PeerGaming

Share the Fun

Creating a Client-Side Multiplayer Gaming Framework for the Web,
which handles Distributed Logic on Multiple Systems in "Real-Time"

Overview

Idea

Design

Code

Background

Browser Based Games

Console + Mobile Market

Browser Based Games

Console + Mobile Market

Plugins / Extensions

Browser Based Games

Console + Mobile Market

Plugins / Extensions

Web Technologies

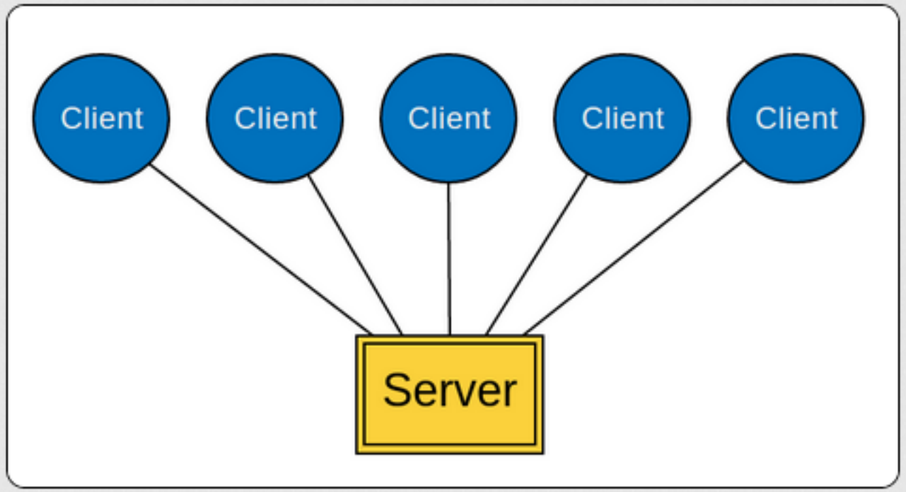
Multiplayer

WebSockets or XHR-Polling

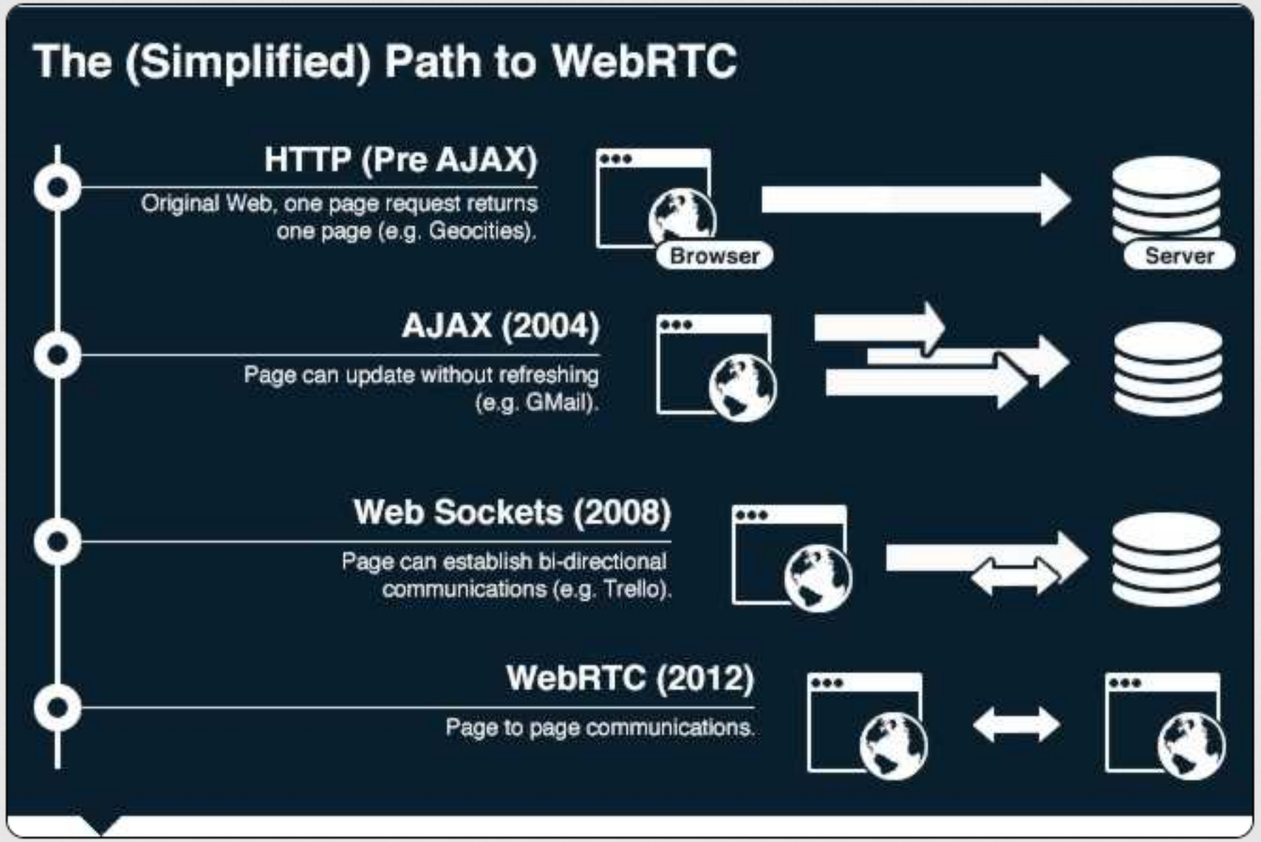
Multiplayer

WebSockets or XHR-Polling

Client-Server Architecture



Evolution



Graphic by Jimmy Lee / jimmylee.info

WebRTC

WebRTC

↙

PeerConnection

Technology

WebRTC

↙ ↓

PeerConnection MediaStream

Technology

WebRTC

↙ ↓ ↘

PeerConnection MediaStream **DataChannel**

Concept

Features

standalone

Features

standalone

configurable

Features

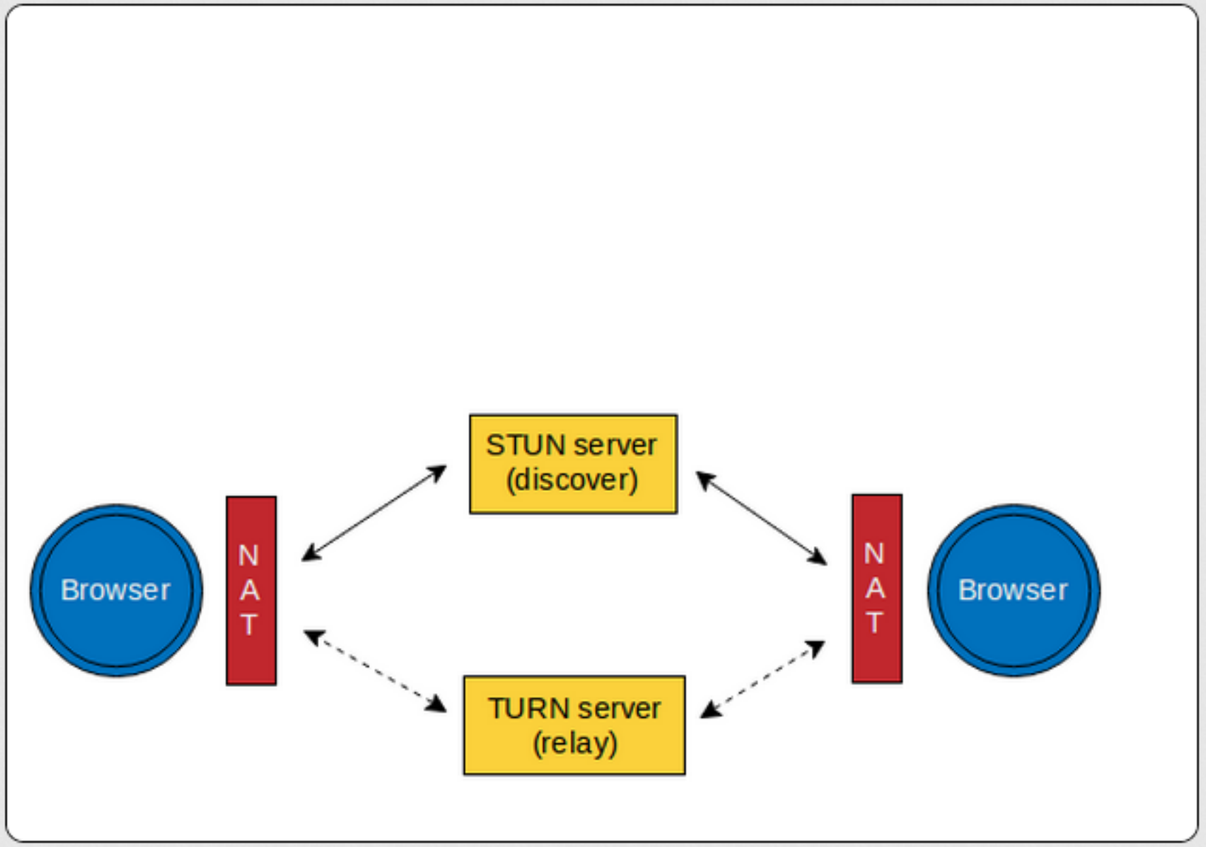
standalone

configurable

simple

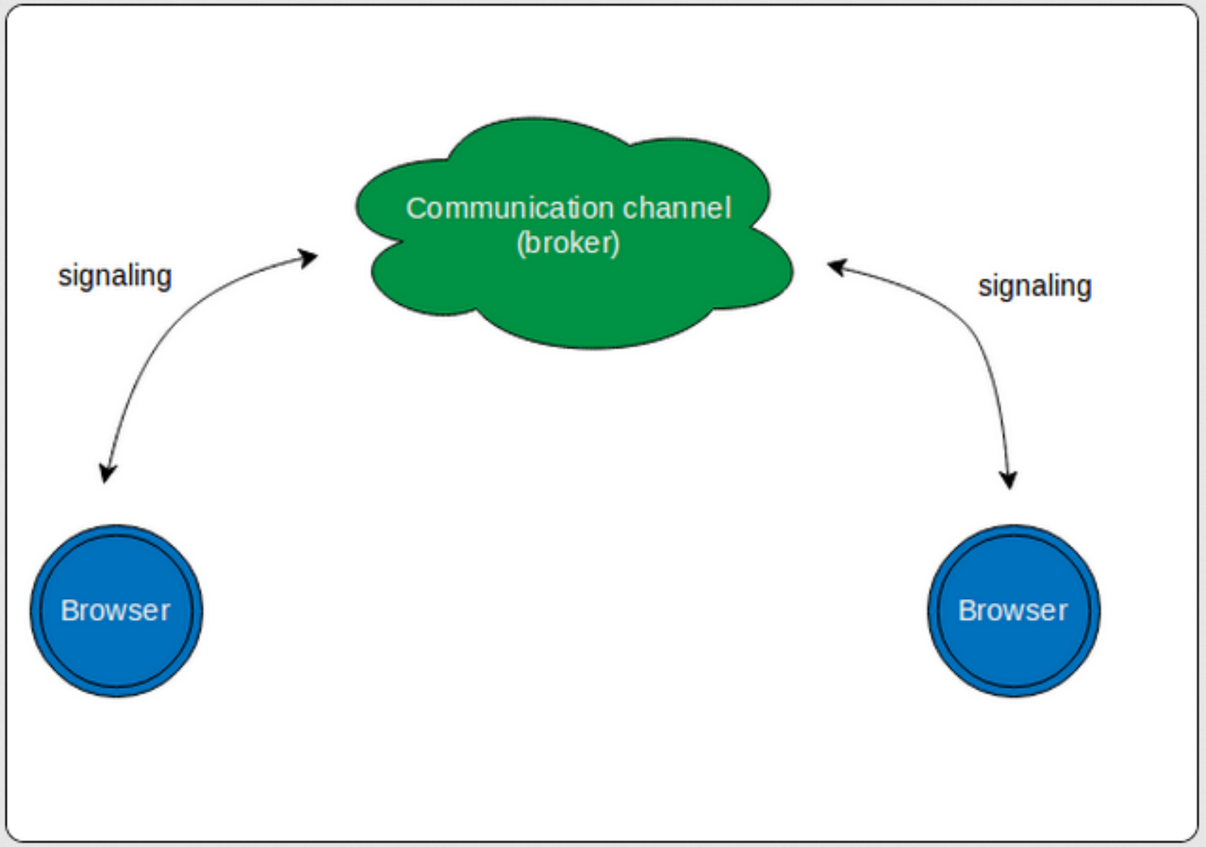
Communication

