MIPRO 2011 – CTS track

Cross-context Web Browser Communication with Unified Communication Models and Context Types

Ivan Zuzak, *izuzak@gmail.com*
School of Electrical Engineering and Computing,
University of Zagreb, Zagreb, Croatia

Marko Ivankovic, *ivankovic.42@gmail.com*
Google GmbH., Zurich, Switzerland

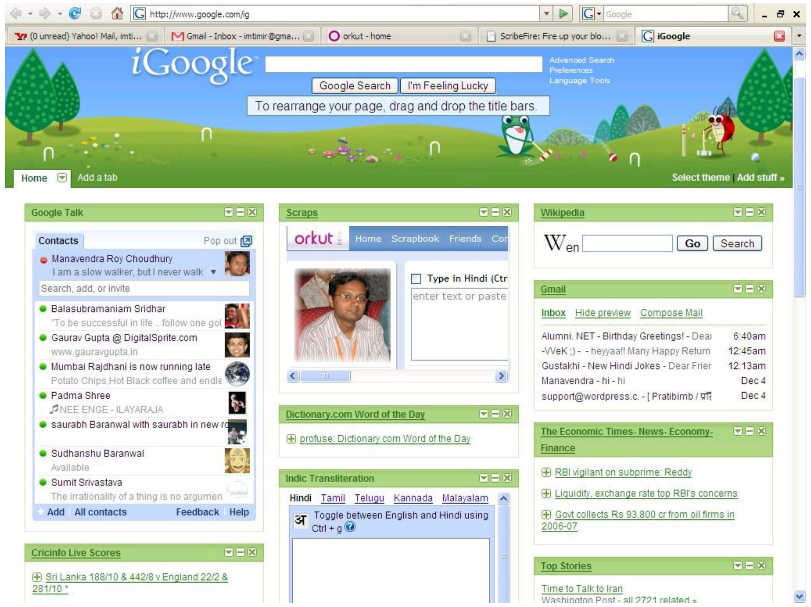
Ivan Budiselic, *ibudiselic@gmail.com*
School of Electrical Engineering and Computing,
University of Zagreb, Zagreb, Croatia

Agenda

- Motivation
  - Web Applications and Cross-Context Communication
  - Cross-Context Communication Ecosystem Systematization
- Pmrpc Library
  - Pmrpc API
  - Performance Evaluation
- Closing Remarks

