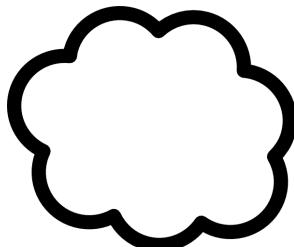
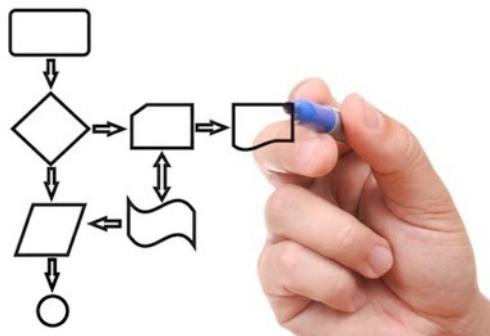


# San Jose State University

CMPE 281  
Cloud Technologies

## Vector Clock Example



# Version Stamps

A field that changes each time data is updated

## Examples:

**Counter** Requires a Single Master to generate unique values.

**GUID** Anyone can generate. But values are large and hard to compare.

**Hash** Hash of content with a big enough hash key size for global uniqueness.

**Timestamp** Nodes have to synchronize clocks.

**Vector Stamps** A vector of version stamps from each node.  
Example: [Node-A: 2, Node-B: 5, Node-C: 10]

# The Actors: Alice, Ben, Cathy, Dave

## Goal: Which day to meet for dinner?

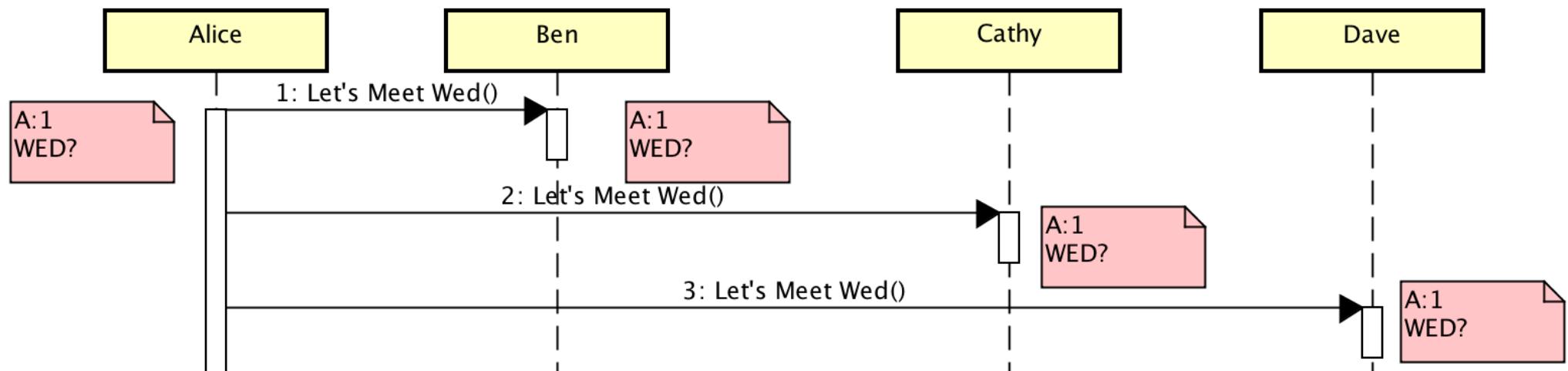
The screenshot shows a portion of the Basho website. At the top, there's a navigation bar with links for ACADEMY, DOWNLOAD RIAK, DOCS, and CONTACT. Below the navigation, there are four orange links: PRODUCTS, INTEGRATIONS, SERVICES, and INDUSTRIES. The main content area has a light gray background and features a dark gray header with the text "Vector Clocks by Example". Below this, the text reads: "We've all had this problem: Alice, Ben, Cathy, and Dave are planning to meet next week for dinner. The planning starts with Alice suggesting they meet on Wednesday. Later, Dave discusses alternatives with Cathy, and they decide on Thursday instead. Dave also exchanges email with Ben, and they decide on Tuesday. When Alice pings everyone again to find out whether they still agree with her Wednesday suggestion, she gets mixed messages: Cathy claims to have settled on Thursday with Dave, and Ben claims to have settled on Tuesday with Dave. Dave can't be reached, and so no one is able to determine the order in which these communications happened, and so none of Alice, Ben, and Cathy know whether Tuesday or Thursday is the correct choice."

