Fakultät für Informatik
Technische Universität München

Enabling realtime collaborative data-intensive web applications

A case study using serverside JavaScript

Betreuer: Sascha Roth

Kooperationspartner: Pentasys AG

PENTASYS
Unser Maßstab ist der Mensch

Outline

1. Node.js – Introduction & Survey
2. Prototypical implementation of a real-time collaboration tool
3. Conclusion & Outlook

Node.js - Introduction

- Developed by Ryan Dahl in 2009
- JavaScript Interpreter
- JavaScript outside of the browser
- Extends Googles V8 with low-level bindings
  - Filesystem, Sockets,...
- Every binding is asynchronous, event loop
- Single threaded
- Current version: 0.10

Node.js - Survey

- Hypotheses
  - JavaScript developers are unsatisfied with the current tool support
  - Developers do not like the syntax of JavaScript
  - JavaScript code is hard to maintain
  - Node.js is suitable for enterprise applications
- Participants
  - 100 complete answers
  - Countries:

Germany:	37%
USA:	22%
UK:	3%

  - Mostly Web-Developers

Node.js - Survey

- Key results about JavaScript

  - 53.7% are satisfied with their editors
    - Most popular: IntelliJ, WebStorm, Emacs
    - Missing features: code completion, code navigation, debugging

  - 65% like the syntax of JavaScript
    - CoffeeScript and TypeScript are not planed to be used in future

  - 44% do not think, JS is hard to maintain
  - 65% said JS code is easy to read
  - 42% use testing frameworks
  - The usage of testing frameworks influences maintainability

Node.js - Survey

- Key results about Node.js
  - 88% have heard about Node.js
  - Node.js projects tend to be smaller
    - 41.5% of projects: 1 poeple
    - 22.6% of projects: 3 poeple
  - Typical kinds of projects
    - 61.9% of projects: Web Application

Node.js - Survey

- Key results about Node.js

  - Reasons for using Node.js
    - Simplicity
    - Performance
    - „Good fit for Web Applications“
    - No „phase shift“
    - Realtime capabilities
    - Event-driven

  - 64.8% confirmed the enterprise readiness
    - Scalability
    - Stability
    - Short time to market
    - Same language at client & server

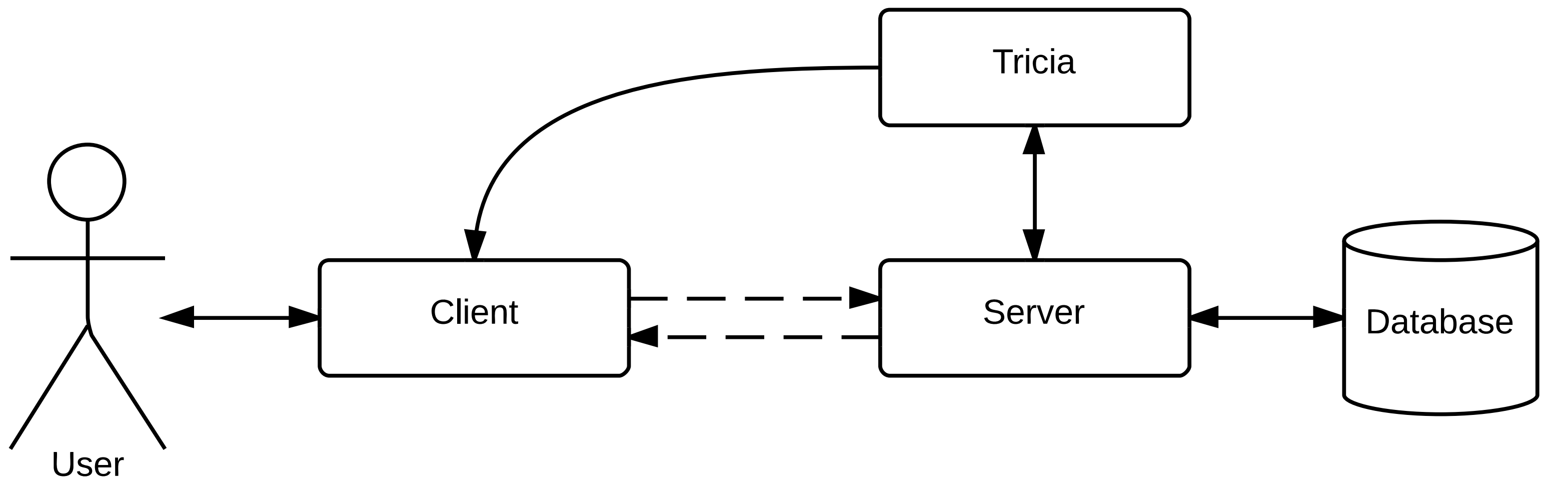
Prototypical implementation of a Real-Time collaboration tool

Demo

The Real-Time Architecture

- Proposed by Alex MacCaw (JavaScript Web Applications)
  - Real-Time architecture = event-driven
  - Driven by user interactions
- Client-side MVC
- Which model updates need to be distributed ?
- Who needs to be notified ?
- PubSub pattern

Prototypical Implementation – Fundamental Architecture



Prototypical Implementation – Technology Stack

- Client UI with Twitter Bootstrap
- Client MVC with EmberJS:
  - Model:

```
CollaborativeEditor.Documents = Ember.A([]);

CollaborativeEditor.Document = Ember.Object.extend({
            name : null,
            lastModified : null,
            numberOfPeopleEditing : null,
            tags : Ember.A([])
});
```

Prototypical Implementation – Technology Stack

- Client MVC with EmberJS:
  - Controller:

```
CollaborativeEditor.DocumentsController = Ember.ArrayController.extend({
            createNewDoc : function() {
                        ...
            },
            ...
```

