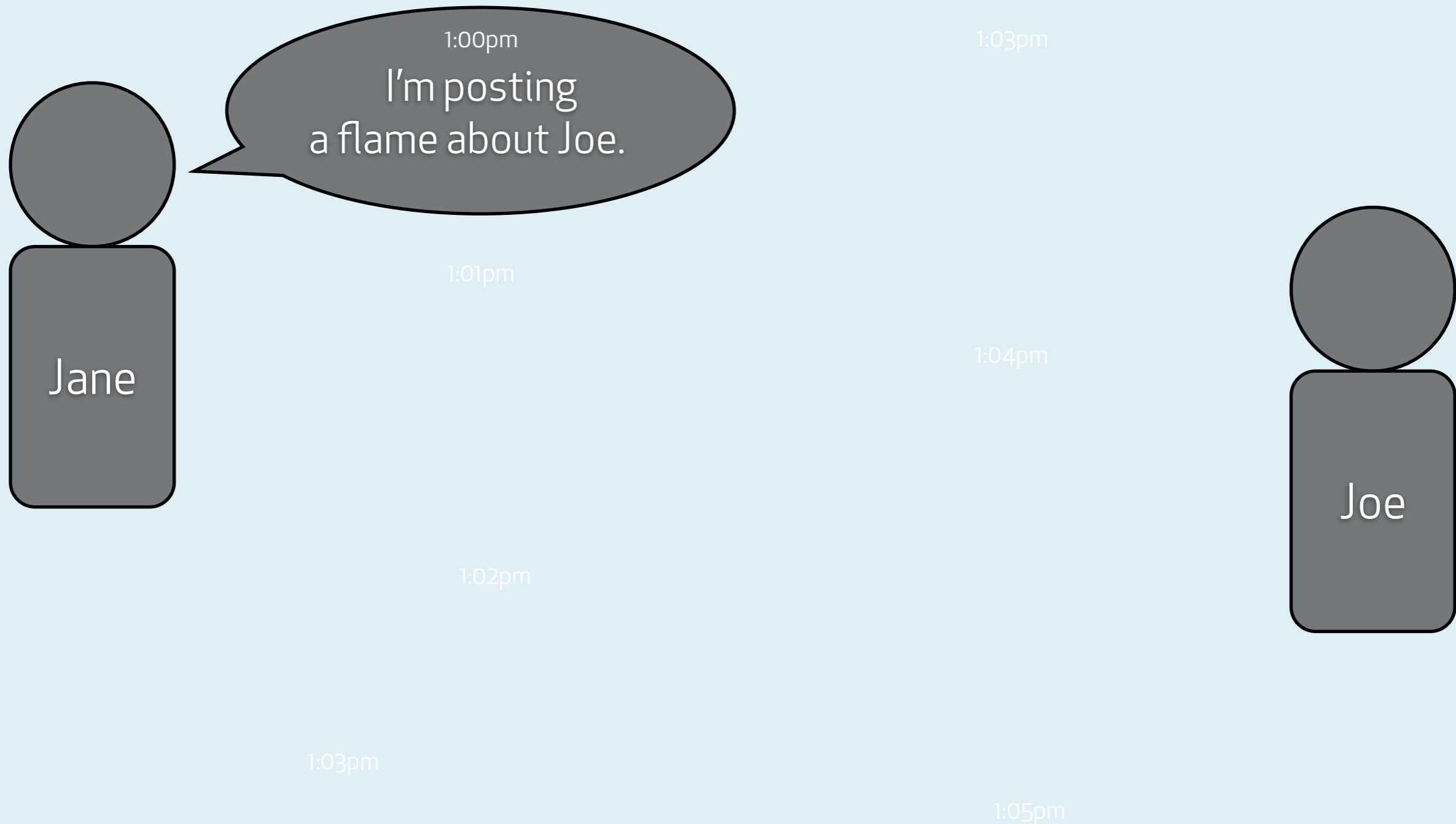
1:03pm

1:04pm

1:05pm



The problems of inconsistency



The problems of inconsistency



The problems of inconsistency



The problems of inconsistency



The problems of inconsistency



The problems of inconsistency



The problems of inconsistency

What happens when Jane saves a node tagged with “flame” to one server, and Joe deletes the tag “flame” from another server? They have to reconcile.

A few options:

Delete the node because it uses a now-invalid tag.

Re-add the tag so the node has all the specified tags.

Keep the node but **without the tag**.

Ask what to do once the conflict is realized.



It's the application's problem

Applications need to understand and intelligently resolve conflicts.

This is not something we can teach the **database** to do.

Many conflicts have **obvious** resolutions.



We already have this problem

We use MySQL replication, which is **asynchronous**.

We already identify queries that need to see changes **immediately**.

We even balance the user's need to see their own changes from others' need to see other users' changes **eventually**.

It's just one more step to supporting inconsistency.



Bayou

A Xerox PARC project that ended in **1997**.

Studied these the idea of designing for **disconnected** operation.

Created a database designed to reconcile data and ask the application to solve the conflicts.