*The story changes, but the end result is always the same: you ask two people for the latest version of a piece of information, and they reply with two different answers, and there's no way to tell which one is **really** the most recent.*

*Vector clocks to the rescue, but how? Simple: tag the date choice with a vector clock, and then have each party member update the clock whenever they alter the choice.*

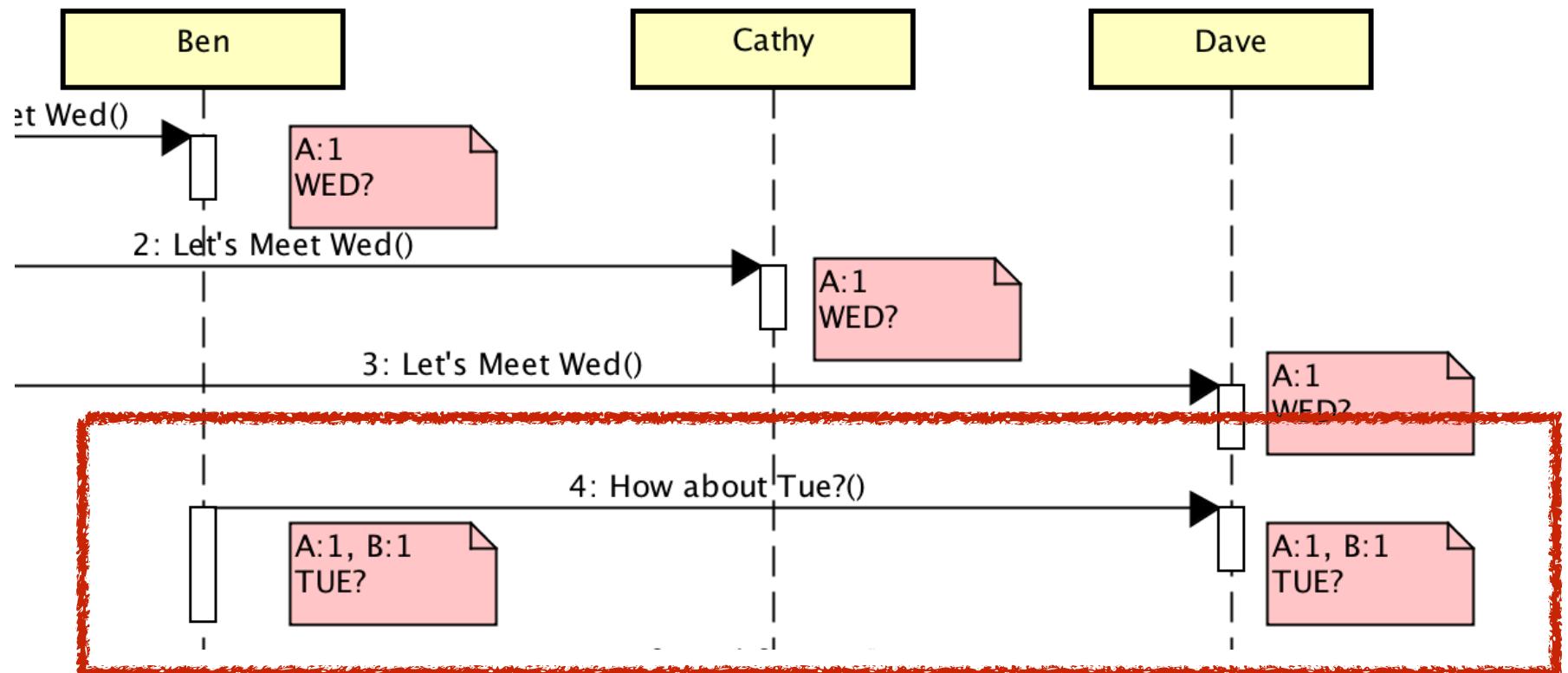
# Message I: Alice asks Ben, Cathy and Dave if “Wednesday” is Okay?