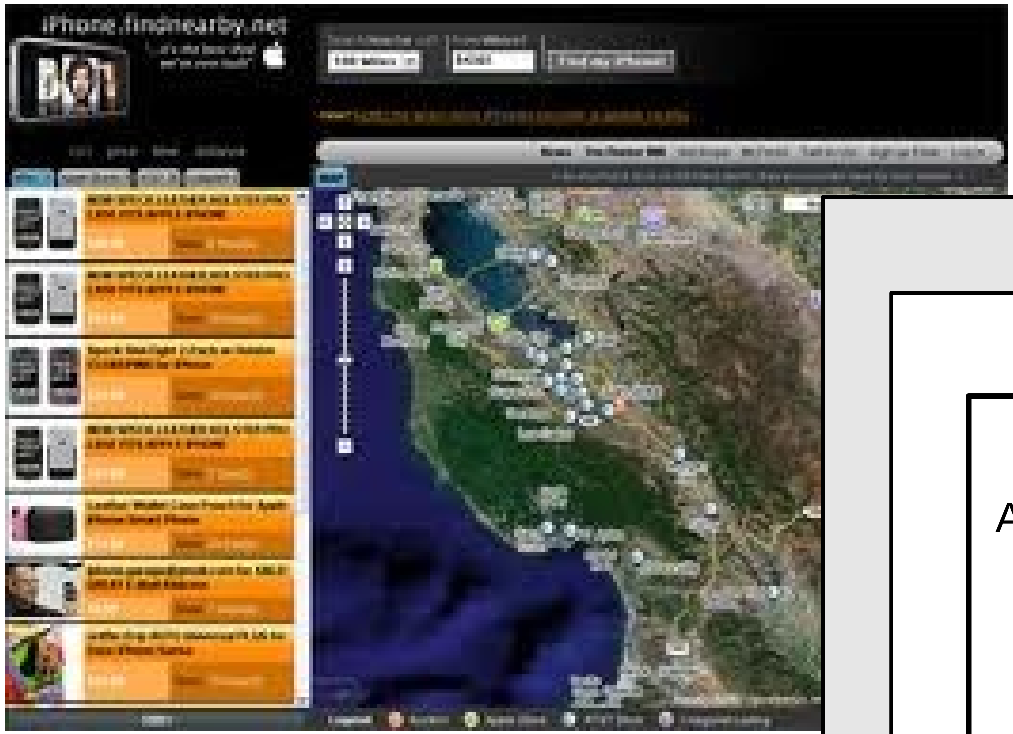
Motivation

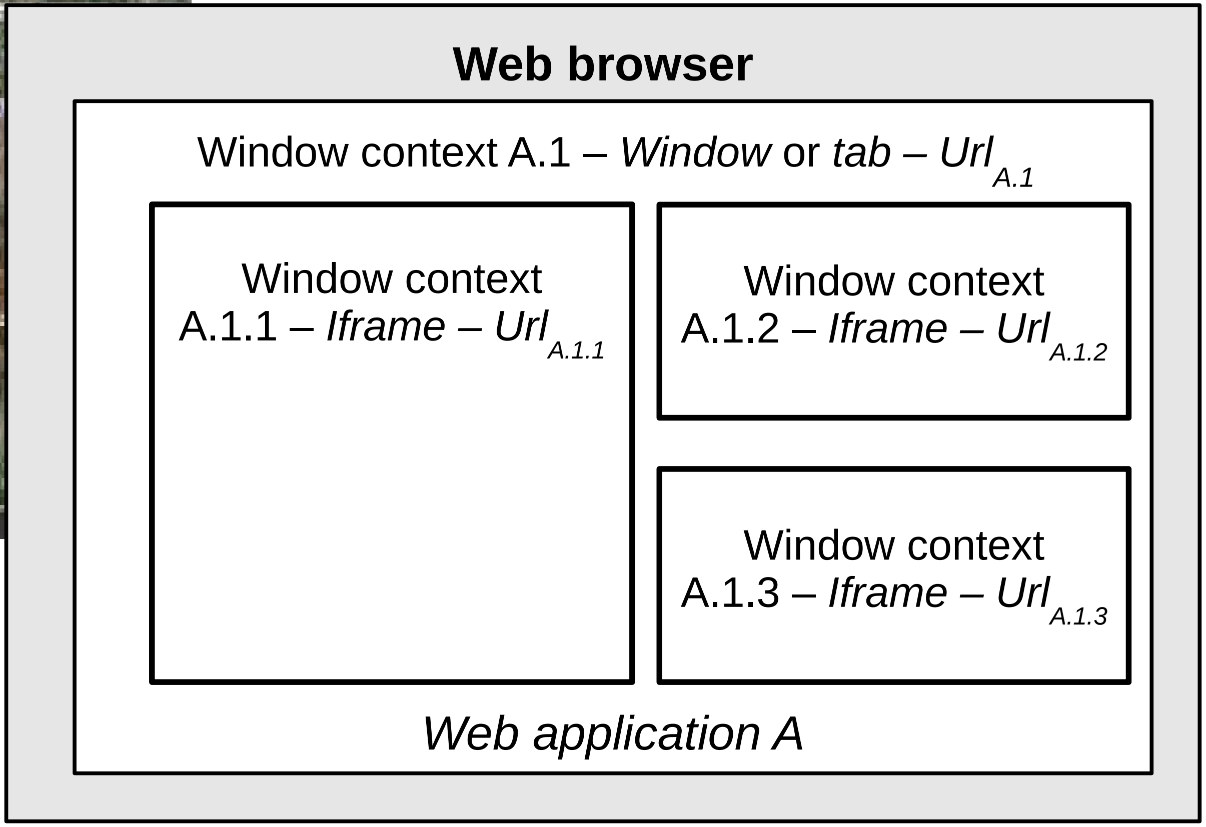
- Rich Internet Applications, Web 2.0
  - Mashups
  - Widget-based environments
  - Client-side background processing



Multi-context Web Applications

- Multi-context Web applications




Web browser
Window context A.1 – *Window* or *tab* – *Url*$_{A.1}$
Window context A.1.1 – *Iframe* – *Url*$_{A.1.1}$
Window context A.1.2 – *Iframe* – *Url*$_{A.1.2}$
Window context A.1.3 – *Iframe* – *Url*$_{A.1.3}$
Web application A

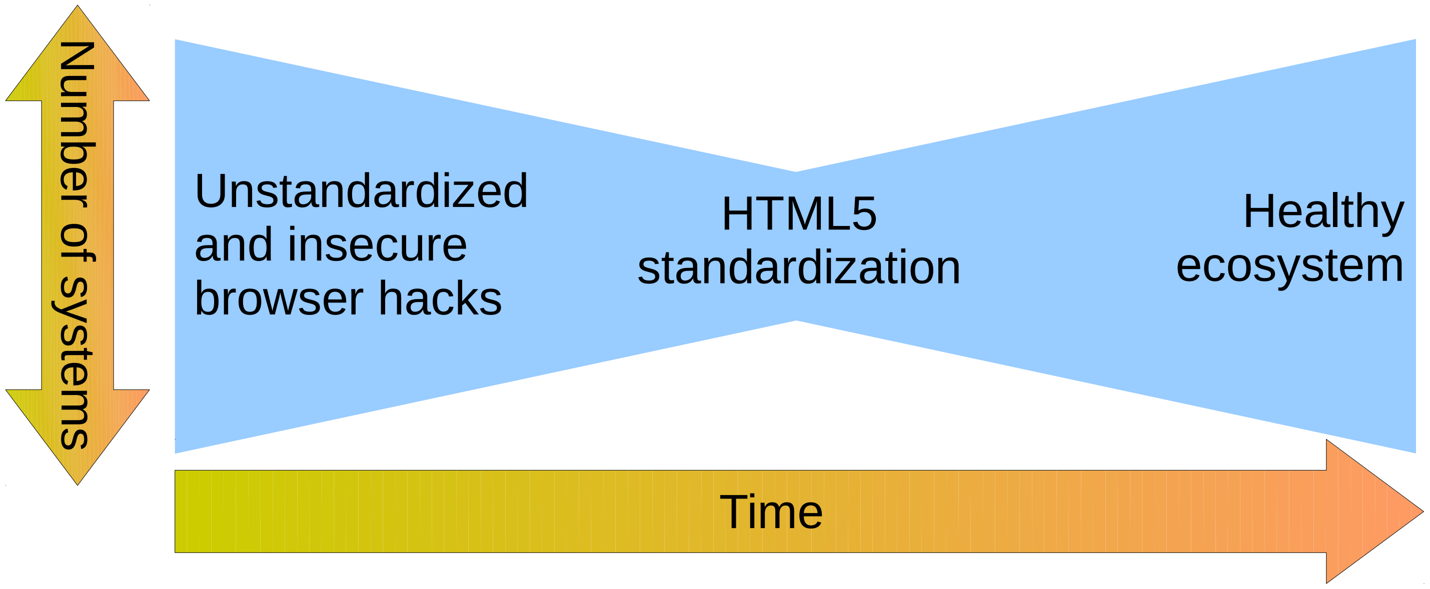
Multi-context Web Applications

- Multi-context Web applications
  - Context types
    - Visual Window contexts = GUI + event loop (HTML+JS)
    - Background Worker contexts = event loop (JS only)
  - Cross-context communication
    - Origin-based context isolation

