About SIP

An introduction to Transactions & Dialogs

Jonas Borjesson

@borjessonjonas

NOTE

- This version has been adapted to be viewed without transitions.
- Go to **aboutsip.com** to download the original version.
- Also be sure to check out **vimeo.com/aboutsip** for any recorded presentations.
- Follow **@borjessonjonas** to receive updates.

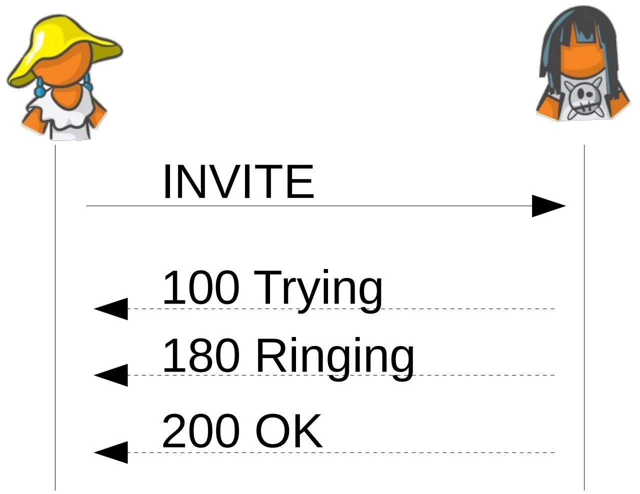
SIP Messages

recap

- SIP – Request/Response model
  - UAC sends the request, UAS responds
- Two types of responses
  - Provisional (1xx)
  - Final (2xx - 6xx)

Transactions

- SIP is a transactional protocol.
- Every request & response goes within a transaction.
- Transactions are independent of each other.
- SIP transaction:
  - 1 request
  - 0..* provisional responses
  - 0..* final responses