Action: Alice version stamps her message with “A: 1”



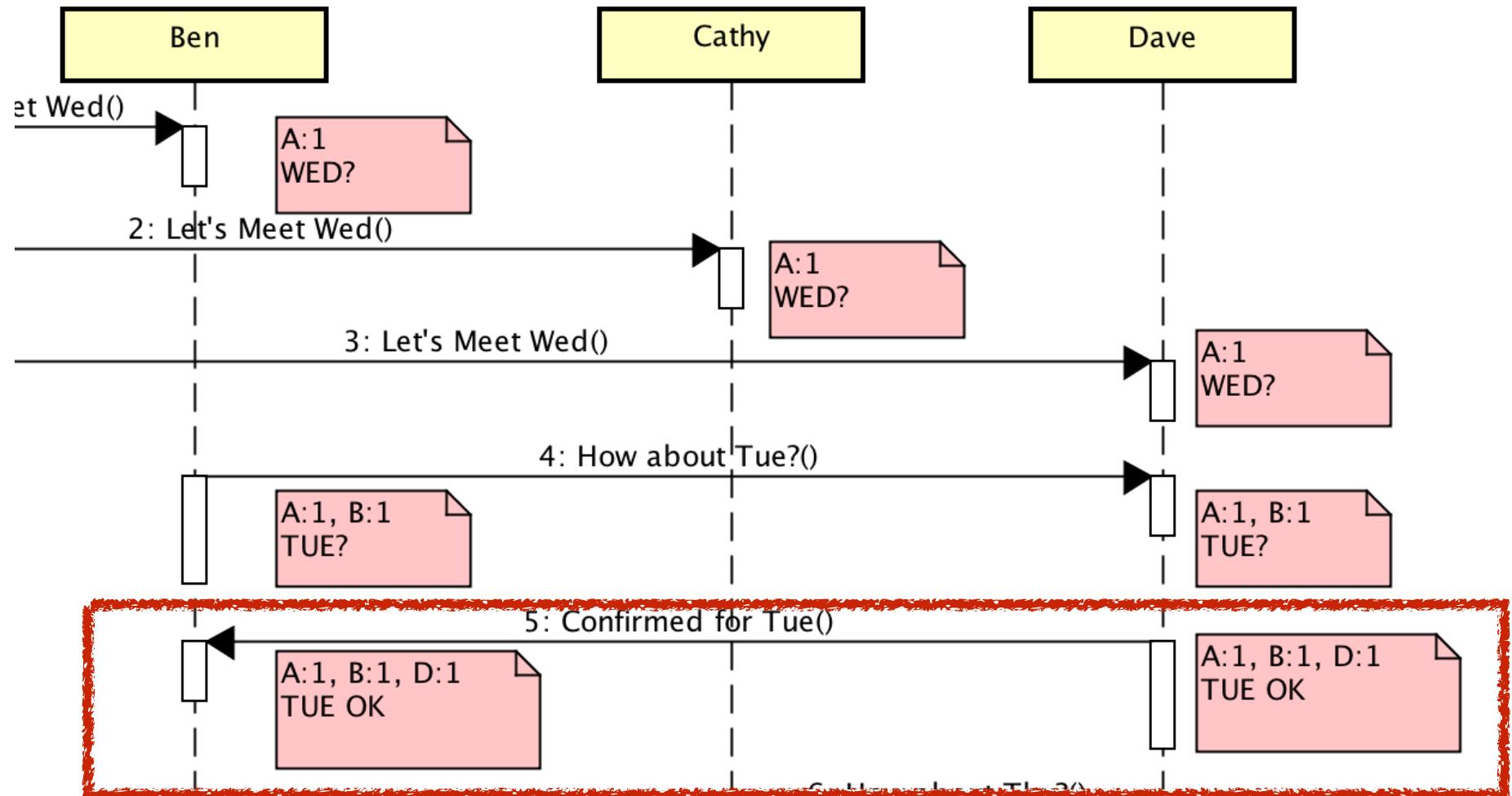
# Message 2: Ben asks Dave “How about Tuesday”?

