WebRTC-Wonder Rückruffunktionalität

Johannes Hamfler

Hochschule für Telekommunikation Leipzig

johannes.hamfler@hftl.de

16. Februar 2015

Übersicht

Anforderungen

Aufgabe

- Rückruffunktionalität ermöglichen
- Robustheit testen
- Zuweisung des korrekten `messagingStubs` beim Rückruf
- Minimales Eingreifen in den Quelltext bzw. die Funktionalität

Zusatzaufgabe

- Binden des `messagingStubs` an eine conversation

Begriffsklärung

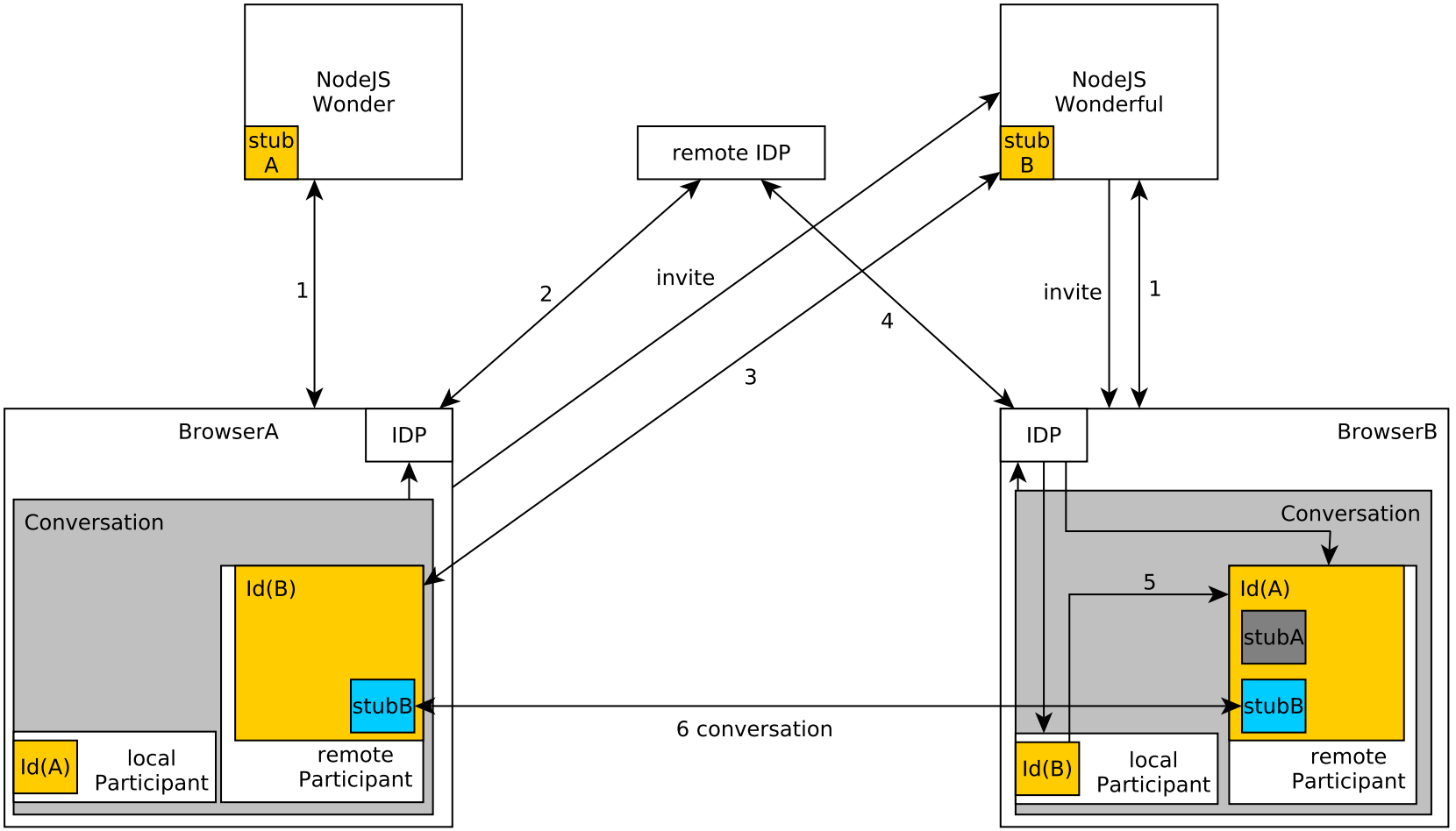
- Rückrufproblem tritt nur bei unterschiedlichen Domains / Messaging-Servern auf
- Zwei participants auf gleichem Server sind nicht betroffen
- `messagingStub` ist ein Objekt zum wrappen einer `messagingStub`-Implementierung

`messagingStub` Beispiel

```
▼ messagingStub: MessagingStub
  ▶ buffer: Array[0]
  ▶ impl: MessagingStub_NodeJS_1338
  ▶ listeners: Array[3]
    message: "stub downloaded from: http://127.0.0.1:80/stubs/tlabs/MessagingStub_NodeJS_1338.js"
  ▶ __proto__: MessagingStub
```