Multi-context Web Applications

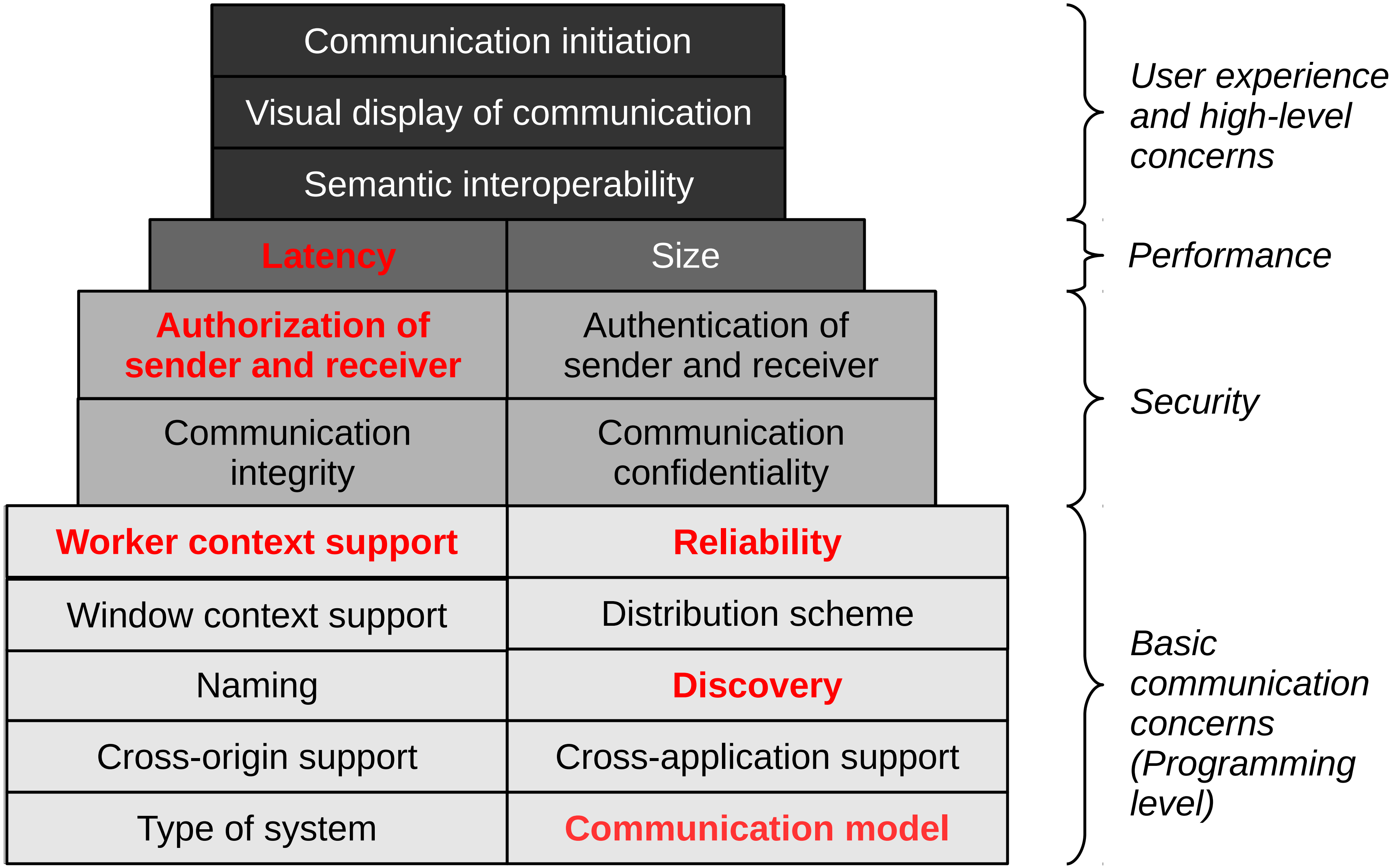
- Multi-context Web applications
  - Context types
    - Visual Window contexts = GUI + event loop (HTML+JS)
    - Background Worker contexts = event loop (JS only)
  - Cross-context communication
    - Origin-based context isolation
- Similar to Operating Systems
  - Processes and inter-process communication
  - *The browser is the new OS*