Action: Ben version stamps his message:  
“A: I, B: I” (Leaving Alice’s Stamp Alone)



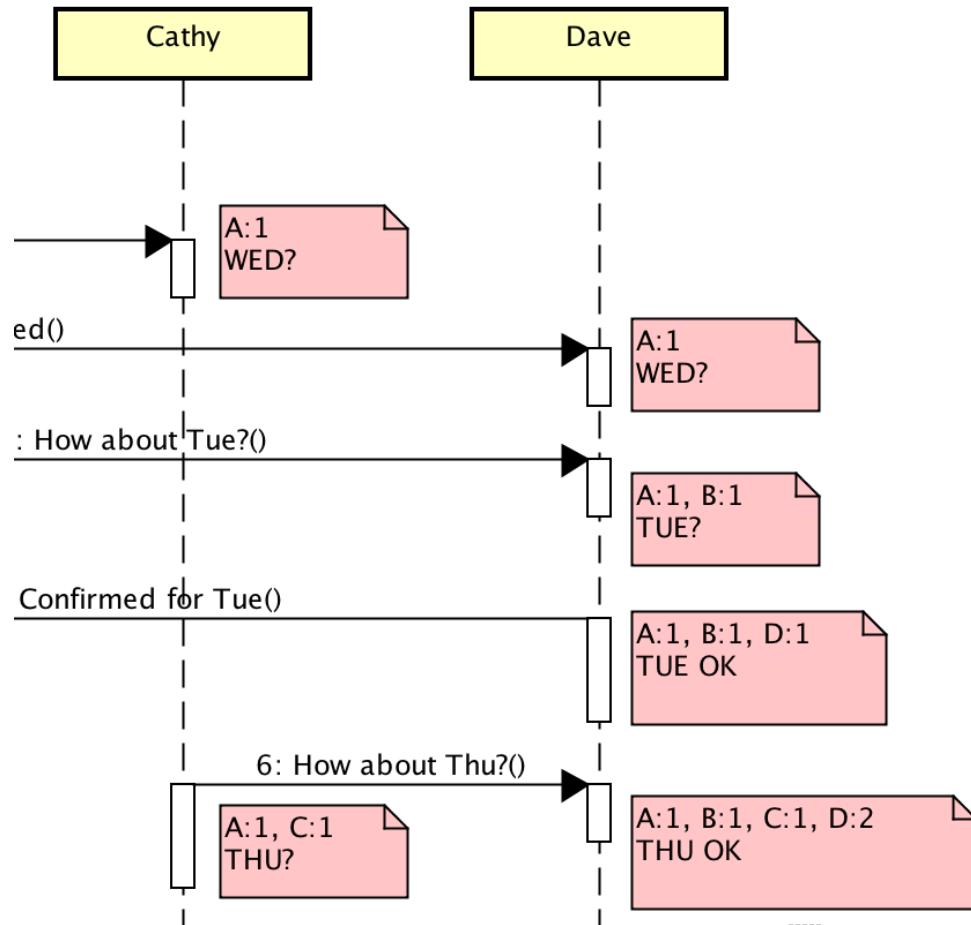
# Message 3: Dave replies to Ben “Confirms Tuesday”