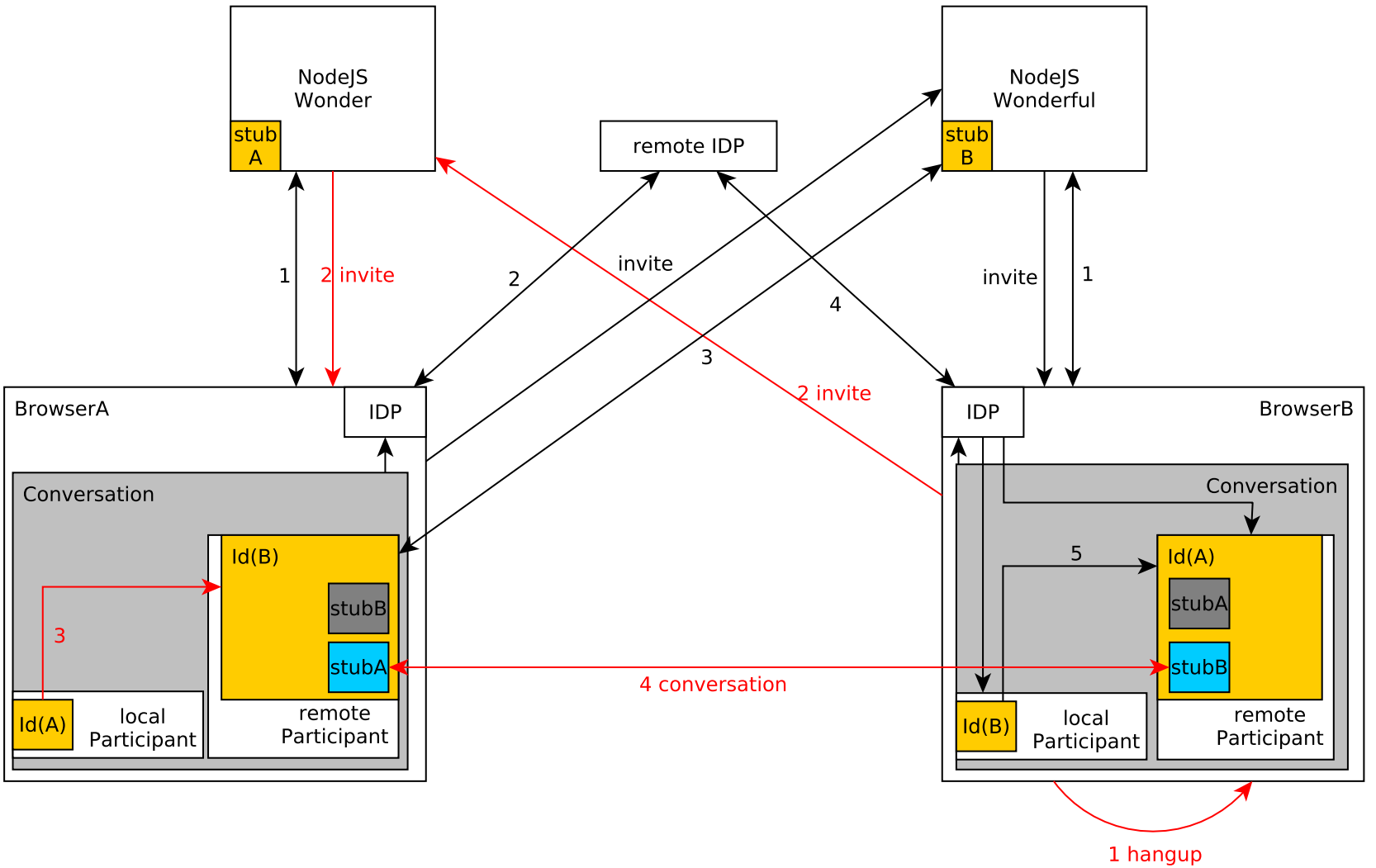
no-`messagingStub` Beispiel

```
▼ messagingStub: MessagingStub
  ▶ buffer: Array[0]
    impl: null
  ▶ listeners: Array[3]
    message: "No implementation downloaded and assigned to this stub yet!"
  ▶ __proto__: MessagingStub
```



- Remote `identity` bekommt korrekten `messagingStub`
- Eigener `messagingStub` wird beim Angerufenen kopiert

Rückruf



- Remote identity behält nach hangup den `messagingStub`

Teillösung 1

Conversation.prototype.acceptInvitation Zeile 308

```
1 toIdentity.originalStub = toIdentity.
    messagingStub;
2 toIdentity.messagingStub = that.myParticipant.
    identity.messagingStub;
```

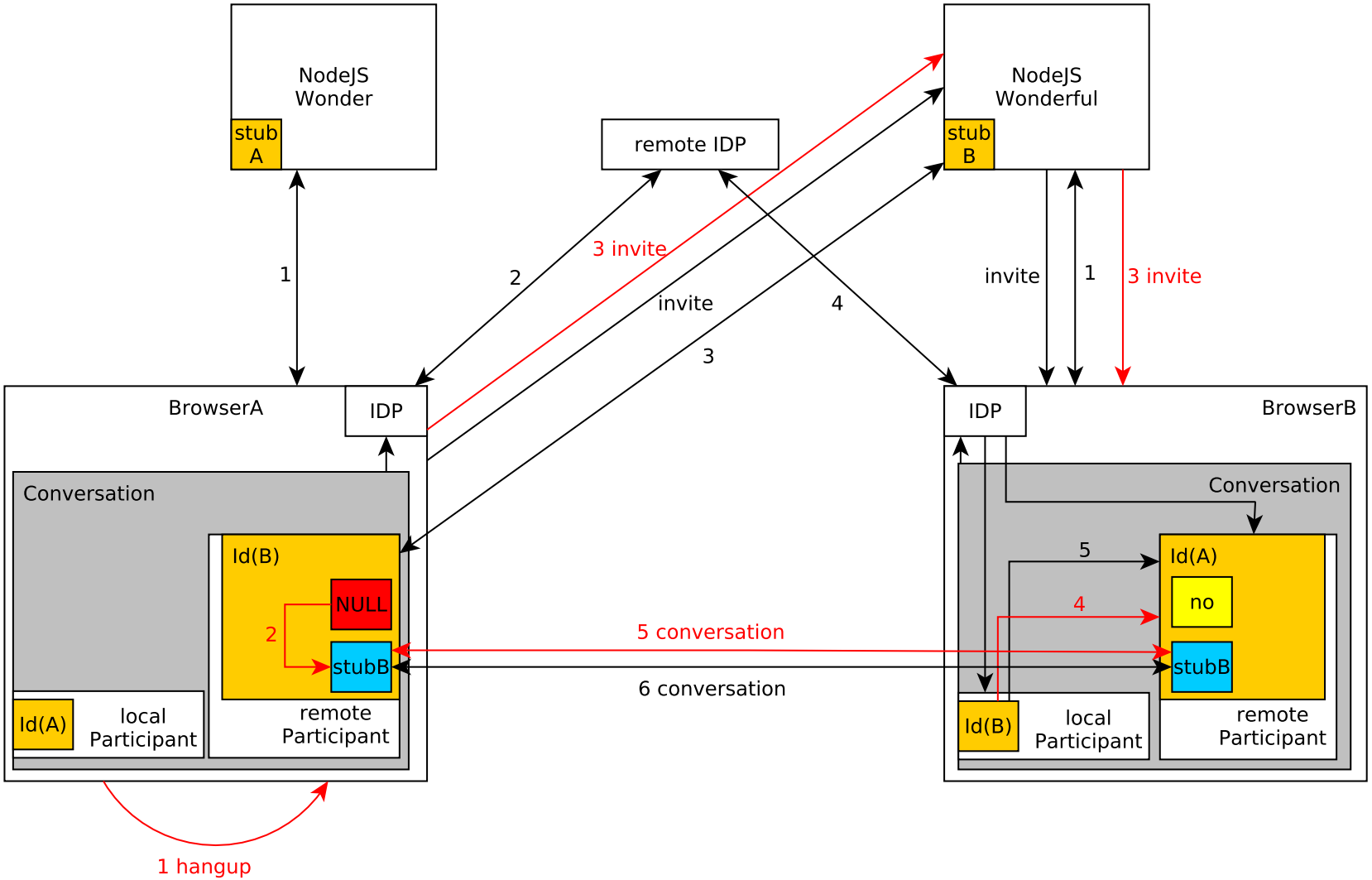
Conversation.prototype.close Zeile 498

```
1 element.identity.messagingStub = element.
    identity.originalStub;
```