ICE / JSEP



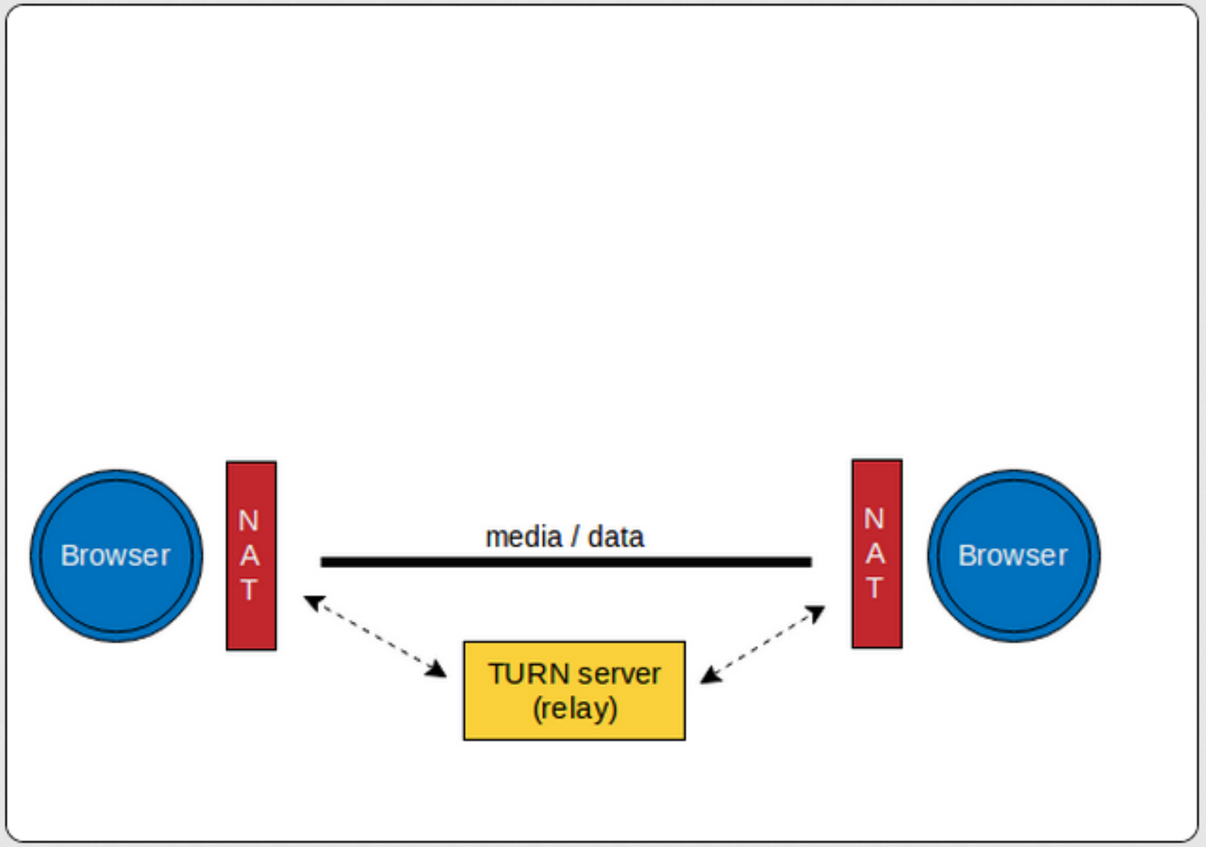
Communication

ICE / JSEP



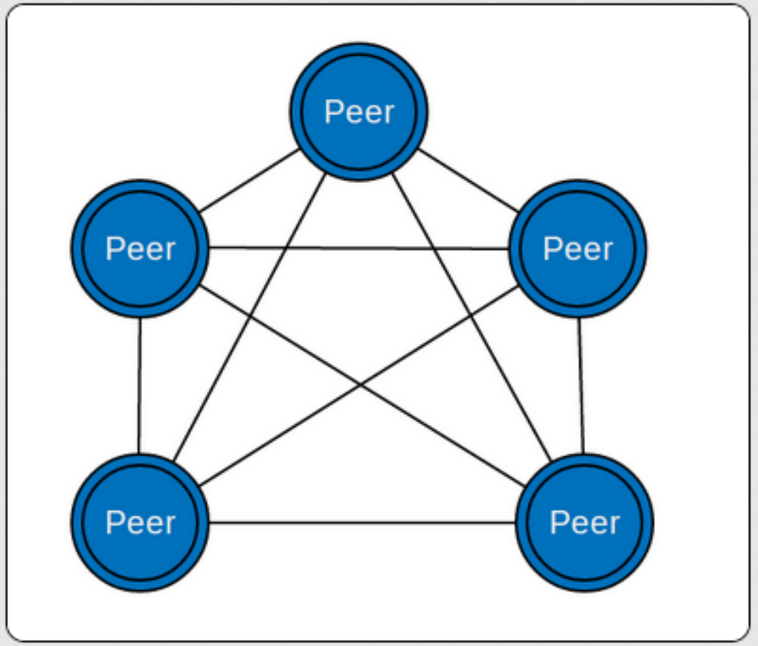
Communication

ICE / JSEP



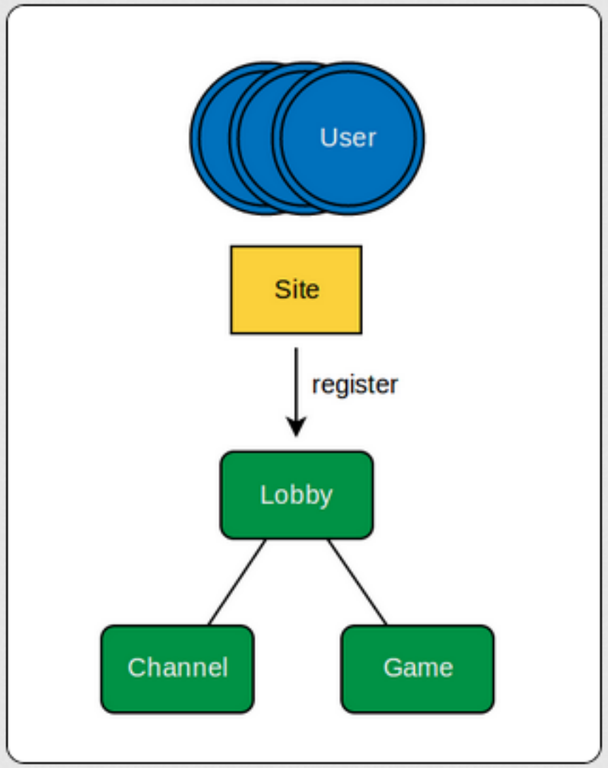
Topology

Full Mesh → 1 : N



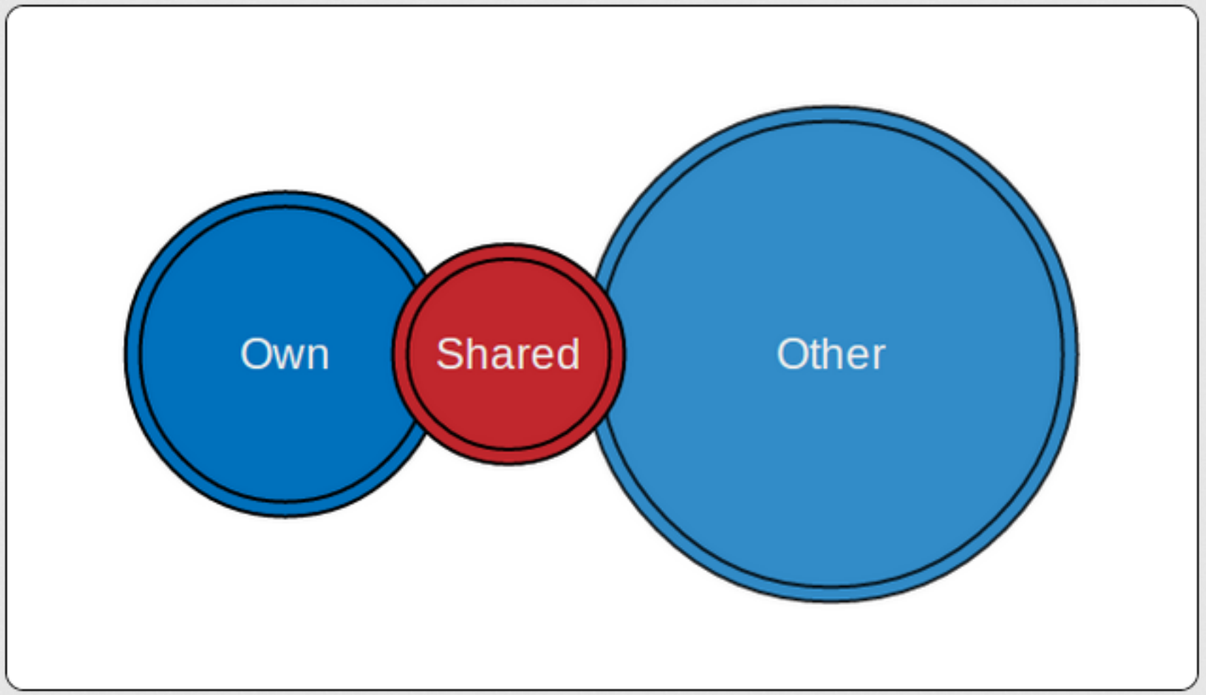
Subnetting

User Flow



Data

Types



Implementation

Components

```
/** API Overview **/

var pg = {

  noConflict    : fn  - 'reset namespace'
  VERSION       : obj - 'refers to the current version'
  info          : obj - 'information about the state'


  // config     : fn  - 'configuration for the network'
  // watch      : fn  - 'notifications by internal events'
  // login      : fn  - 'set identifier and create player'
  // routes     : fn  - 'define default and custom routes'
  // channel    : fn  - 'handler for a "Channel"'
  // game       : fn  - 'handler for a "Game"'
  // player     : obj - 'own user instance (writeable)'
  // peers      : obj - 'list of connected players (readable)'
