```
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 viewTimeline(user: User) = {
  // `range` returns `Rushed[List[TweetID]]`
  user.timeline.range(0,10) match { // use match to unpack
    case Consistent(tweets) =>
      for (tweetID <- tweets)
        display(loadTweet(tweetID))
    case Inconsistent(tweets) =>
      // tweets may not have fully propagated yet
      for (tweetID <- tweets)
        // guard load and retry if there's an error
        val tweet = Try { loadTweet(tweetID) } retryOnError
        display(tweet)
  }
}
```