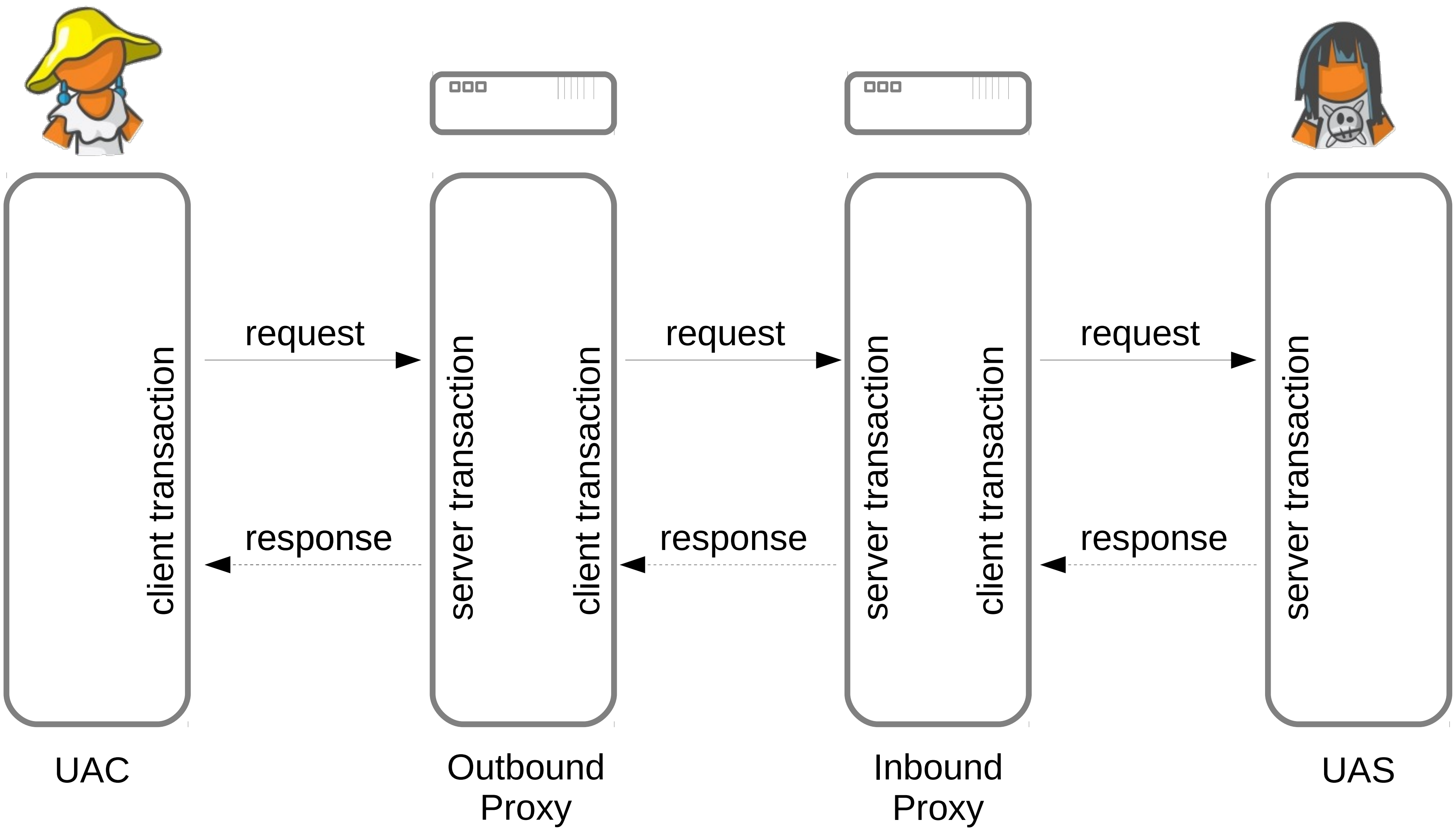
Client Transactions

- Responsible for:
  - Receiving request from the TU it and...
  - Deliver requests reliably to the server transaction
  - Processing responses:
    - Filter out retransmissions
    - Filter out disallowed responses
    - Pass response to the TU
  - Generating ACK for non 2xx final responses to INVITE transactions

Server Transactions

- Responsible for:
  - Receive requests and pass them up the TU
  - Filter out any retransmissions
  - Accepts responses from the TU and sends them.
  - Absorbing the ACK request for non 2xx final responses on invite transactions

Transaction Relationships



@borjessonjonas

Transaction Identifier

- Each transaction is uniquely identified by:
  - the branch-id on the Via-header plus
  - the Cseq header
- 3261 branch-id starts with “z9hG4bK”