  // data       : arr - 'shortcut to access peers.data + player.data'
  // sync       : obj - 'synchronized shared object'
  // loop       : fn  - 'synchronized rendering process'
};
```

Components

```
/** API Overview **/

var pg = {

  // noConflict : fn  - 'reset namespace'
  // VERSION    : obj - 'refers to the current version'
  // info       : obj - 'information about the state'

  config        : fn  - 'configuration for the network'
  watch         : fn  - 'notifications by internal events'
  login         : fn  - 'set identifier and create player'
  routes        : fn  - 'define default and custom routes'
  channel       : fn  - 'handler for a "Channel"'
  game          : fn  - 'handler for a "Game"'

  // player     : obj - 'own user instance (writeable)'
  // peers      : obj - 'list of connected players (readable)'
  // data       : arr - 'shortcut to access peers.data + player.data'
  // sync       : obj - 'synchronized shared object'
  // loop       : fn  - 'synchronized rendering process'
};
```

Components

```
/** API Overview **/

var pg = {

  // noConflict : fn  - 'reset namespace'
  // VERSION    : obj - 'refers to the current version'
  // info       : obj - 'information about the state'
  // config     : fn  - 'configuration for the network'
  // watch      : fn  - 'notifications by internal events'
  // login      : fn  - 'set identifier and create player'
  // routes     : fn  - 'define default and custom routes'
  // channel    : fn  - 'handler for a "Channel"'
  // game       : fn  - 'handler for a "Game"'


  player        : obj - 'own user instance (writeable)'
  peers         : obj - 'list of connected players (readable)'
  data          : arr - 'shortcut to access peers.data + player.data'
  sync          : obj - 'synchronized shared object'
  loop          : fn  - 'synchronized rendering process'
};
```

Account

```
/** define a name through user input **/

// with plain text
pg.login( 'Autarc' );

// later with a 3rd party service via OAuth
pg.login( 'Autarc', 'Github' );
```