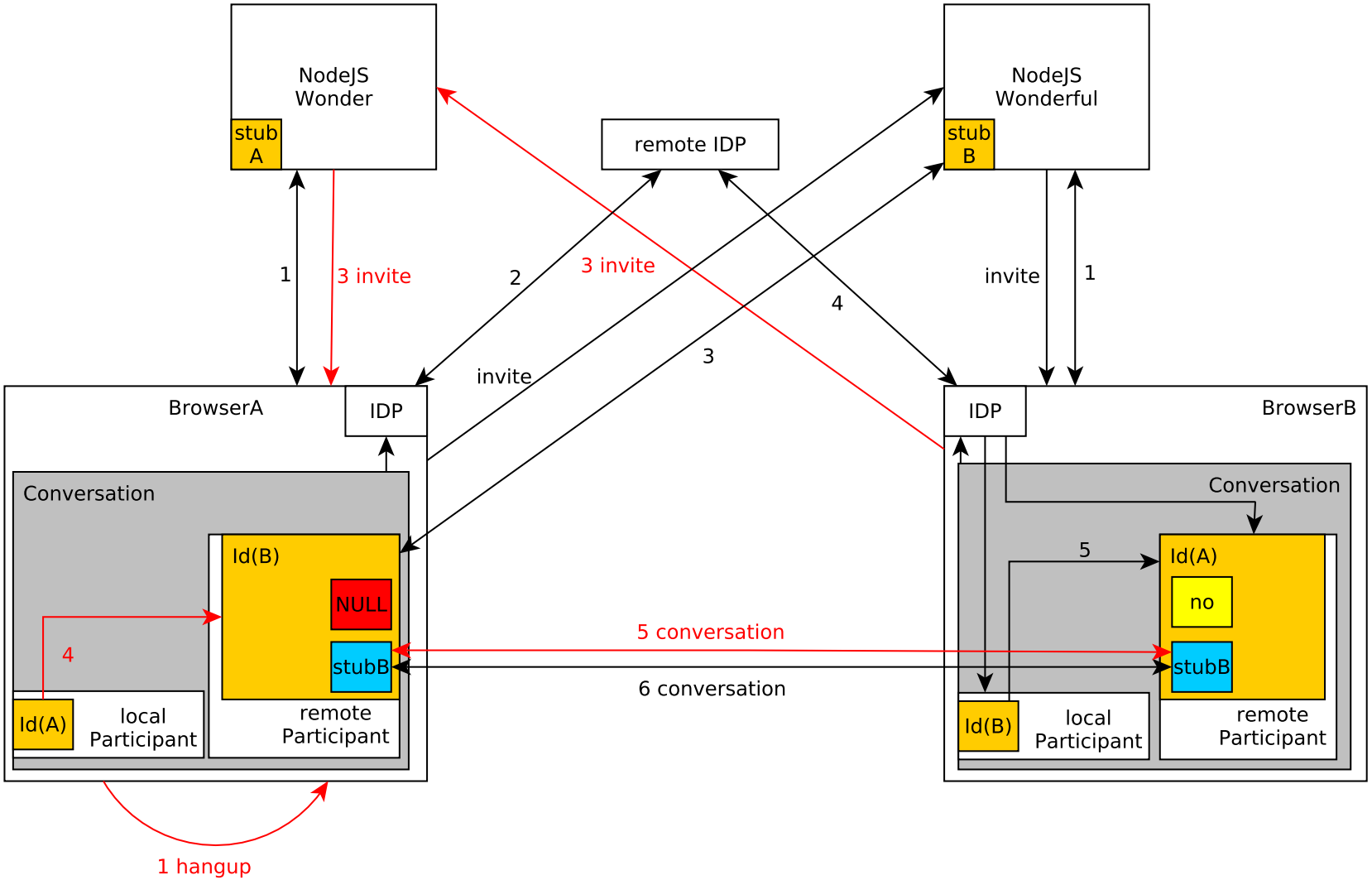
Conversation.prototype.bye Zeile 522

```
1 element.identity.messagingStub = element.
    identity.originalStub;
```