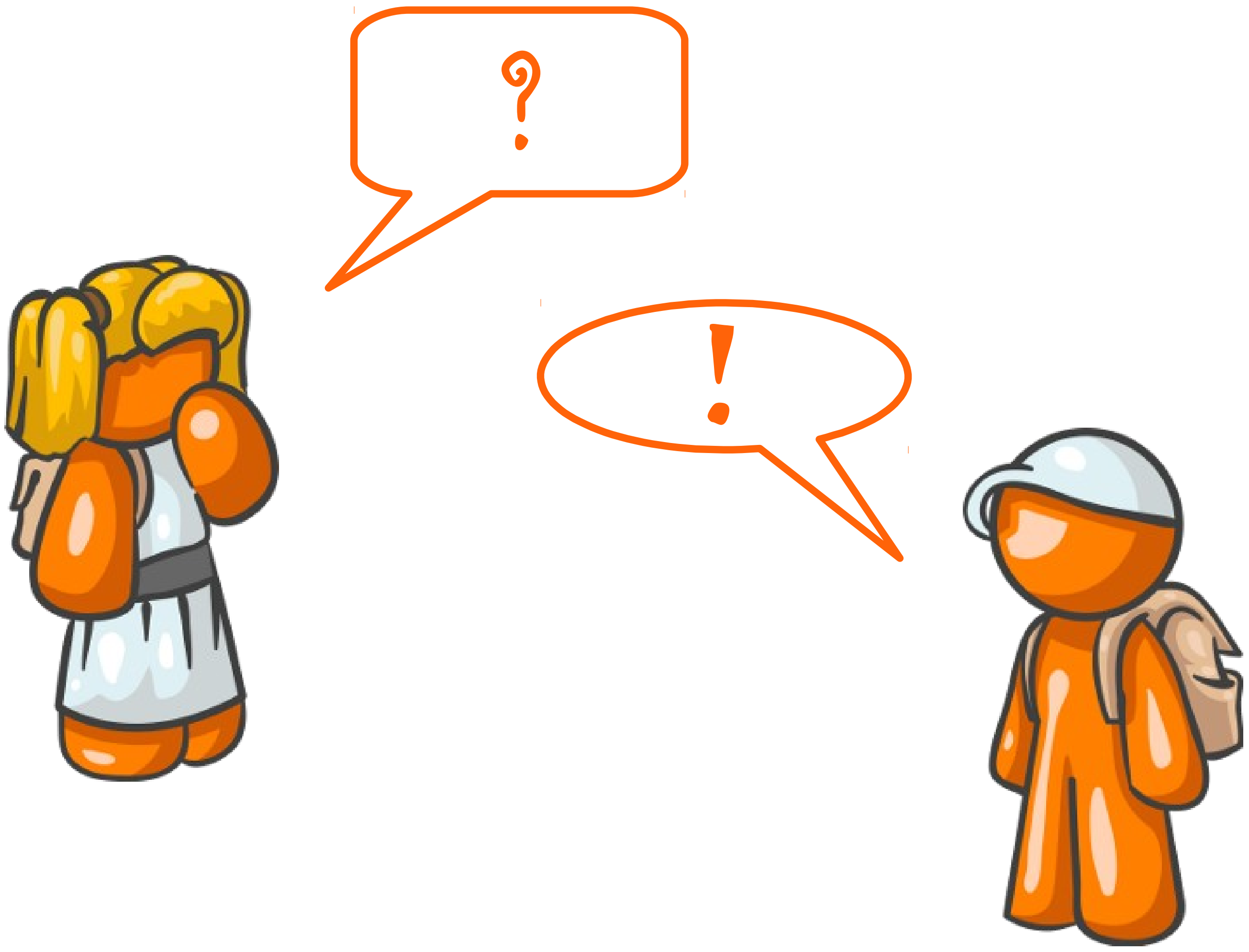
```
   INVITE sip:bob@aboutsip.com SIP/2.0
3  Via: SIP/2.0/UDP <ip_3>;branch=z9hG4bK-lkjsalkfjoijlkjlkj
2  Via: SIP/2.0/UDP <ip_2>;branch=z9hG4bK-8jijlk-asfk-iji0kj
1  Via: SIP/2.0/TCP <ip_1>;branch=z9hG4bK-llkowe-lkjko39d
   CSeq: 1 INVITE
   ...
```

Transaction Lifecycle

- Slightly different depending on:
  - Server or client transaction
  - Invite or non-invite transaction
  - Reliable vs. non-reliable transport (e.g. udp vs tcp)
- To summarize them all*
  - Starts when request is sent or received
  - Final responses takes it to completed state
  - Timer fires and moves it to terminated state.

*You can't really group them together like this. Please view this information as a very generic summary of the life cycle of a SIP transaction. In reality, it is much more complicated.