Prototypical Implementation – Technology Stack

- Client MVC with EmberJS:
  - View:

```
<script type="text/x-handlebars" data-template-name="documents">
      <div class="hero-unit">
        <h2>Documents</h2>
      </div>
      <a href="#" class="btn" {{action "createNewDoc"}}>
        <i class="icon-plus"></i>
      </a>
      <table class="table table-striped">
      ...
        {{#each controller}}
          <tr>
            <td>{{name}}</td>
            <td>{{lastModified}}</td>
```

Prototypical Implementation – Technology Stack

- Serverside with Node.js
- Socket.io library for realtime communiction
    - Abstracts underlying technology (Long-polling, Web-sockets,...)
    - Server:

```
socket.on('storeAndDistributeDocument', function(doc, callback) {
		documentManager.storeDocument(doc);
		socket.broadcast.emit('newDoc', doc);
		callback();
});
```

- Client:

```
socket.emit('storeAndDistributeDocument', doc, function() {
		ui.addDocument(doc);
});
```

Prototypical Implementation – Technology Stack

- Share.js library for concurrent editing
  - Concurrent Editing of plain text
  - Operational Transformation based
- Database
  - Redis

Conclusion & Outlook

Conclusion & Outlook

- Node.js is enterprise ready

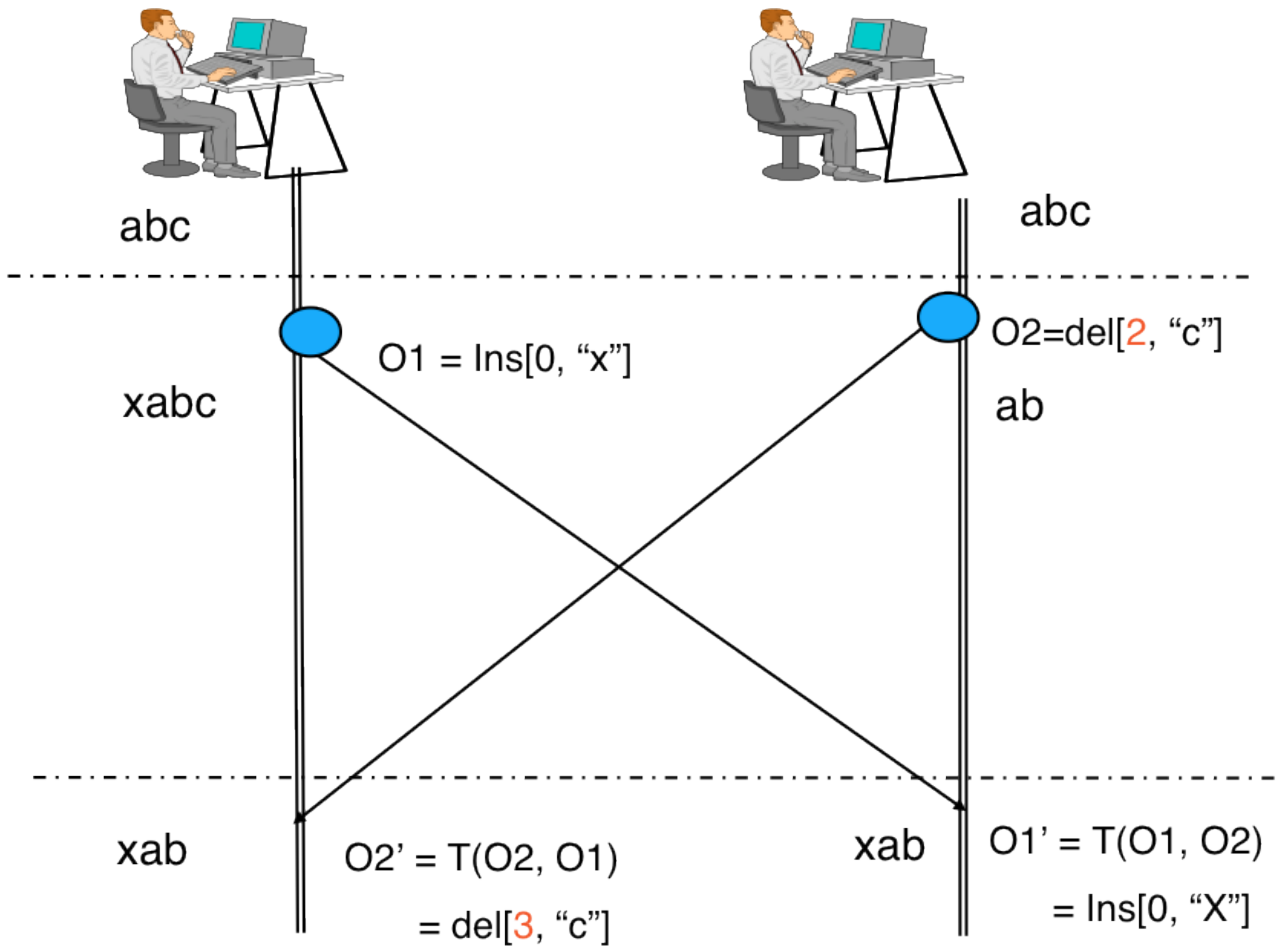
- Node.js is a good fit for real-time web applications
  - Event-driven itself
  - Non-blocking IO
  - Offers lot of real-time functionalities

- Pattern for Tricia real-time functionalities

- Further Research:
  - Collaborative editing of rich text
  - Collaborative editing of general models

Thank you for your attention

OT


abc
abc
xabc
O1 = Ins[0, "x"]
O2=del[2, "c"]
ab
xab
O2' = T(O2, O1)
= del[3, "c"]
xab
O1' = T(O1, O2)
= Ins[0, "X"]