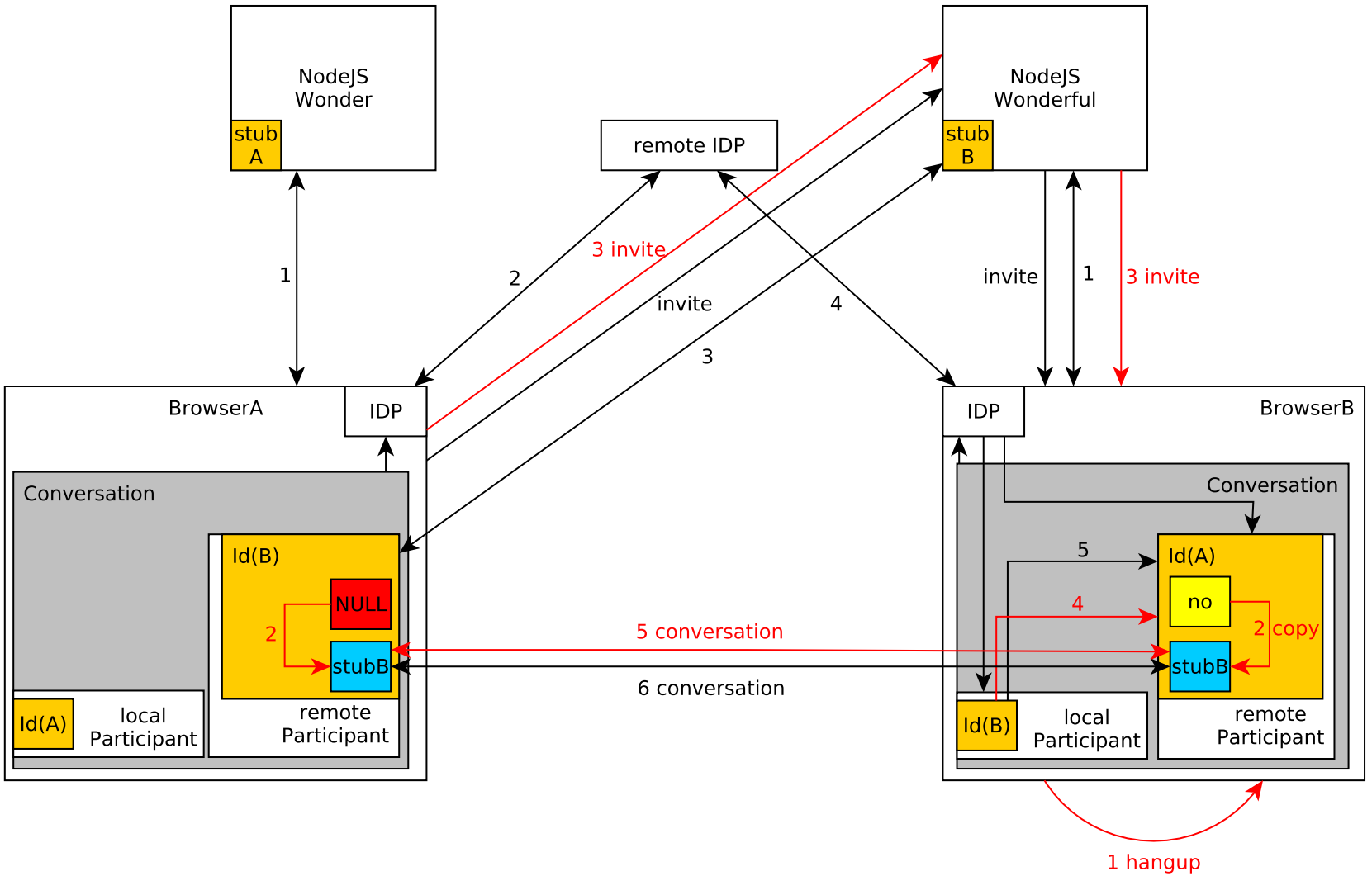
Teillösung 1



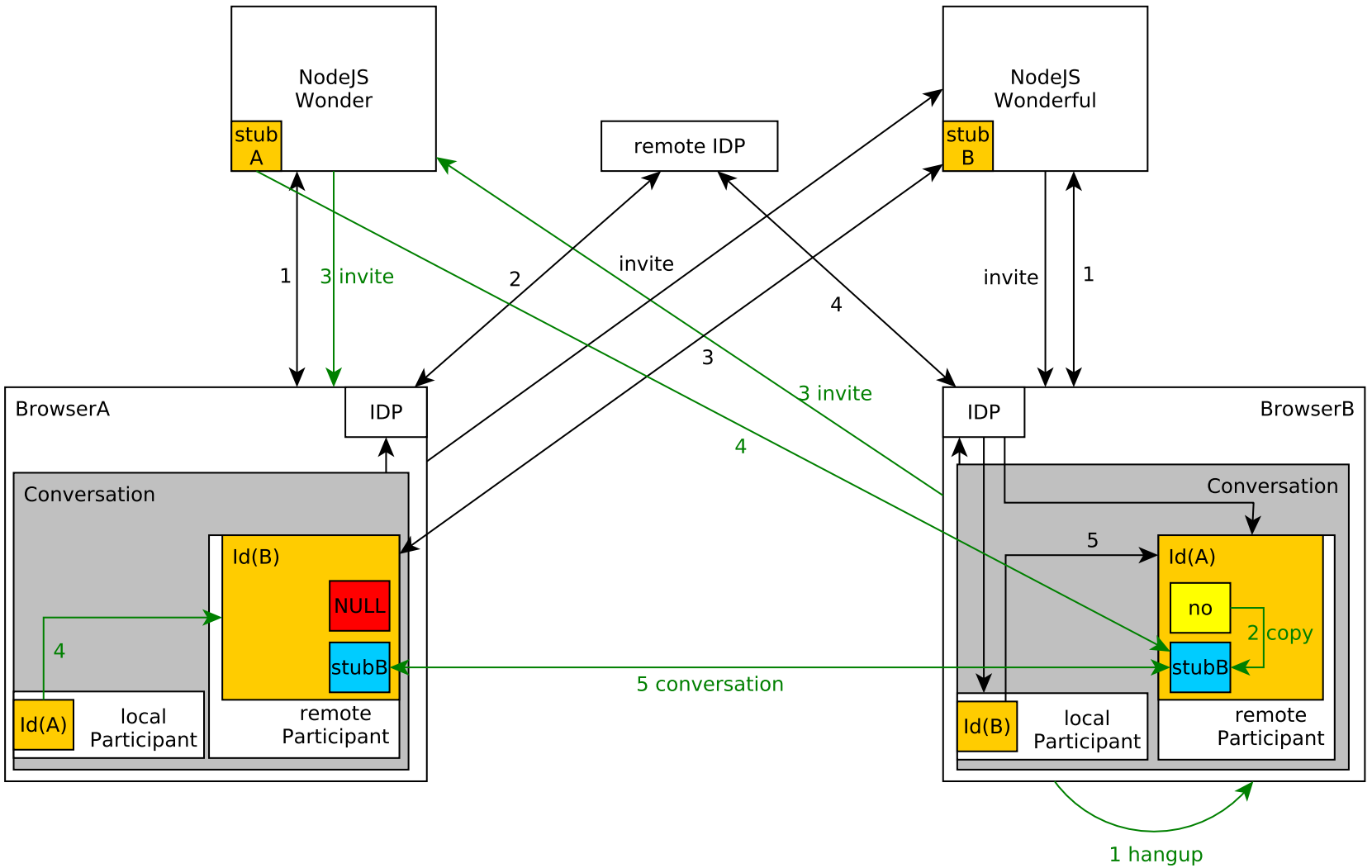
Teillösung 1



Teillösung 1



Teillösung 1



Teillösung 1

- Rückruf jetzt möglich, jedoch nicht
  - Wenn der Anrufer aufhängt
  - Wenn der Angerufene auflegt und der Anrufer erneut anruft
- Kopieren des `messagingStubs` löst das Problem nur teilweise beim Auflegen des Angerufenen
- 1/4 Probleme gelöst, jedoch auch 1/4 Probleme erzeugt
- Nicht funktionstüchtig
  - A hangup & A call B
  - A hangup & B call A
  - B hangup & A call B
- Funktionstüchtig
  - B hangup & B call A

Teillösung 1 Nullstub-Fehler umgehen

Identity.prototype.resolve Zeile 88

```
1  Identity.prototype.resolve = function( callback ) {
2      var that = this;
3      console.log( "resolving identity: " + this.
           rtcIdentity);
4      // if ( ! this.messagingStub.impl ) {
5      // oder: if ( ! (this.messagingStub && this.
           messagingStub.impl) )
6      if ( ! this.messagingStub || ! this.messagingStub.
           impl ) {
7          var knownStub = Idp.getInstance().getResolvedStub
               (this.messagingStubLibUrl);
```

Teillösung 2

- Im angerufenen remote participant wird nur noch bei `Conversation.prototype.bye` der `messagingStub` verändert