Cross-Contex Communication Systems

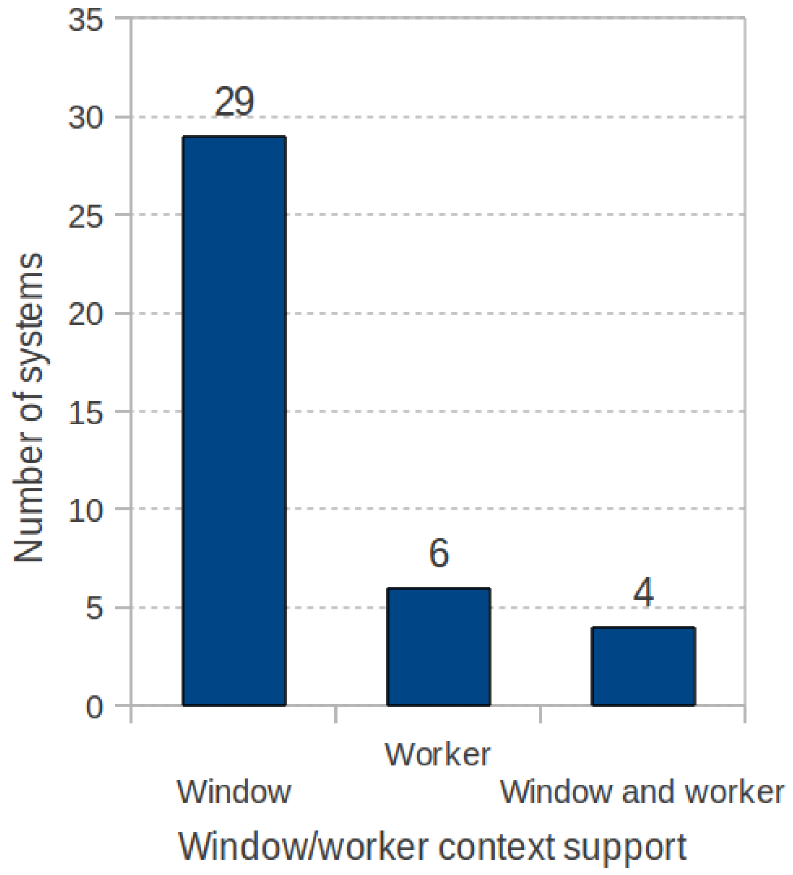


- No research on analyzing properties of such systems

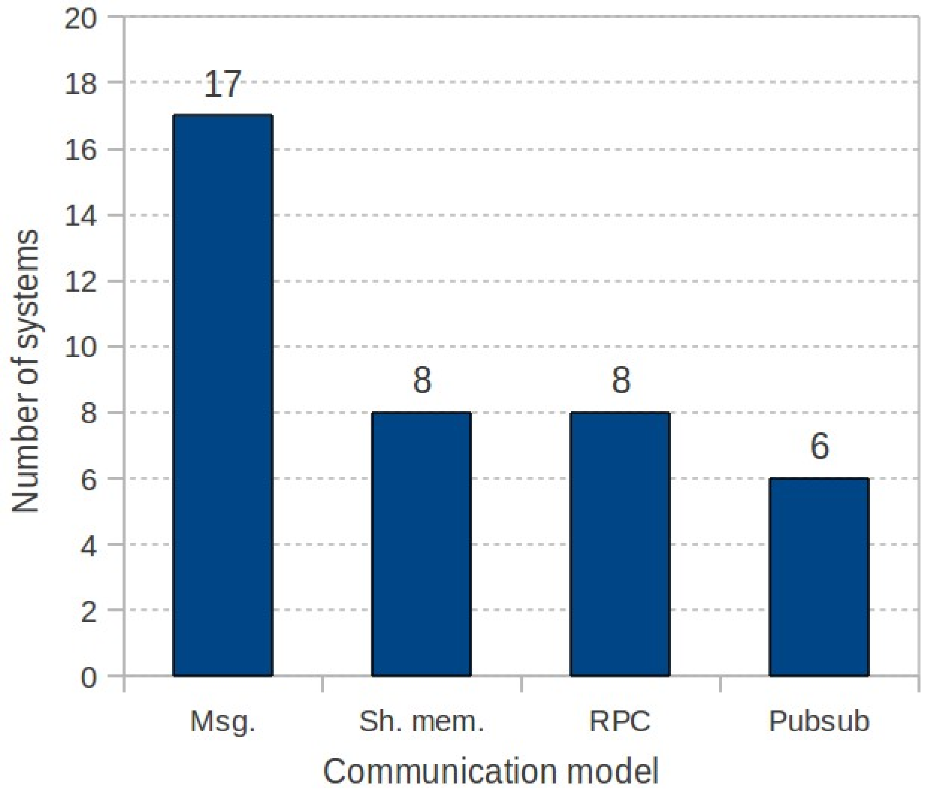
Cross-Context Communication Design Space


Communication initiation
Visual display of communication
Semantic interoperability
Latency
Size
Authorization of sender and receiver
Authentication of sender and receiver
Communication integrity
Communication confidentiality
Worker context support
Reliability
Window context support
Distribution scheme
Naming
Discovery
Cross-origin support
Cross-application support
Type of system
Communication model
User experience and high-level concerns
Performance
Security
Basic communication concerns (Programming level)

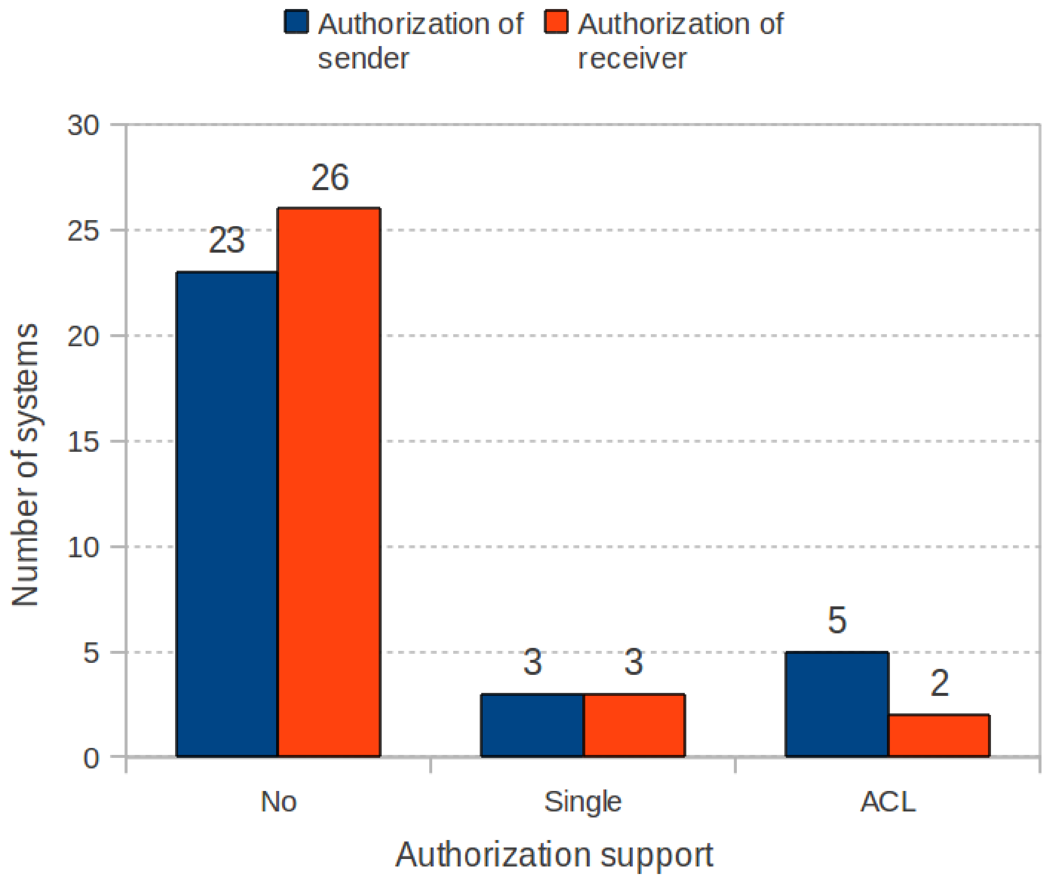
Cross-Contex Communication Ecosystem Systematization Results