Dialogs



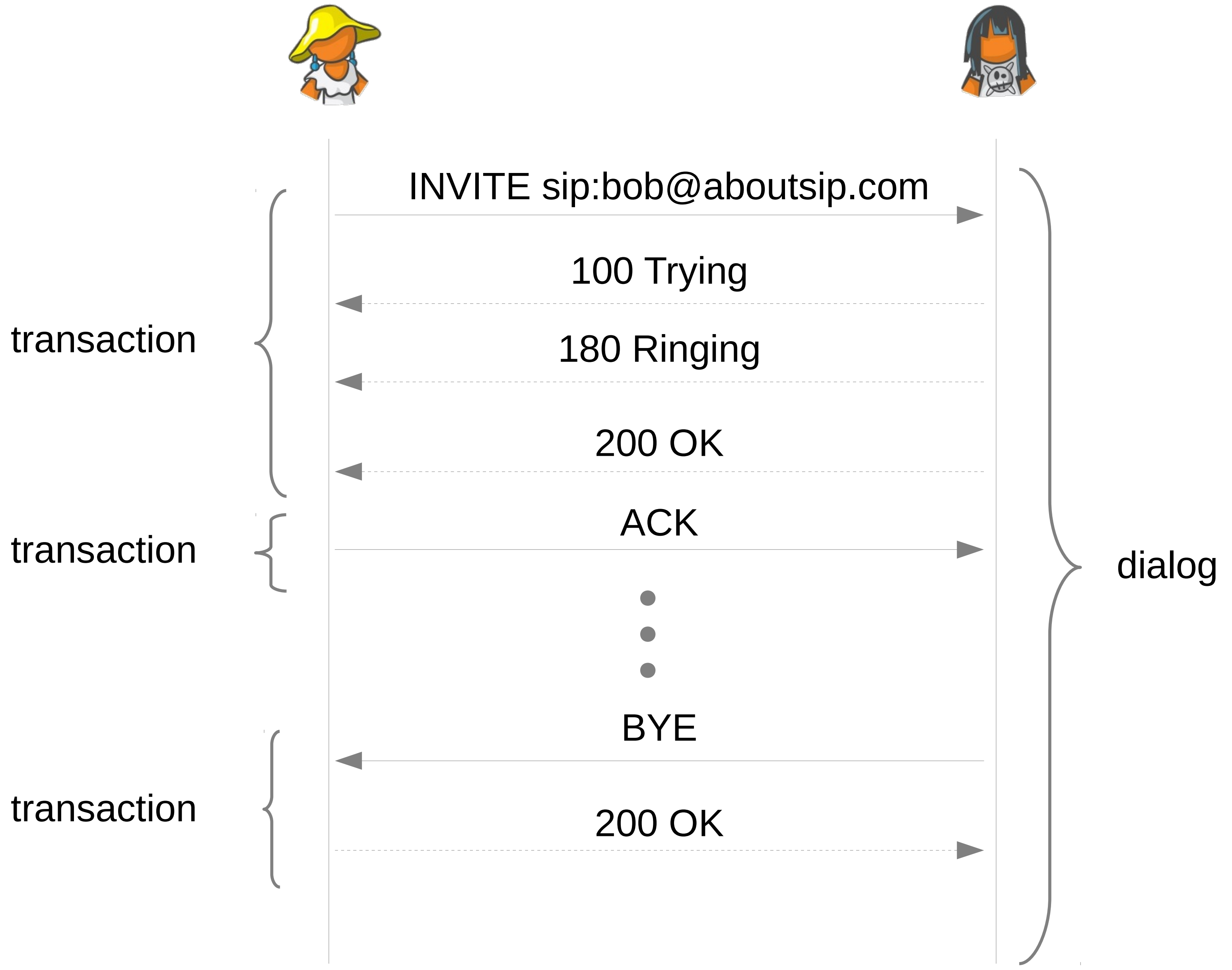
Dialogs

- Dialogs are extremely important within SIP
- A dialog:
  - represent a p2p relation between two SIP endpoints.
  - exists for some time
  - contains important routing information
  - facilitates proper sequencing of messages
  - consists of a sequence of transactions

Dialog establishing methods

- Not all methods establish a dialog!
- The ones that do are:
  - INVITE (RFC 3261)
    - For establishing a dialog
  - SUBSCRIBE (RFC 3265)
    - Creates a subscription. Used e.g. in presence scenarios
  - REFER (RFC 3515)
    - Also a subscription but only for the refer event-package. Used e.g. for call transfer