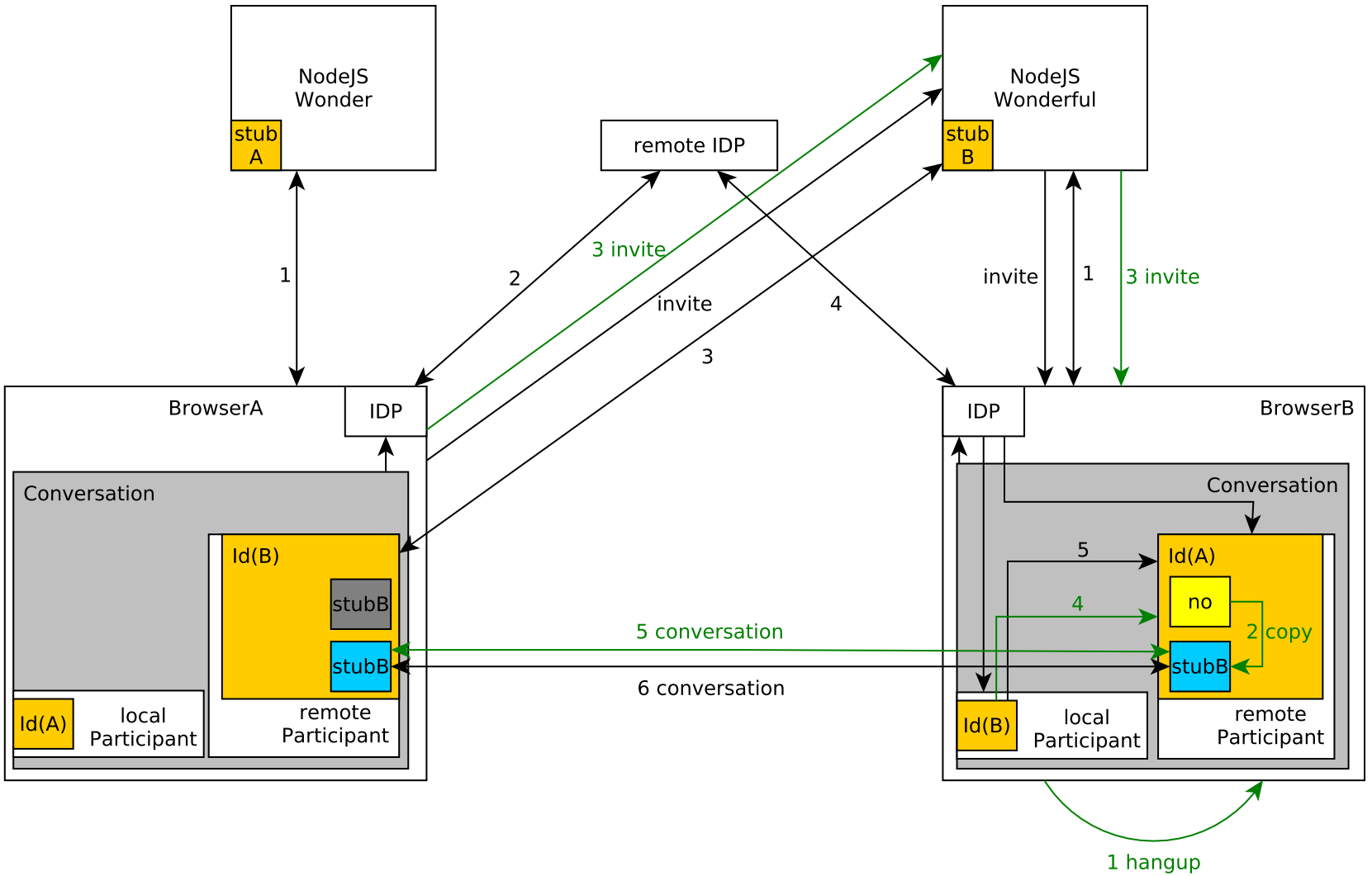
Conversation.prototype.acceptInvitation

```
1 toIdentity.originalStub = toIdentity.messagingStub;
2 toIdentity.messagingStub = that.myParticipant.identity.
      messagingStub;
```

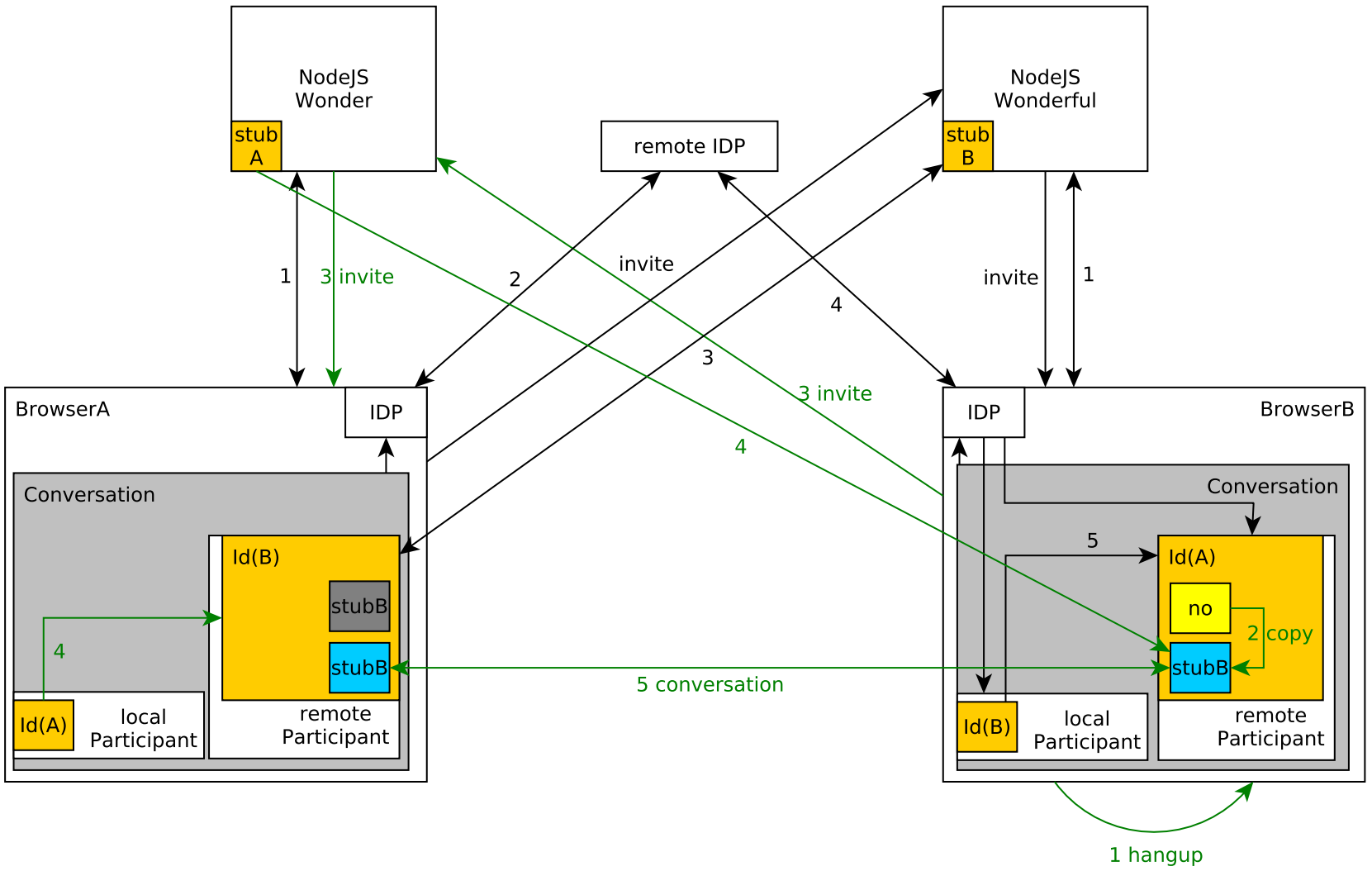
Conversation.prototype.bye

```
1 element.identity.messagingStub = element.identity.
      originalStub;
```