Meta

Rooms = Channel / Game

```
/**
 *  default scheme:
 *
 *    channel - 1 parameter || http://{HOST}.{TLD}/#!/{channel}/
 *    game    - 2 parameter || http://{HOST}.{TLD}/#!/{channel}/{id}/
 */

// example for a game
http://peergaming.net/pg-catch/42

// handle game events
pg.game( 'pg-catch', function ( game ) {

  game.on('enter', function ( peer ) { ... });

});
```

PeerList (Server)

```
/** select a random peer for the initial connection **/

// addr: 'pg-catch/42',
// id  : 'a1097e1f-4d28-4695-b7ac-0e399690040e'

function getPartner ( addr, id ) {

  var keys = Object.keys( rooms[ addr ] ),
      partnerID = null;

  if ( keys.length > 1 ) partnerID = keys[(Math.random() * keys.length)|0];
  return ( partnerID !== id ) ? partnerID : this.getPartner( addr, id );
}
```

PeerRouting (P2P)

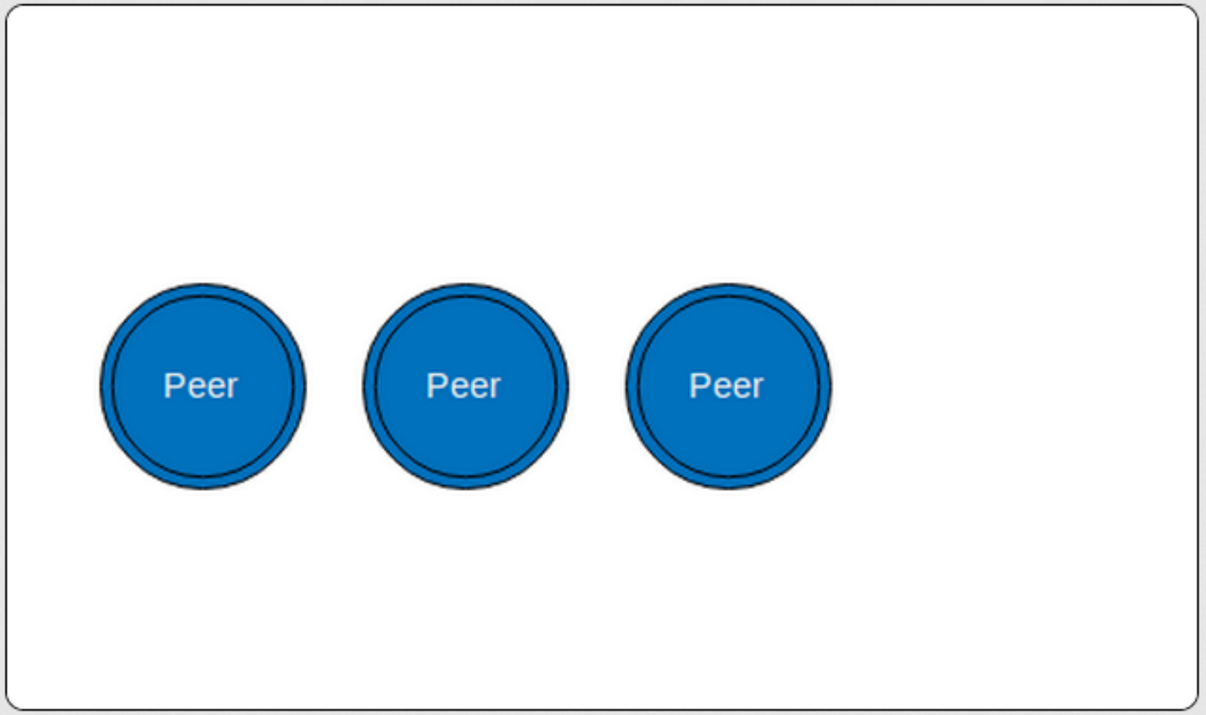
```
/** use brokering for new connections **/

function send ( action, data ) {

  var remote = this.info.remote;

  // use an already known peer
  if ( this.info.transport ) {
    var proxy = { action: action, local: instance.id, remote: remote };
    return this.info.transport.send( 'register', data, proxy );
  }

  // send via server
  socket.send({ action: action, data: data, remote: remote });
}
```

Contact

** manually **

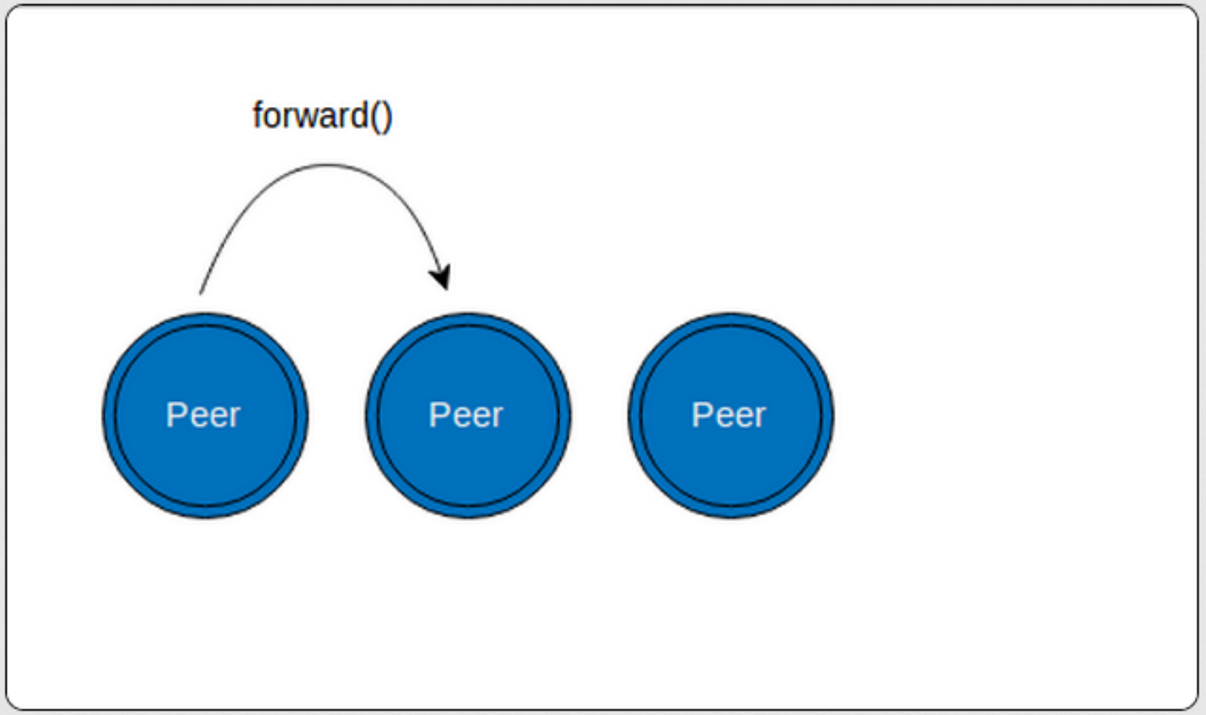
```
/** alternative hook for custom handling the credential exchange **/

pg.config.noServer( function ( msg, conn ) {

  // show own information ( iceCandidates, SDP packages )
  document.body.innerHTML += msg.type + ':' + msg.data;

  // reference to add a partners configurations
  function setOwnCredentials( msg ) {
    conn.set( msg.type, msg.data );
  }

});
```

