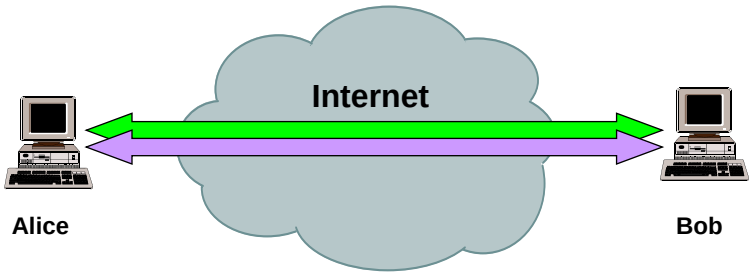
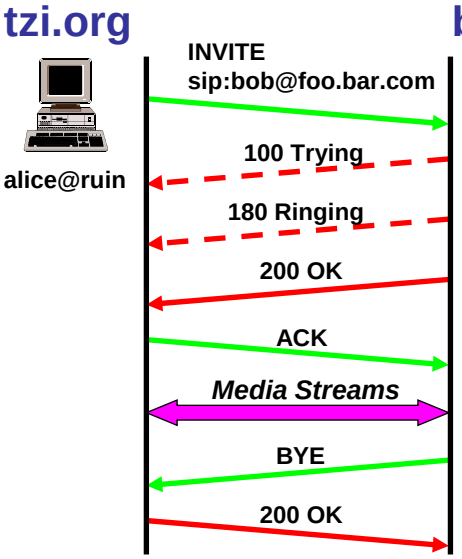
Application Scenario 1: Direct Call UA–UA



Call signaling

Media streams

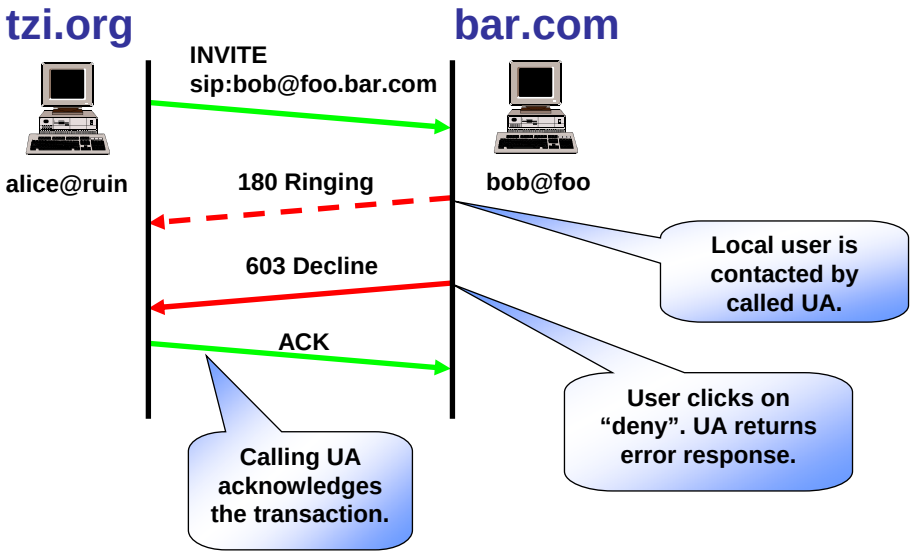
Direct Call



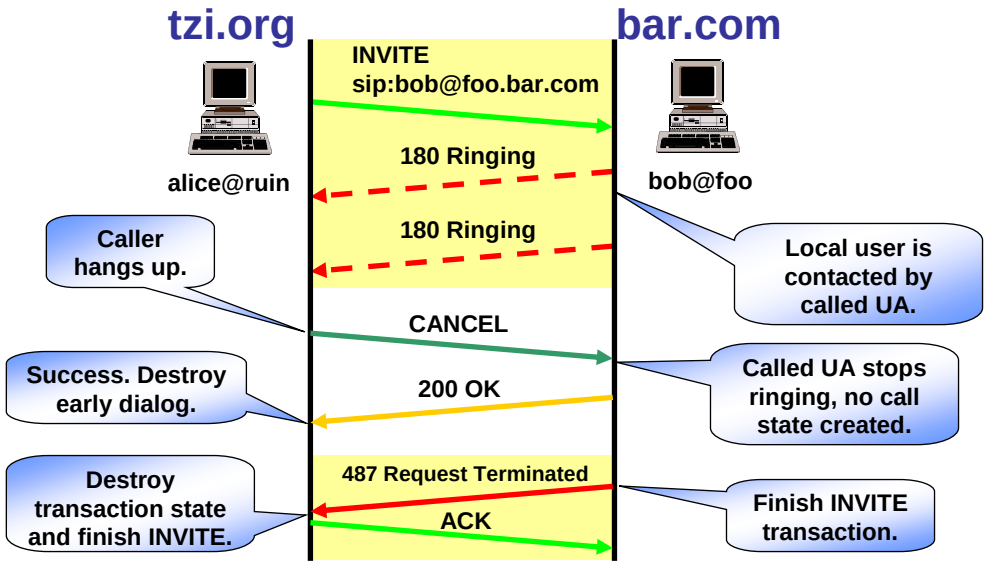
Note:
Three-way handshake is performed only for INVITE requests.