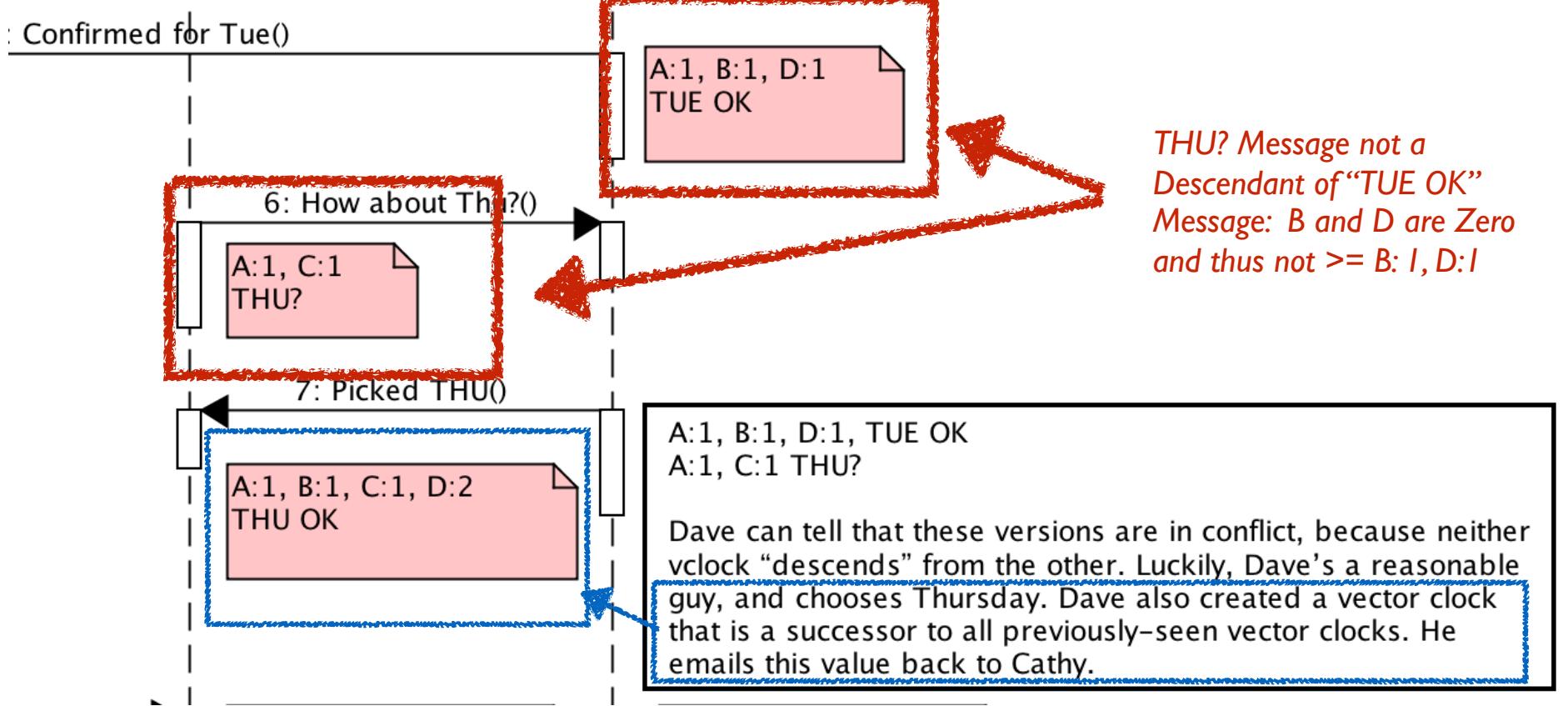
Action: Dave version stamps his message: “A: 1, B: 1, D: 1”