Setup



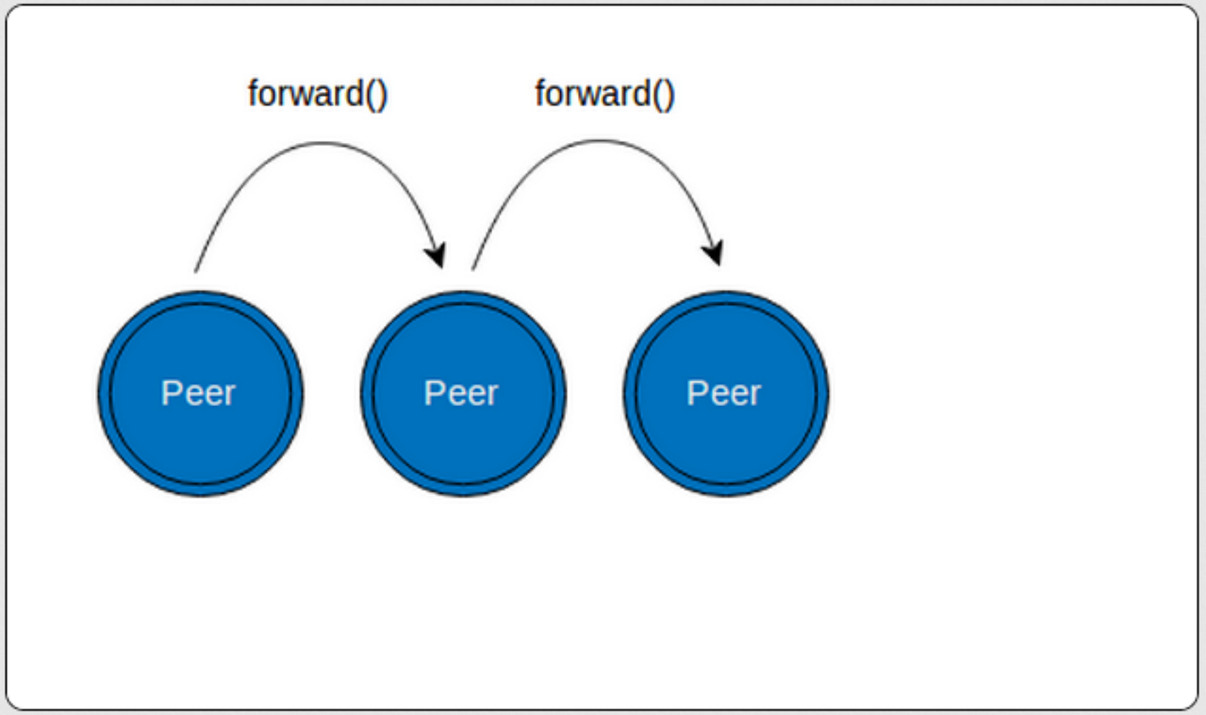
Setup

PeerChain



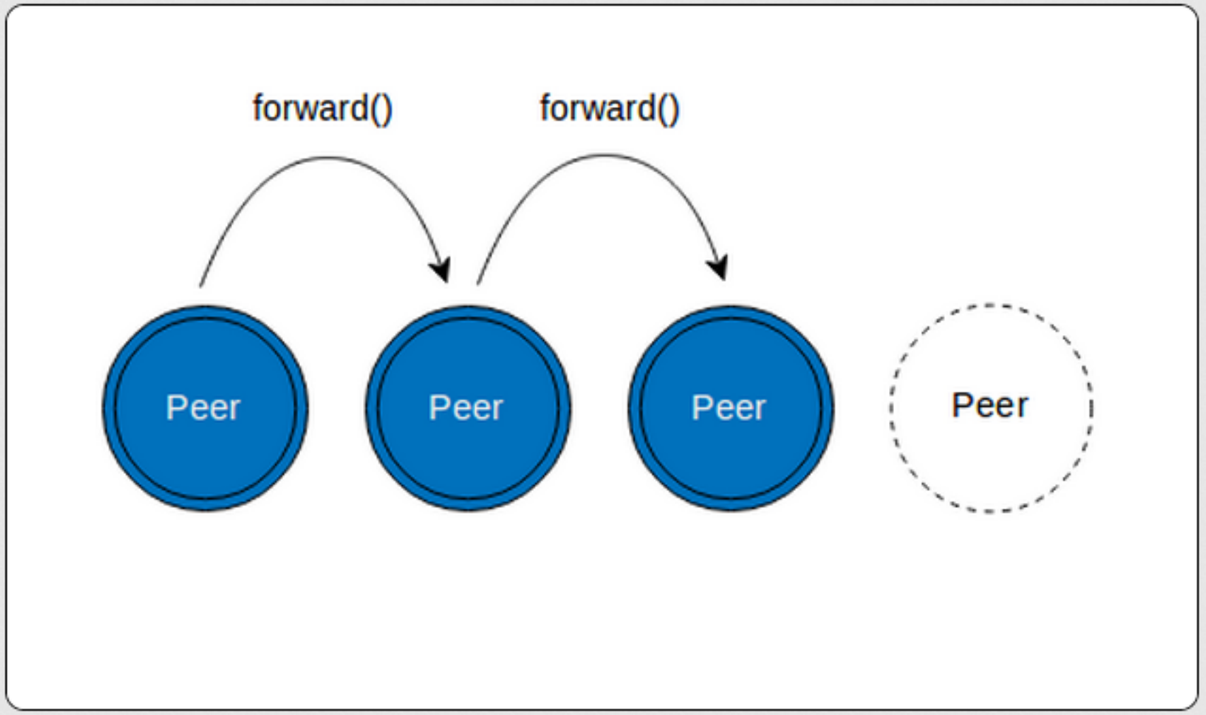
Setup

PeerChain



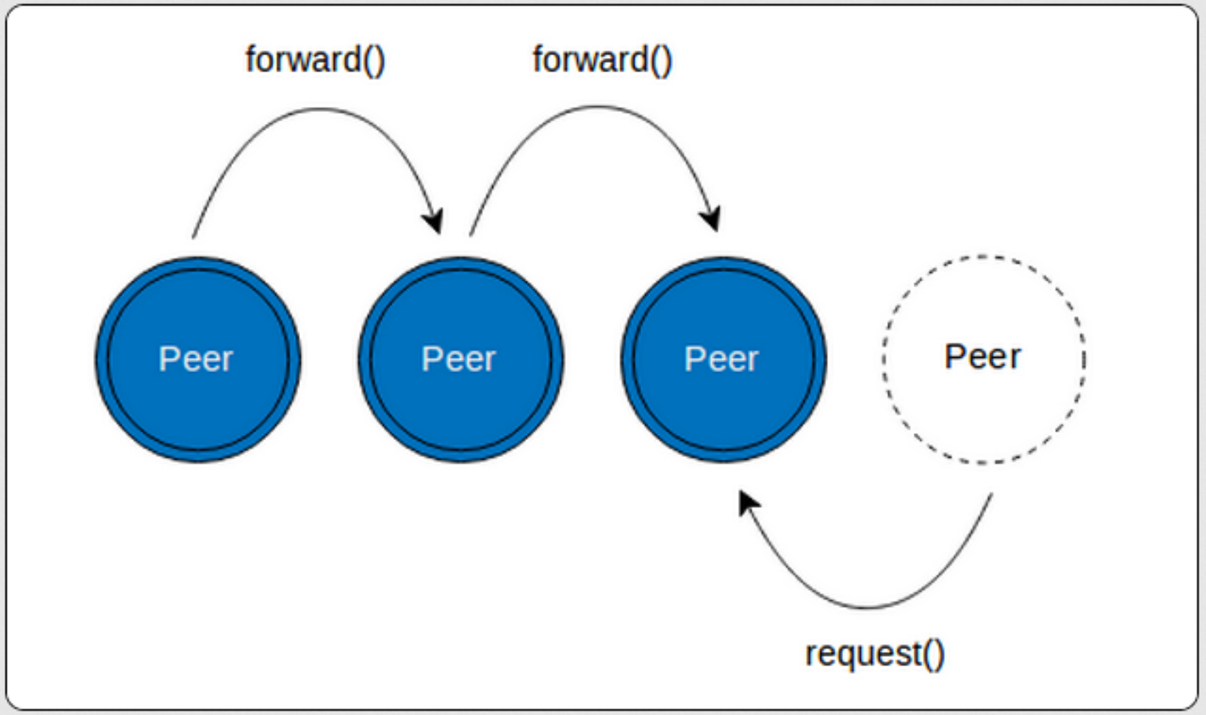
Setup

PeerChain



Setup

PeerChain

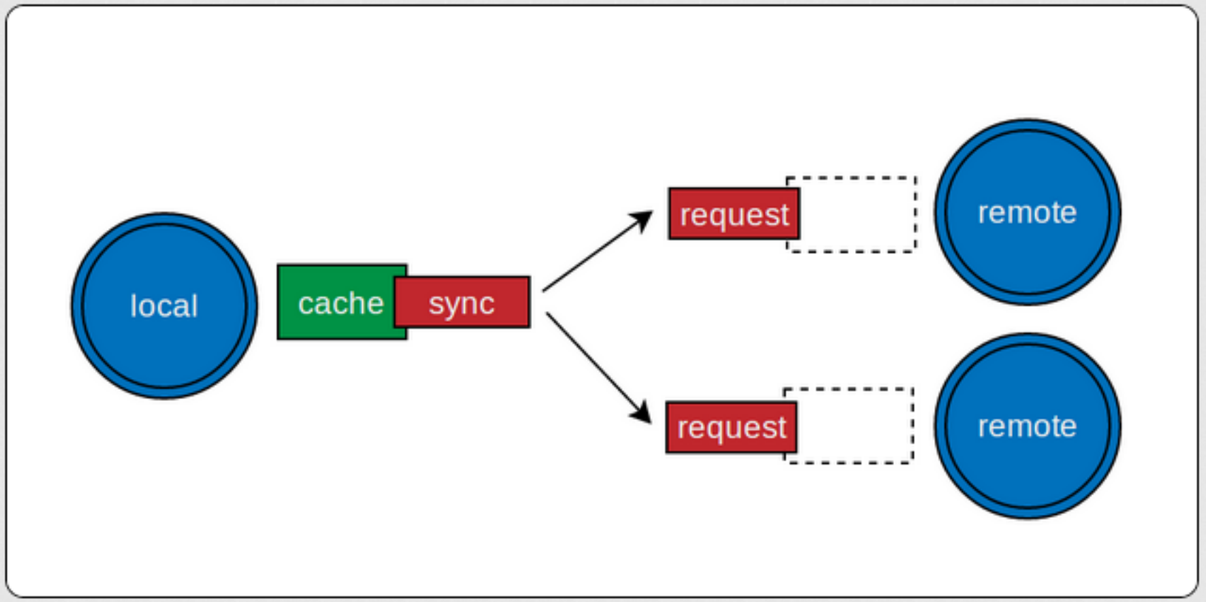


Reactive Data

```
/** using getter & setter for values **/

// define internal reference
var player = pg.player.data;

player.posX = 1;
player.keys = [ 'a', 'b', 'c', 'd' ];


// shared object betweeen all peers
pg.sync.state = 'start';

pg.sync.cookies = [];
pg.sync.cookies.push({ x: 100, y: 50, r: 20 });
```