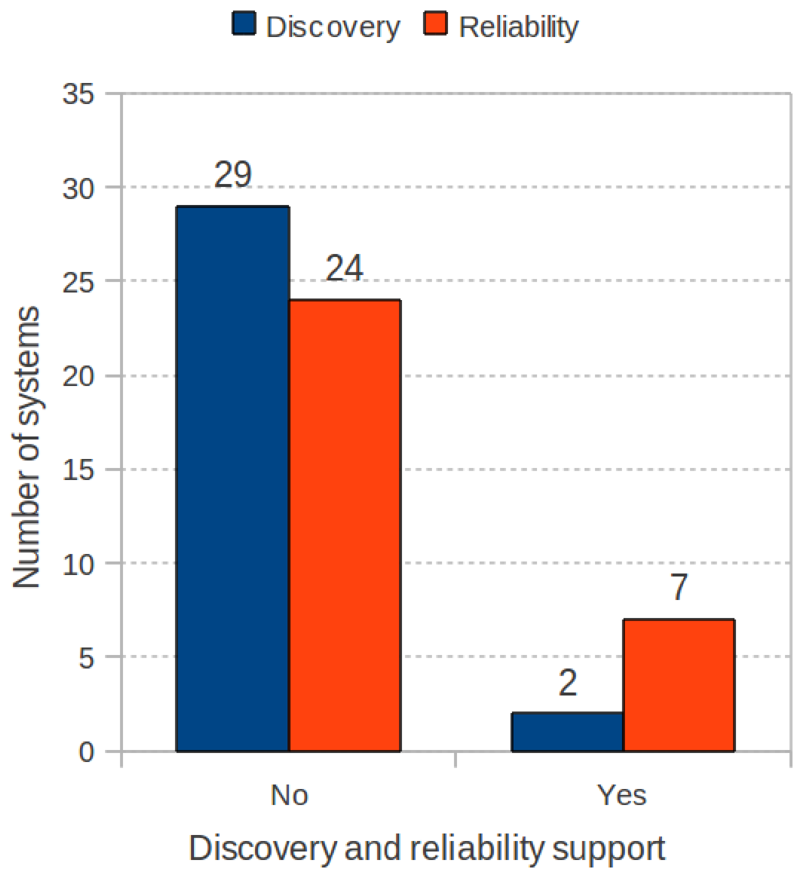


Cross-Contex Communication Ecosystem Systematization Results



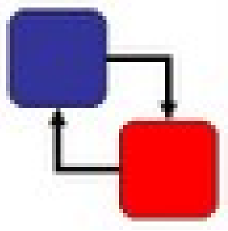
Cross-Contex Communication Ecosystem Systematization Results





Pmrpc Library

- Pmrpc
  - Cross-context communication JavaScript library
    - Advanced communication features
  - Rule of least liability
    - *“Systems should minimize the liability that the user undertakes to ensure application security.”*
    - Hiding complexity of cross-context communication **is desired**



Pmrpc Library

- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)

Pmrpc Library

- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)
- **Unified Web Worker and Window context support**
  - Wrap and unify browser primitives

Pmrpc Library

- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)
- **Unified Web Worker and Window context support**
  - Wrap and unify browser primitives
- **Unified communication models**
  - Message-based communication
  - Remote procedure call
  - Publish-subscribe

Pmrpc Library

- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)
- **Unified Web Worker and Window context support**
  - Wrap and unify browser primitives
- **Unified communication models**
  - Message-based communication
  - Remote procedure call
  - Publish-subscribe
- **Discovery support**
  - Discovery of contexts, procedures and channels

Pmrpc Library

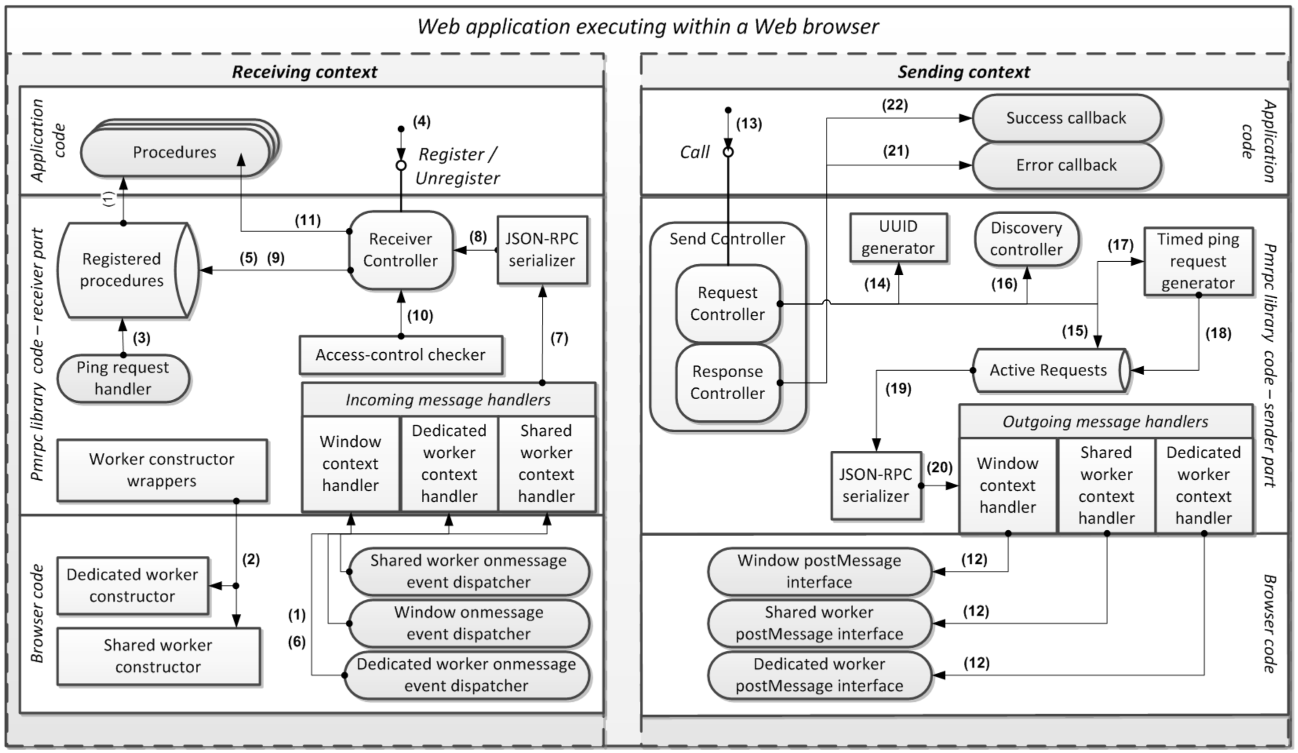
- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)
- **Unified Web Worker and Window context support**
  - Wrap and unify browser primitives
- **Unified communication models**
  - Message-based communication
  - Remote procedure call
  - Publish-subscribe
- **Discovery support**
  - Discovery of contexts, procedures and channels
- **Reliability support**
  - Message-level handshake and retry mechanism