INVITE Scenario



@borjessonjonas

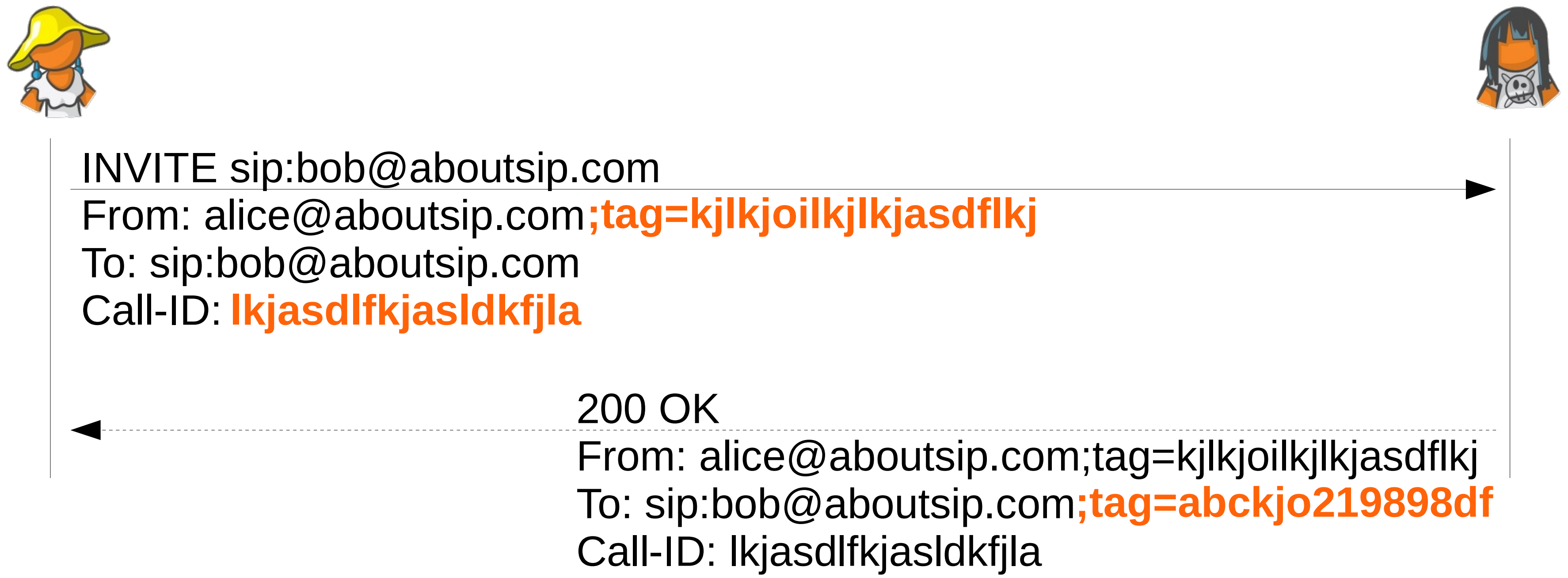
Dialog Id

- A dialog is uniquely identified by:
  - The Call-ID header
  - The remote-tag
  - The local-tag
- My remote tag is your local tag and vice versa
- Therefore, the dialog id is different for both ends (a very common thing SIP stacks mess up)

```
SIP/2.0 200 OK
To: <sip:bob@aboutsip.com>;tag=e103a059-a9ca-4bc2-96d1-779636810bfe
From: <sip:alice@aboutsip.com>;tag=f7389c89-ee9f-4802-af3e-636ce53883cb
Call-ID: 9d8eccee-02f2-4952-a1bf-01fe1bae45d6
CSeq: 2 INVITE
Contact: <sip:127.0.0.1:1557;transport=TCP>
```

Establishing a dialog

- UAC
  - creates the initial request.
  - Fills in “half” of the dialog-id.
- UAS
  - establishes the dialog through a 2xx final response
  - fills in the other “half” of the dialog-id.