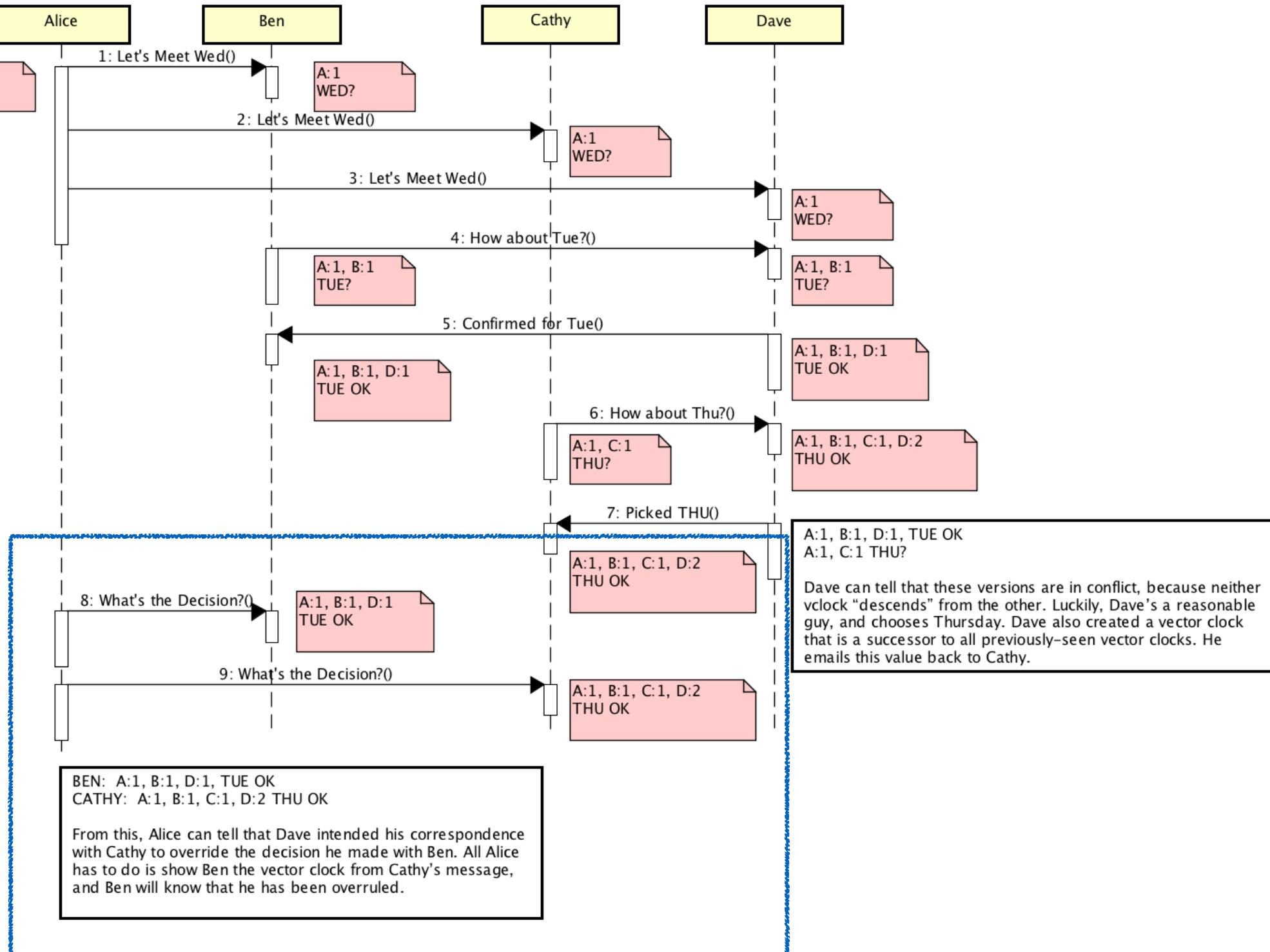
# Message 4: Cathy asks Ben “Thursday?”