Pmrpc Library

- **Client-side framework**
  - No server components used
  - Based on HTML5 and WebWorker postMessage primitives (secure message-passing mechanisms)
- **Unified Web Worker and Window context support**
  - Wrap and unify browser primitives
- **Unified communication models**
  - Message-based communication
  - Remote procedure call
  - Publish-subscribe
- **Discovery support**
  - Discovery of contexts, procedures and channels
- **Reliability support**
  - Message-level handshake and retry mechanism
- **Authorization support**
  - Whitelist access control list mechanism

Pmrpc Architecture

- Not enough time, see the paper
  - Pmrpc = wrapper for different context types, communication models, reliability …



Web application executing within a Web browser
Receiving context
Sending context
Application code
Pmrpc library code – receiver part
Browser code
Procedures
(4)
Register / Unregister
(1)
(11)
Registered procedures
(5) (9)
Receiver Controller
(8)
JSON-RPC serializer
(10)
(3)
Ping request handler
Access-control checker
(7)
Incoming message handlers
Window context handler
Dedicated worker context handler
Shared worker context handler
Worker constructor wrappers
(2)
Dedicated worker constructor
Shared worker constructor
(1)
(6)
Shared worker onmessage event dispatcher
Window onmessage event dispatcher
Dedicated worker onmessage event dispatcher
Call
(13)
(22)
Success callback
(21)
Error callback
Application code
Send Controller
Request Controller
Response Controller
UUID generator
Discovery controller
(17)
Timed ping request generator
(14)
(16)
(15)
(18)
Active Requests
(19)
JSON-RPC serializer
(20)
Outgoing message handlers
Window context handler
Shared worker context handler
Dedicated worker context handler
Pmrpc library code – sender part
(12)
Window postMessage interface
(12)
Shared worker postMessage interface
(12)
Dedicated worker postMessage interface
Browser code

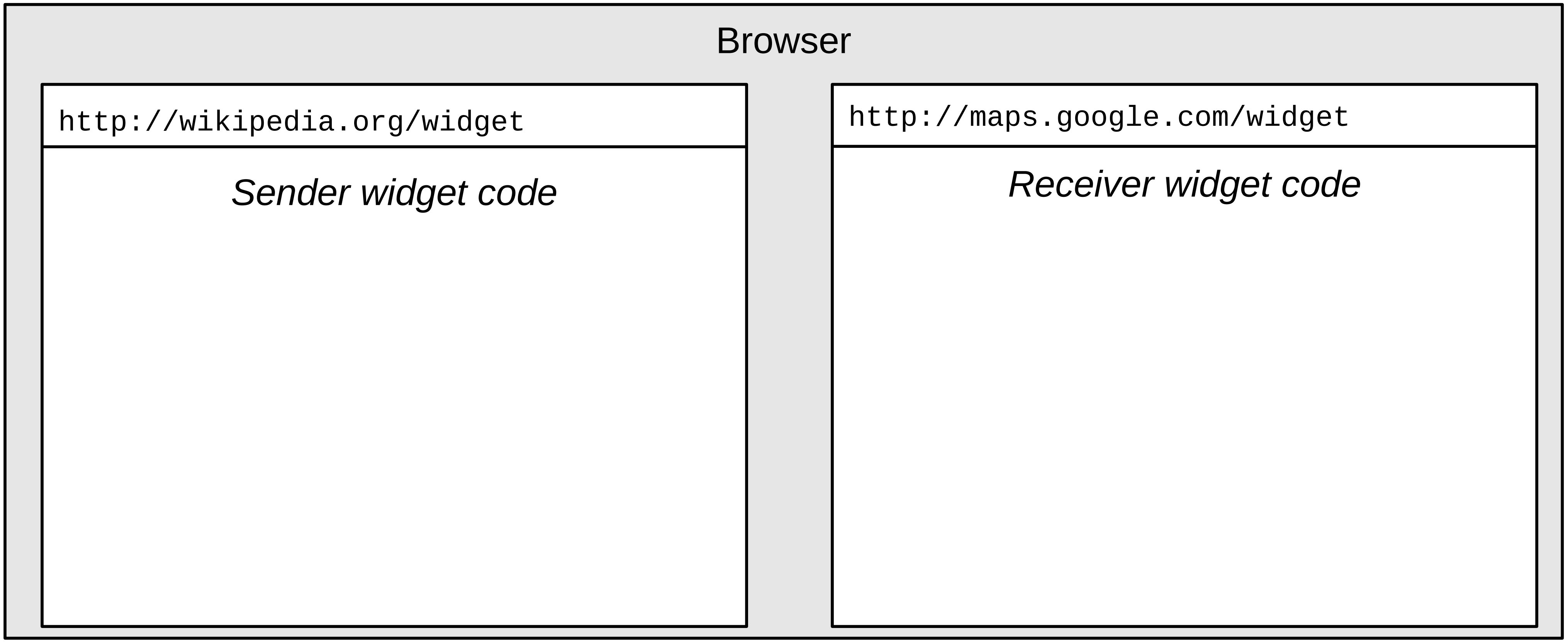
Pmrpc API

- **`pmrpc.register(handler, procedureName, ?acl)`**
  - Register procedure (for RPC)
  - Subscribe to channel (for pubsub)
  - ACL whitelist of authorization rights
- **`pmrpc.unregister(procedureName)`**
  - Unregister procedure (for RPC)
  - Unsubscribe from channel (for pubsub)