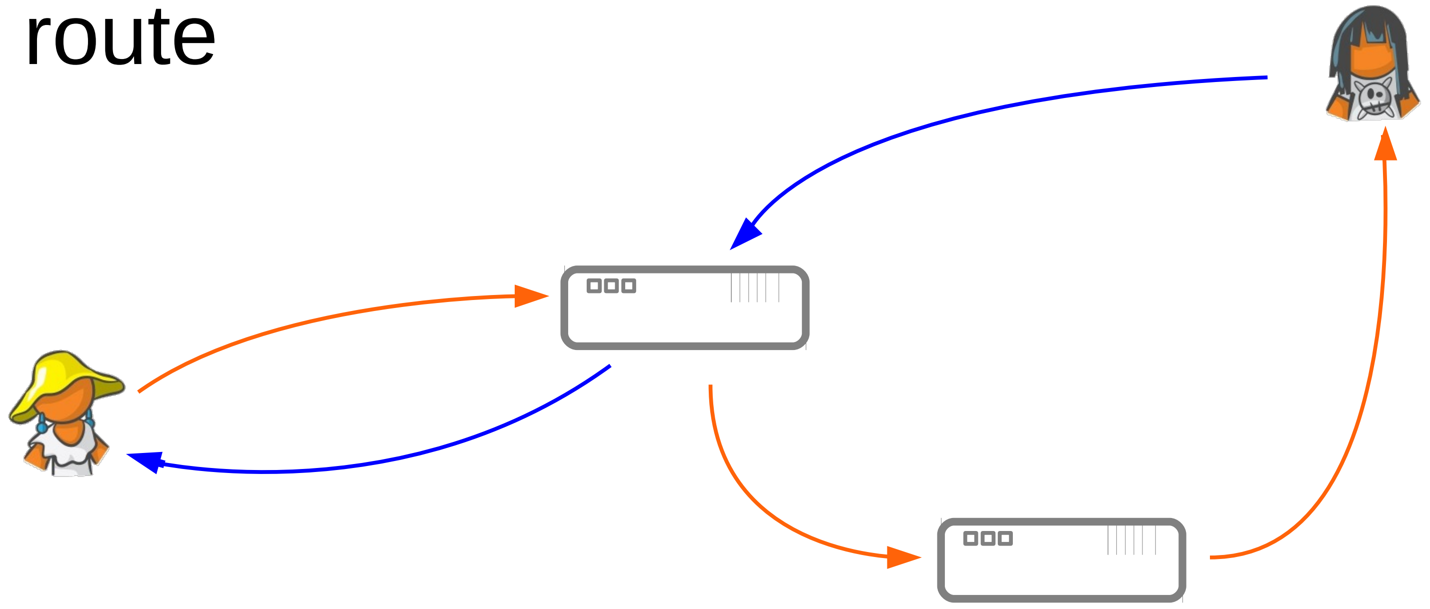
```
INVITE sip:bob@aboutsip.com
From: alice@aboutsip.com;tag=kjlkjoilkjlkjasdflkj
To: sip:bob@aboutsip.com
Call-ID: lkjasdlfkjasldkfjla
```

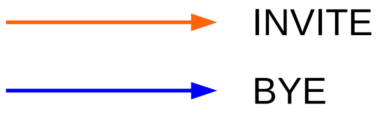
```
200 OK
From: alice@aboutsip.com;tag=kjlkjoilkjlkjasdflkj
To: sip:bob@aboutsip.com;tag=abckjo219898df
Call-ID: lkjasdlfkjasldkfjla
```

@borjessonjonas

Subsequent Requests

- Requests that goes within an established dialog are called subsequent requests
- Subsequent requests follow the established route