Action: Cathy version stamps her message: “A: I, C: I”





Dave can tell that these versions are in conflict, because neither vclock “descends” from the other. **In order for vclock B to be considered a descendant of vclock A, each marker in vclock A must have a corresponding marker in B that has a revision number greater than or equal to the marker in vclock A.** Markers not contained in a vclock can be considered to have revision number zero



# Source of Example

<http://basho.com/posts/technical/why-vector-clocks-are-easy/>

The screenshot shows a web browser window with the title bar "Vector Clocks Explained". The address bar contains the URL "http://basho.com/posts/technical/why-vector-clocks-are-easy/". The page itself is the "Basho BLOG" section, featuring a large header "BASHO BLOG" and a sub-header "Why Vector Clocks are Easy". Below the header, it says "Posted January 29, 2010 | by Bryan Fink | Category: Technical Blog". The main content discusses vector clocks, stating they are confusing at first but simple once understood. It then provides a rule for using them effectively. The bottom of the page starts with "The rest of this post will explain why and how to follow that simple rule. First, I'll explain how vector clocks work with a very". The browser interface includes a toolbar with various icons, a search bar, and language selection for English.

Vector Clocks Explained

basho.com/posts/technical/why-vector-clocks-are-easy/

Paul

SEARCH

PRODUCTS INTEGRATIONS SERVICES INDUSTRIES USE CASES RESOURCES COMPANY BLOG

# BASHO BLOG

## Why Vector Clocks are Easy

Posted January 29, 2010 | by Bryan Fink | Category: Technical Blog

*January 29, 2010*

Vector clocks are confusing the first time you're introduced to them. It's not clear what their benefits are, nor how it is you derive said benefits. Indeed, each Riak developer has had his own set of false starts in making them behave.

The truth, though, is that vector clocks are actually very simple, and a couple of quick rules will get you all the power you need to use them effectively.

The simple rule is: assign each of your actors an ID, then make sure you include that ID and the last vector clock you saw for a given value whenever to store a modification.

The rest of this post will explain why and how to follow that simple rule. First, I'll explain how vector clocks work with a very

The screenshot shows a web browser window with the title bar "Why Vector Clocks Are Hard - basho.com". The address bar contains the URL "http://basho.com/posts/technical/why-vector-clocks-are-hard/". The page header includes the Basho logo, navigation links for "ACADEMY", "DOWNLOAD RIAK", "DOCS", "CONTACT", "ENGLISH", "SEARCH", and categories like "PRODUCTS", "INTEGRATIONS", "SERVICES", "INDUSTRIES", "USE CASES", "RESOURCES", "COMPANY", and "BLOG". A decorative Greek key pattern runs horizontally across the page. The main content area features a large heading "BASHO BLOG" and a sub-heading "Why Vector Clocks Are Hard". Below the heading, it says "Posted April 5, 2010 | by Justin Sheehy | Category: Technical Blog". A horizontal line with diamond icons separates this from the text "April 5, 2010". The main body text discusses vector clocks and their implementation, mentioning Bryan's example and the challenges of incrementing and resolution. At the bottom, there is a note about parties proposing changes ("clients") and actors in vector clocks.

# BASHO BLOG

## Why Vector Clocks Are Hard

Posted April 5, 2010 | by Justin Sheehy | Category: Technical Blog

---

*April 5, 2010*

A couple of months ago, [Bryan wrote about vector clocks](#) on this blog. The title of the post was “Why Vector Clocks are Easy”; anyone who read the post would realize that he meant that they’re easy for a client to use when talking to a system that implements them. For that reason, there is no reason to fear or avoid using a service that exposes the existence of vector clocks in its API.

Of course, actually implementing such a system is not easy. Two of the hardest things are deciding what an actor is (i.e. where the incrementing and resolution is, and what parties get their own field in the vector) and how to keep vclocks from growing without bound over time.

In Bryan’s example the parties that actually proposed changes (“clients”) were the actors in the vector clocks. This is the model that