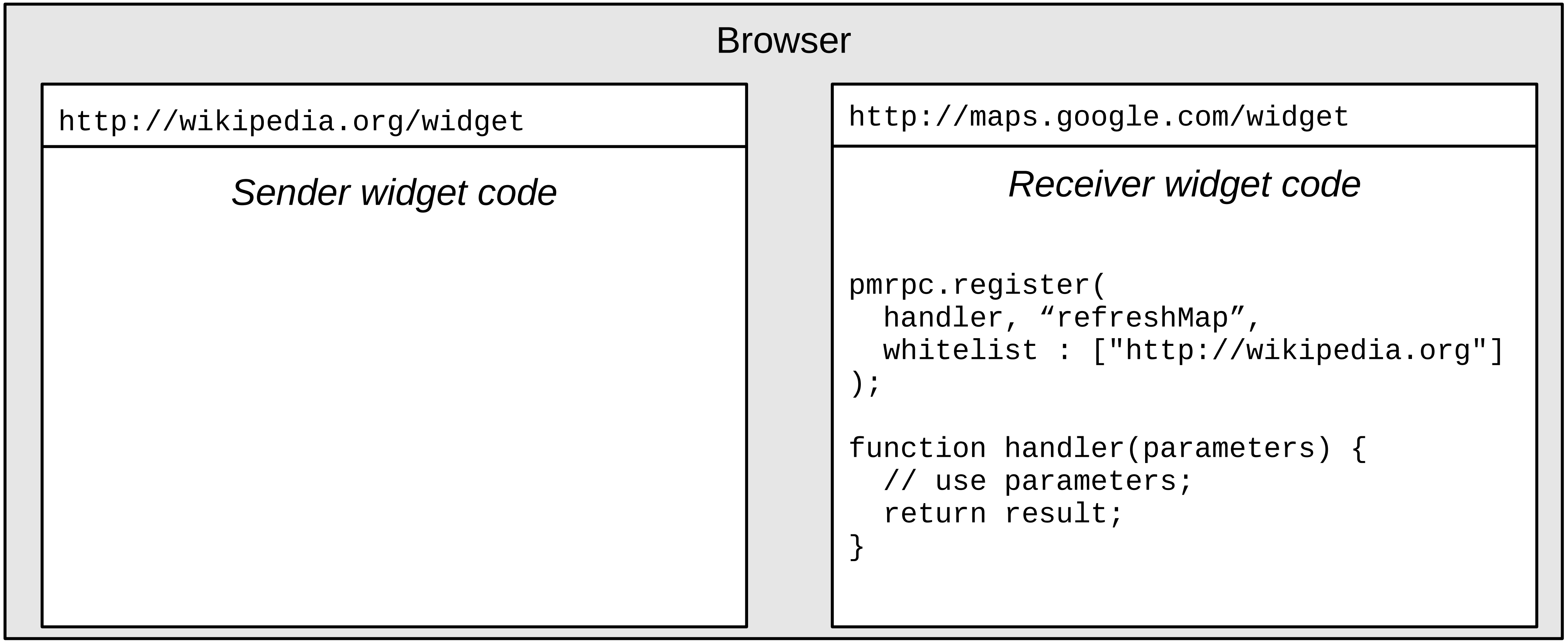
Pmrpc API

- **`pmrpc.call(procedureName, ?destinationContext, ?args, ?acl, ?retries, ?timeout, ?onSuccess, ?onError)`**
  - Invoke procedure (for RPC)
    - destinationContext for addressing
    - onSuccess and onError handlers
  - Publish to channel (for pubsub)
    - Automatic discovery of destination contexts
  - ACL whitelist of authorization rights
  - Messages are retried in case of errors

Pmrpc API

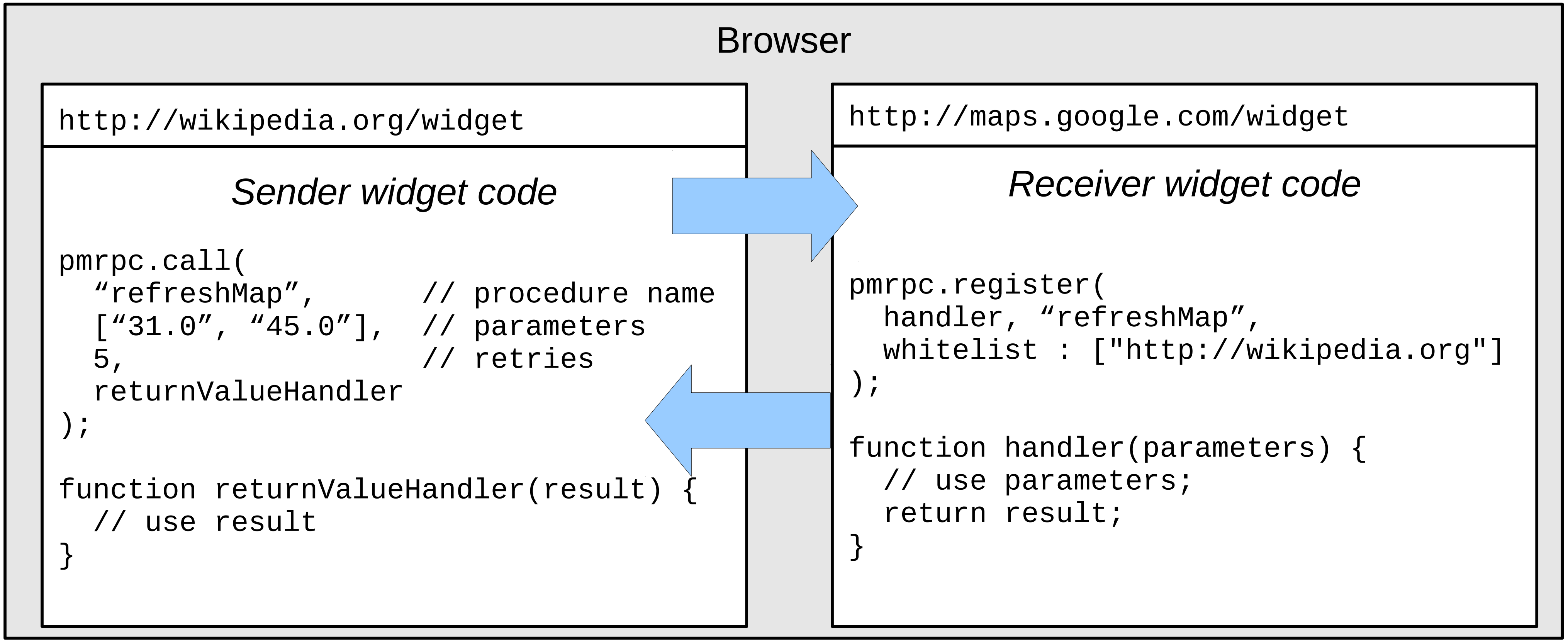

Browser
http://wikipedia.org/widget
Sender widget code
http://maps.google.com/widget
Receiver widget code