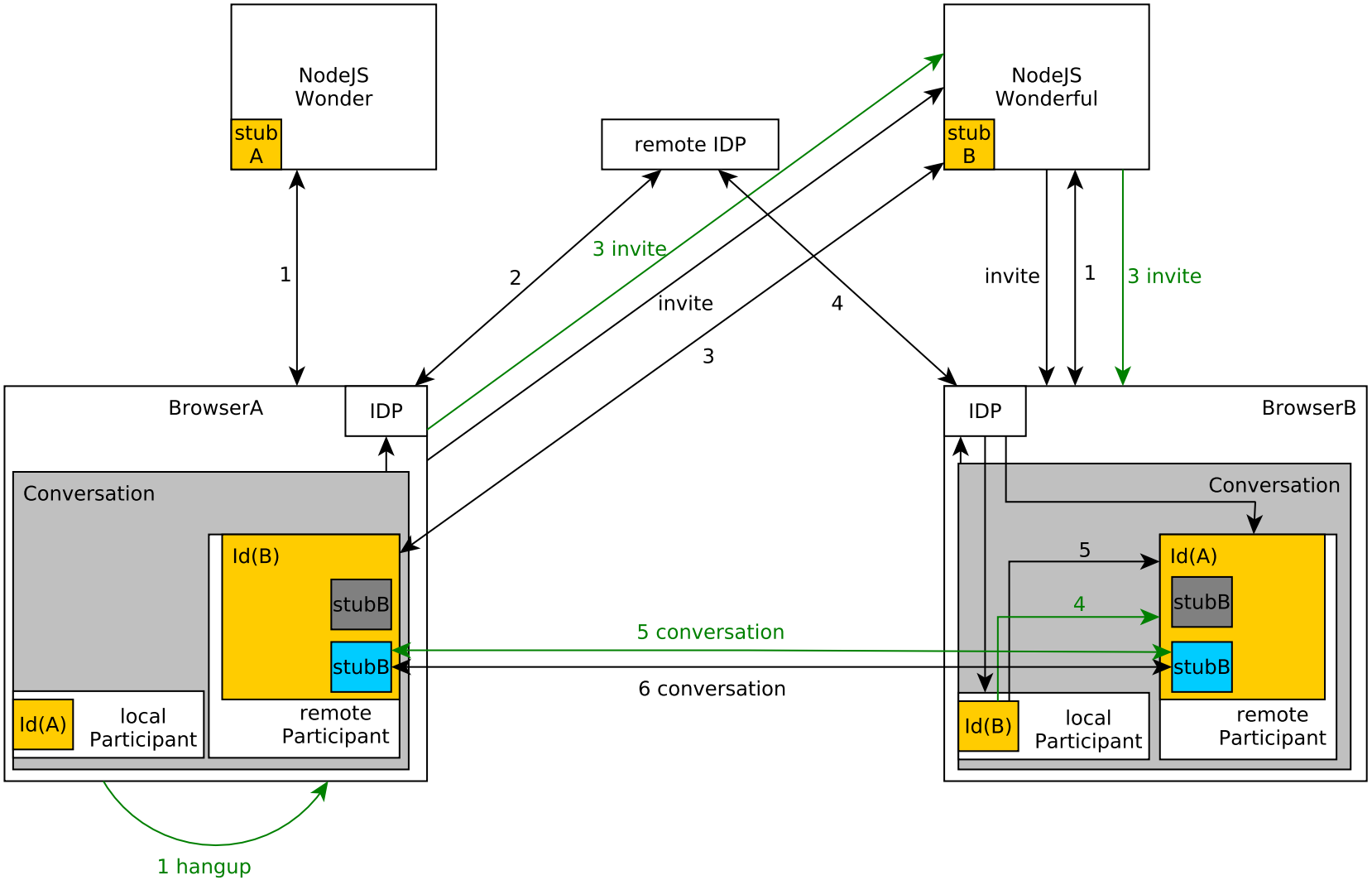
Teillösung 2



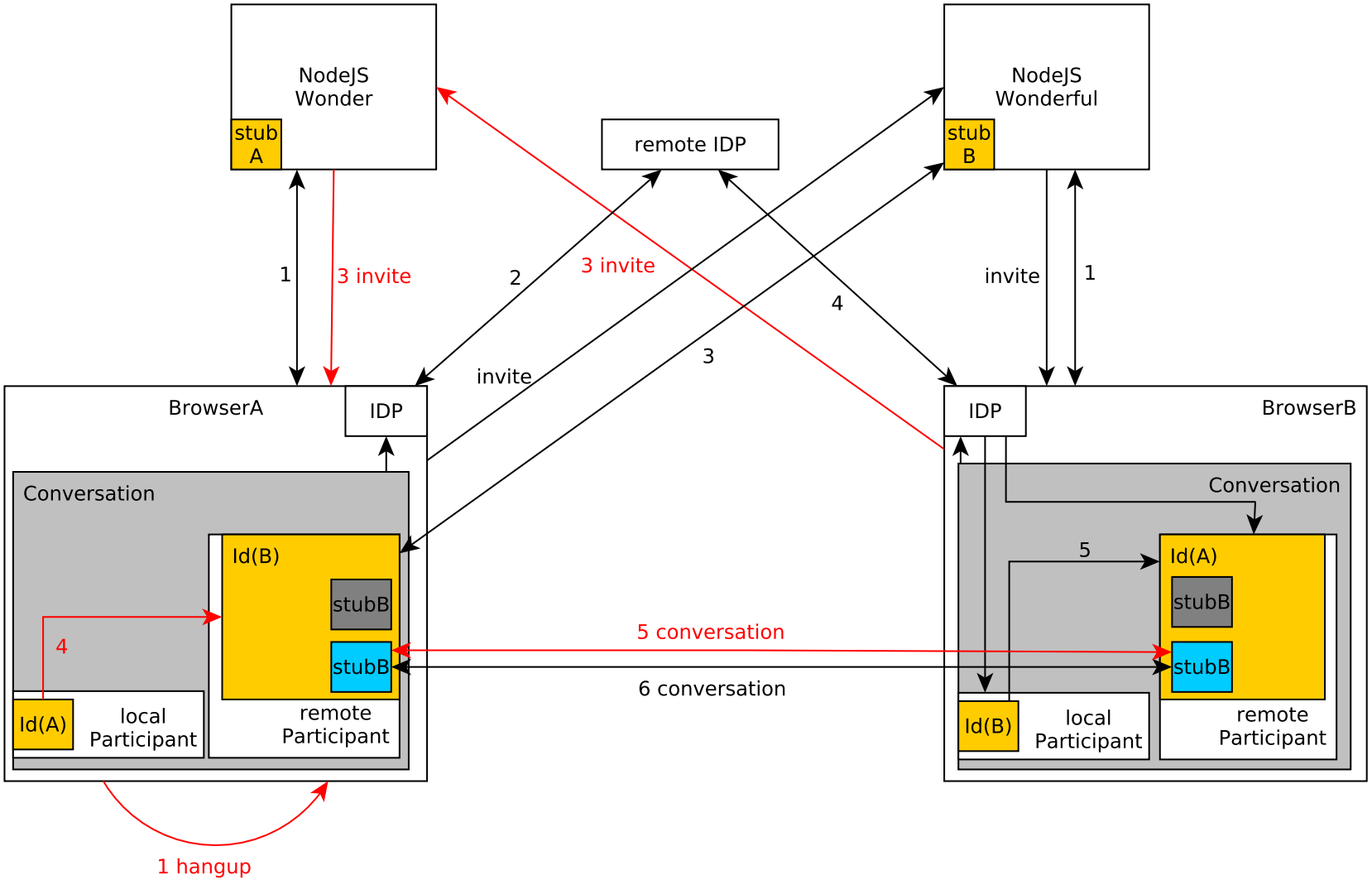
Teillösung 2



Teillösung 2



Teillösung 2



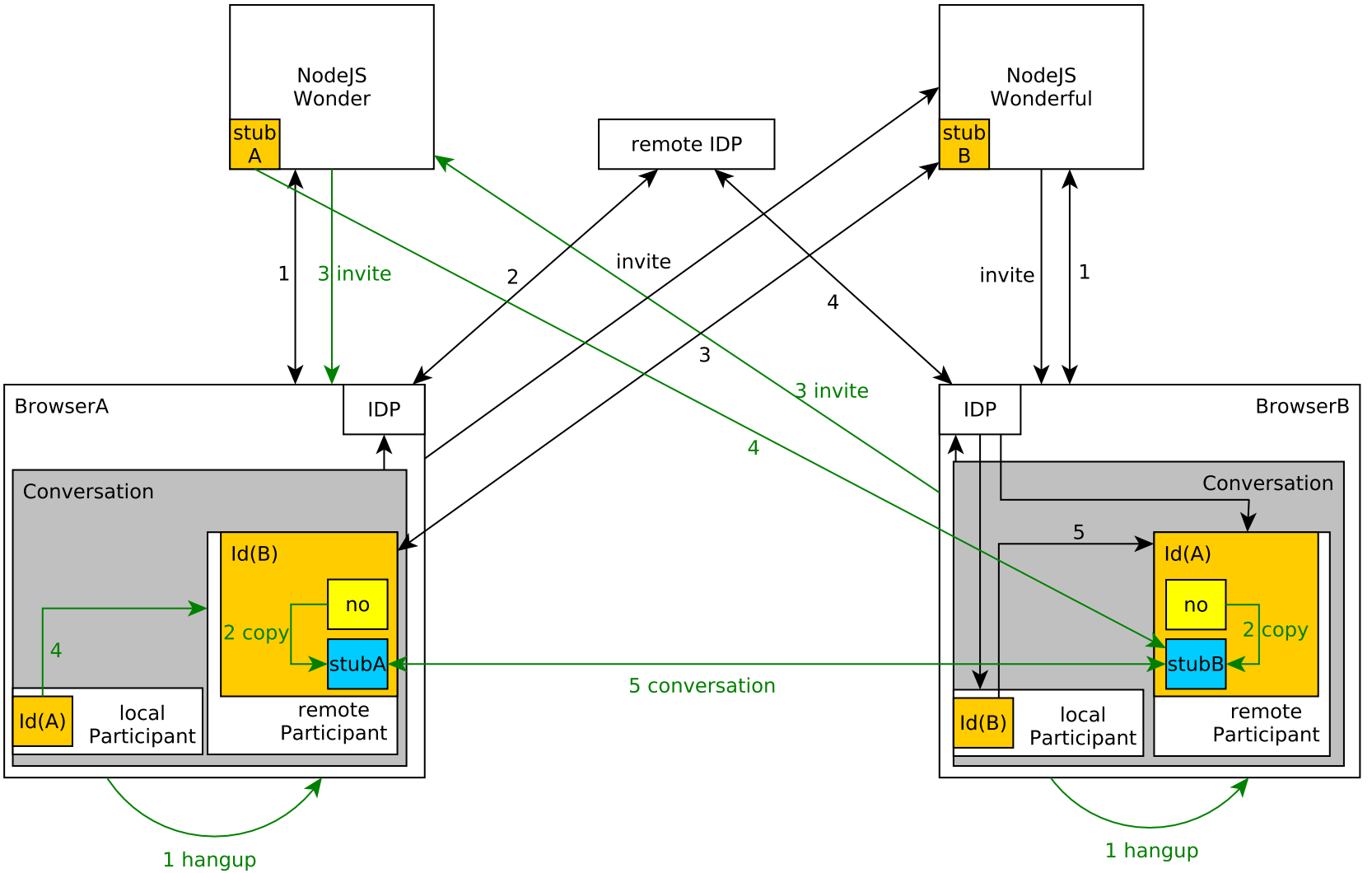
Teillösung 2

- Rückruf jetzt möglich, jedoch nicht
  - Wenn der Anrufer aufhängt
- 3/4 Problemen behoben
- Nicht funktionstüchtig
  - A hangup & B call A
- Funktionstüchtig
  - A hangup & A call B
  - B hangup & A call B
  - B hangup & B call A

Teillösung 2 Wonder-Verhalten

- Wenn ich Anrufinitiierer bin, ändere nichts beim hangup
- Wenn ich Angerufener bin, setze no-stub (`.impl=NULL`) beim hangup
- Wenn ich Angerufener bin, nimm den eigenen `messagingStub`
- Wenn kein `messagingStub` beim Anrufen vorhanden (NULL), tue nichts
- Wenn `messagingStub.impl` vorhanden, nutze diese für einen Anruf
- Wenn `messagingStub.impl` nicht vorhanden, downloade diese für einen Anruf