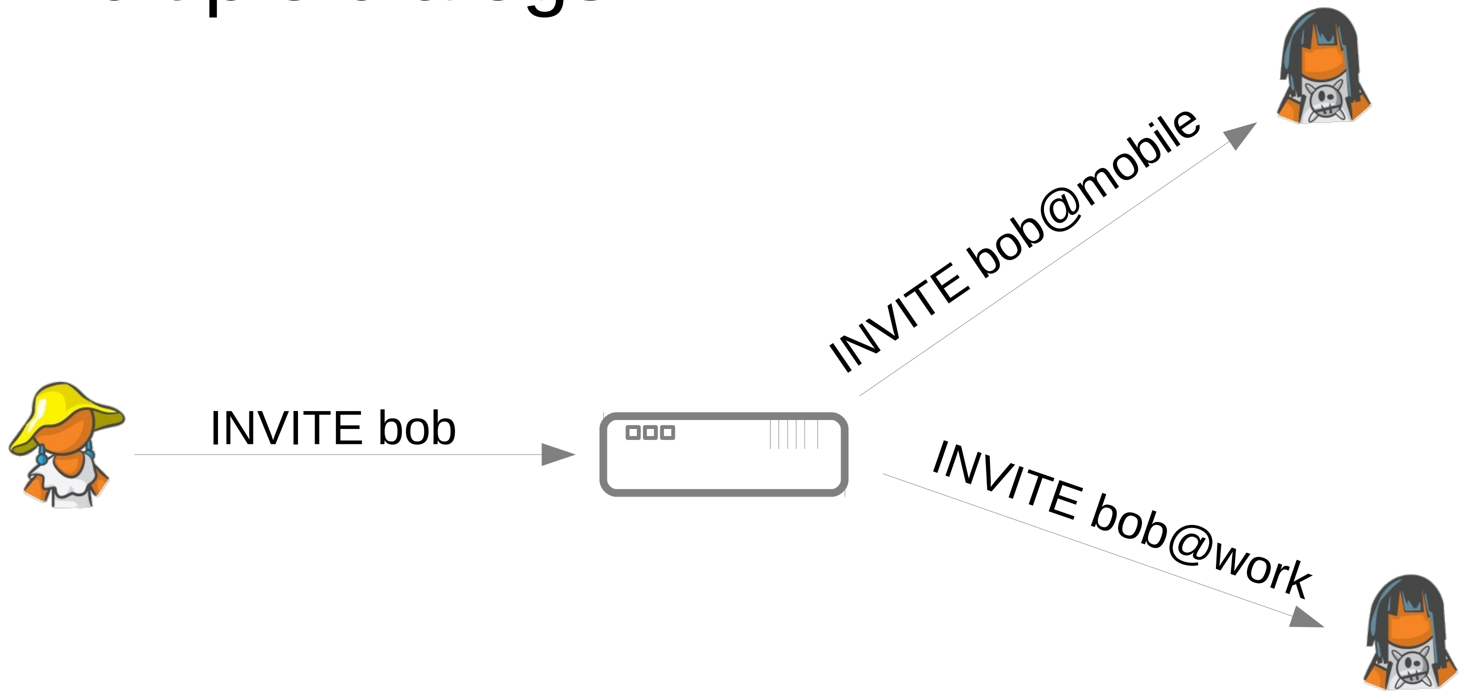




@borjessonjonas

To Complicate things...

- There is such a thing as early-dialogs.
- One INVITE request can be forked and create multiple dialogs.



@borjessonjonas

Tearing down a dialog

- Depends on the method
  - BYE request for INVITE dialogs
  - Un-subscribe request for SUBSCRIBE dialogs
  - REFER dialogs typically die when the reference to which the subscription is referring to goes away.
- Of course, lots of corner cases that can lead to the destruction of a dialog...

Summary

Transactions

- Consists of 1 request and 0..* responses
- Two types of transactions:
  - Invite transactions
  - Non-Invite transactions
- The branch-id on the Via uniquely identifies the transaction (plus the Cseq)

Dialogs

- Consists of a sequence of transactions
- Not all methods establish dialogs.
- Represent a p2p relation between two SIP endpoints.
- Contains routing information.
- The dialog-id is different for the two parties.

More presentations and material
at aboutsip.com

Thanks!