Pmrpc API



```
pmrpc.register(
  handler, “refreshMap”,
  whitelist : ["http://wikipedia.org"]
);

function handler(parameters) {
  // use parameters;
  return result;
}
```

Pmrpc API



```
pmrpc.call(
  “refreshMap”,       // procedure name
  [“31.0”, “45.0”],   // parameters
  5,                  // retries
  returnValueHandler
);

function returnValueHandler(result) {
  // use result
}
```

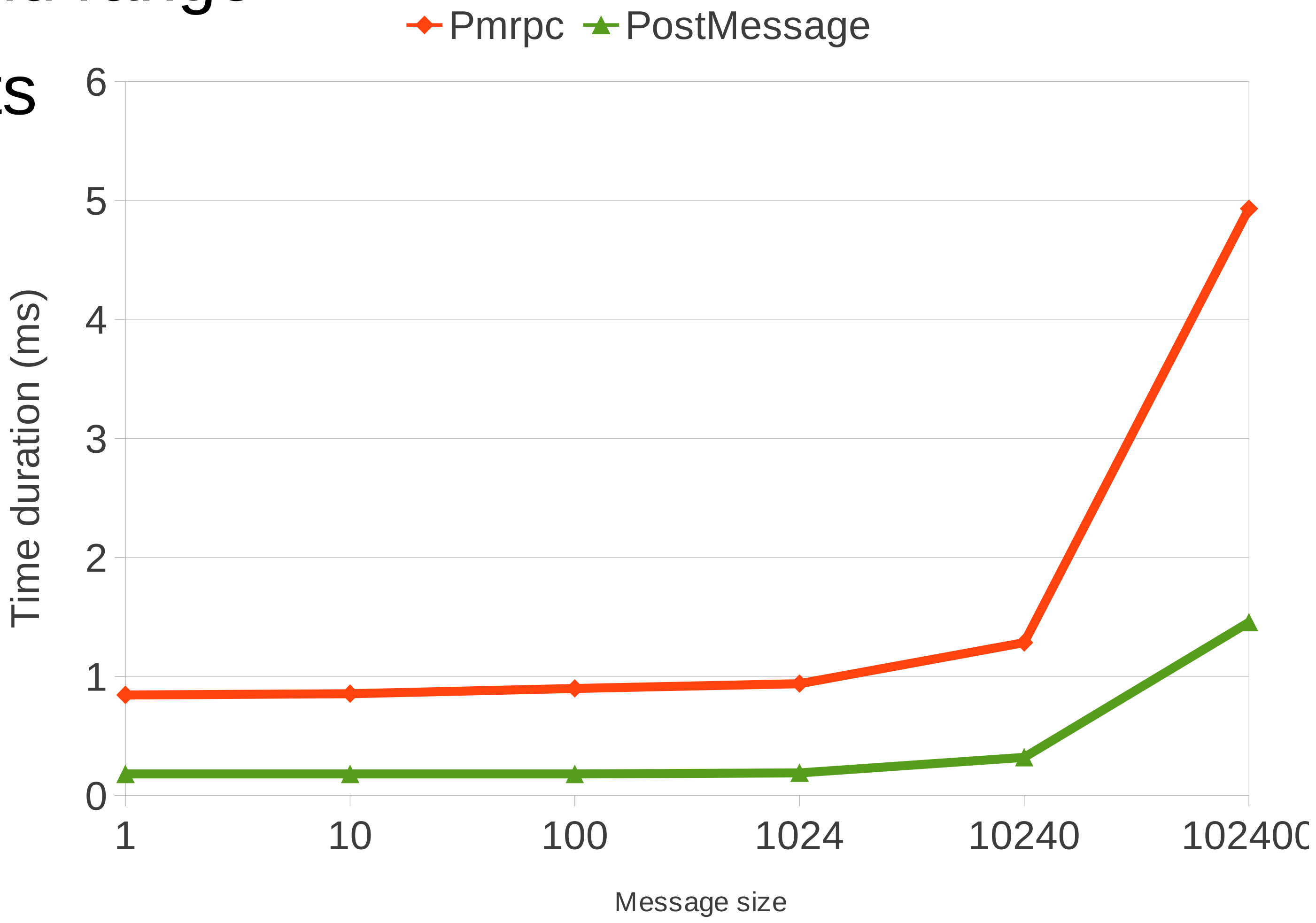
```
pmrpc.register(
  handler, “refreshMap”,
  whitelist : ["http://wikipedia.org"]
);

function handler(parameters) {
  // use parameters;
  return result;
}
```

Performance Evaluation

- What is the tradeoff of complexity?
  - Performance
- Experimental measurements
  - Data transfer (round trip)
    - Native postMessage primitive
    - Pmrpc library
  - Different message sizes

Performance Evaluation

- Pmrpc is 4 times slower than postMessage
- Still in milisecond range
- Expected results
    - Serialization
    - Reliability



Conclusion

- Cross-context communication
  - The foundation of future Web applications
- Pmrpc
  - Open-source and free browser library (MIT license)
  - Hides cross-context communication complexity
  - Performance analysis
    - ~4 times slower than native browser primitives
    - Still in milisecond range, fast enough

Questions?

Thank you!

Pmrpc library:

http://code.google.com/p/pmrpc/

Contact:

izuzak@gmail.com
ivankovic.42@gmail.com
ibudiselic@gmail.com