Lösung


NodeJS
Wonder
stub
A
remote IDP
NodeJS
Wonderful
stub
B
1
3 invite
2
invite
4
3
invite
1
3 invite
4
BrowserA
IDP
IDP
BrowserB
Conversation
Conversation
Id(B)
no
2 copy
stubA
5
Id(A)
no
2 copy
stubB
4
5 conversation
Id(A)
local
Participant
remote
Participant
Id(B)
local
Participant
remote
Participant
1 hangup
1 hangup

Lösung

- `Conversation.prototype.bye` und `.close` nicht geeignet
- Beide Teilnehmer müssen jedoch die `Participant.prototype.leave`-Methode ausführen
- Recall jetzt in jeder Kombination möglich

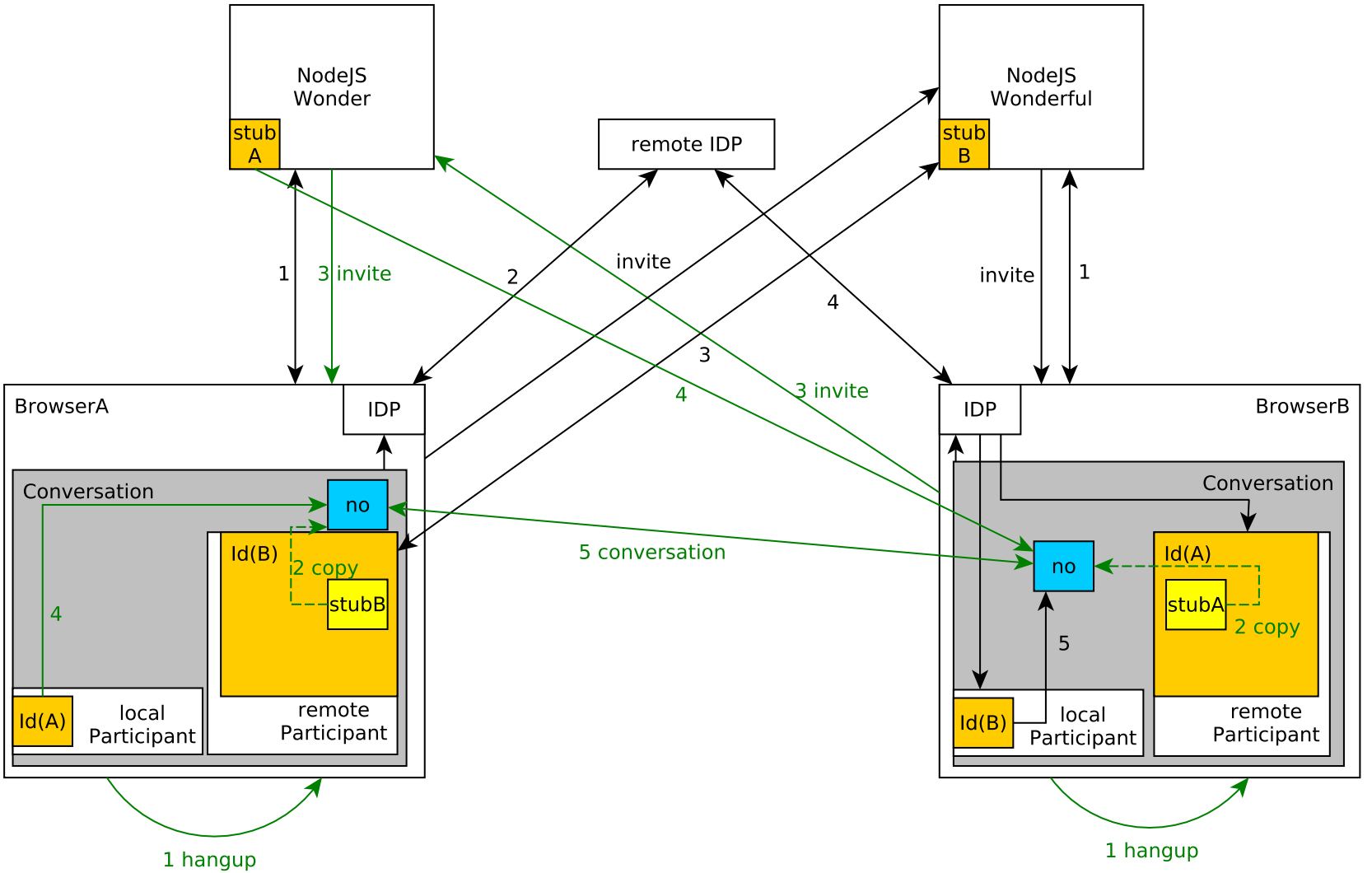
Participant.prototype.leave Zeile 1057

```
1  if(this.identity.rtcIdentity == this.me.identity.rtcIdentity){
2      this.RTCPeerConnection.getLocalStreams().forEach(function(element, index,
           array){array[index].stop();});
3      if(sendMessage==true)
4          this.sendMessage("",MessageType.BYE,"","",function(){},function(){});
5  } else {
6      if(sendMessage==true) this.sendMessage("",MessageType.BYE,"","",function
           (){},function(){});
7      this.dataBroker.removeDataChannel(this.identity);
8      if(this.RTCPeerConnection.signalingState && this.RTCPeerConnection.
           signalingState != "closed") this.RTCPeerConnection.close();
9      this.identity.messagingStub=this.identity.originalStub;
10 }
```

Zusatzaufgabe: `messagingStub` an conversation binden

- Vorteile
  - Ein `messagingStub` pro conversation
  - Zentrale Verwaltung der `messagingStubs`
  - Problem von NULL-stubs und `messagingStub`-Tausch im remote participant ausgelagert
  - Eventuell schnellere Ausführung, je nach Anzahl der participants
- Aufwand
  - Mehrere Stellen im Quelltext gleichzeitig ändern
  - Kein inkrementelles Vorankommen
  - Geeignetes Feld in conversation suchen wie z.B. `hosting`, was in verschiedenen Zuständen immer vorhanden ist
  - Zugriff aus allen relevanten Funktionen sicherstellen

Zusatzaufgabe: `messagingStub` an conversation binden



Zusammenfassung

- Recall ist in jeder Kombination möglich
- Keine Abhängigkeit vom conversation-owner
- Zwischenspeicherung des `messagingStubs` der remote identity wünschenswert
- Realisierung des `messagingStubs` über das hosting-Feld in einer conversation wäre vorteilhaft

Vielen Dank
für eure Aufmerksamkeit.

Sind Fragen offen?