```
class User(id: UserID, name: String,
  followers: Set[UserID] with LatencyBound(20 ms),
  timeline: List[TweetID] with LatencyBound(20 ms))

class Tweet(id: TweetID, user: UserID, text: String,
  retweets: Set[UserID] with Size(ErrorTolerance(5%)))

def display_timeline(user: User) = {
  // ignore inconsistent timeline, user won't notice
  val tweets = endorse(user.timeline.range(0, 10))
  for (tweet <- tweets) {
    val rct: Interval[Int] = tweet.retweets.size()
    if (rct.max - rct.min > 100)
      // abbreviate large retweet counts (e.g. "10k")
      display("${rct.min/1000}k retweets")
    else if (rct.max - rct.min > 0)
      display("~${rct.mid} retweets")
    else
      display("${rct.mid} retweets")
    // contains returns Consistent[T] so is coercible
    if (!tweet.retweets.contains(user))
       display_retweet_button()
  }
}
```