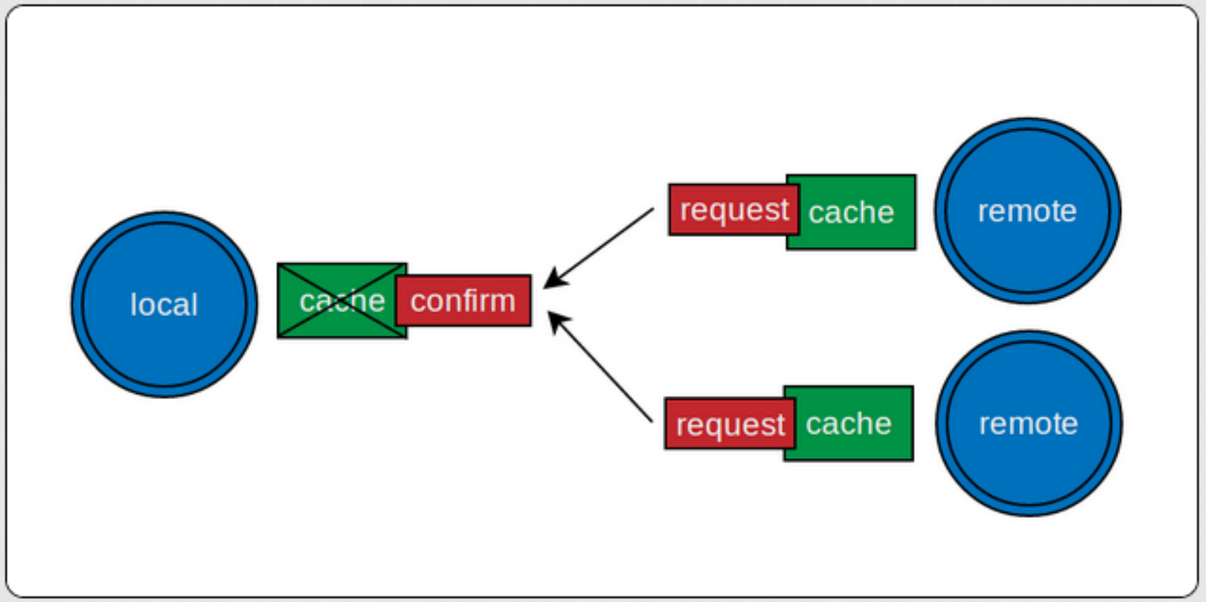
Information

Synchronized



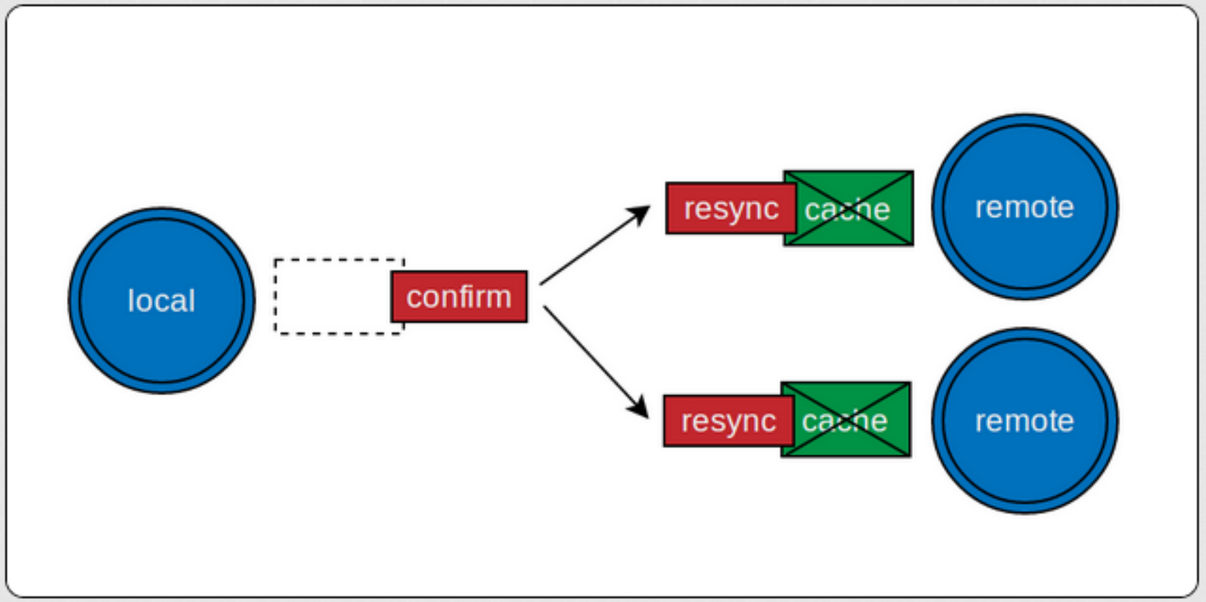
Information

Synchronized



Information

Synchronized



Synchronized Rendering

```
/** automatic synchronisation **/

// define internal reference
var entries = pg.data;

pg.loop( function render ( dt ) {

  for ( var i = 0, l = entries.length; i < l; i++ ) {

    console.log( entries[i] );
  }

});
```

Example

Quick Start

1.) include script tag

Quick Start

1.) include script tag

2.) setup room handler

Quick Start

1.) include script tag

2.) setup room handler

3.) login - create user

Quick Start

1.) include script tag

2.) setup room handler

3.) login - create user

4.) initialize the game

Quick Start

1.) include script tag

2.) setup room handler

3.) login - create user

4.) initialize the game

5.) use player, data & sync

Demo

pg-catch

~~http://demo.peergaming.net~~

"**Issue 2270**: Sending binary data fails using Chrome with SCTP data channels enabled"

Plans

reliable transfer

Plans

reliable transfer

interoperability

Plans

reliable transfer

interoperability

local development

Plans

reliable transfer

interoperability

local development

room options

Plans

reliable transfer

interoperability

local development

room options

custom handler

Questions