- Caller knows callee's hostname or address
- Called UA reports status changes
- After Bob accepted the call, OK is signaled
- Calling UA acknowledges, call is established
- Media data are exchanged (e. g. RTP)
- Call is terminated by one participant

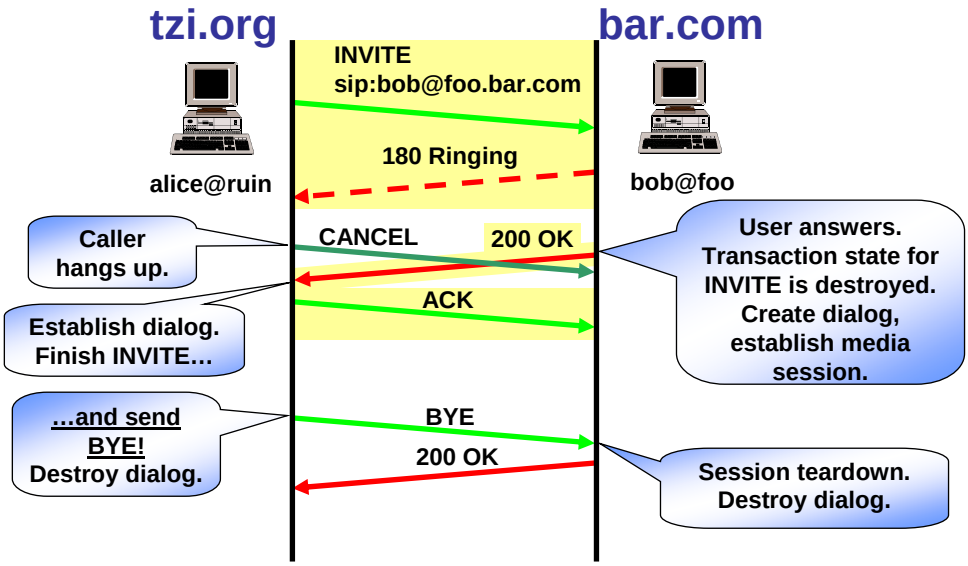
Callee Declines Call



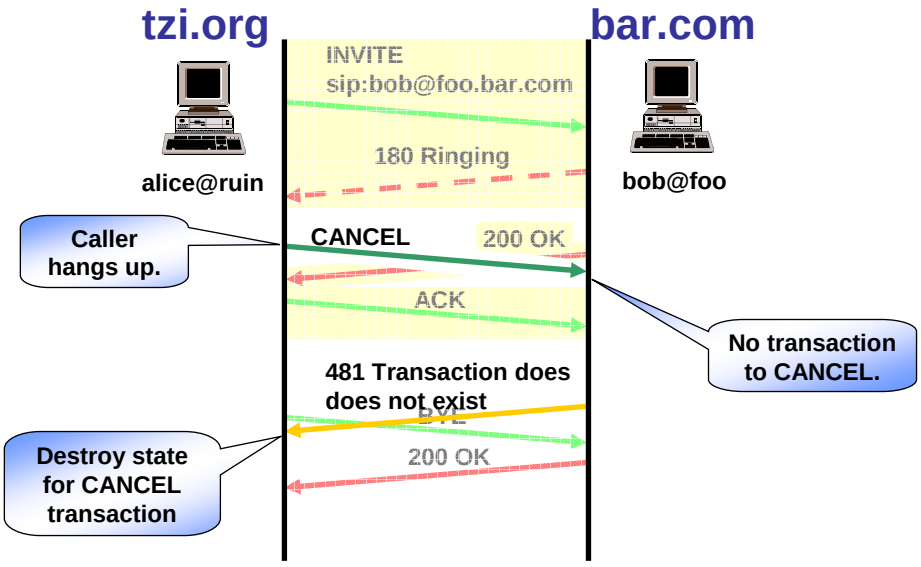
Caller Gives Up



Caller Gives Up While Call Established



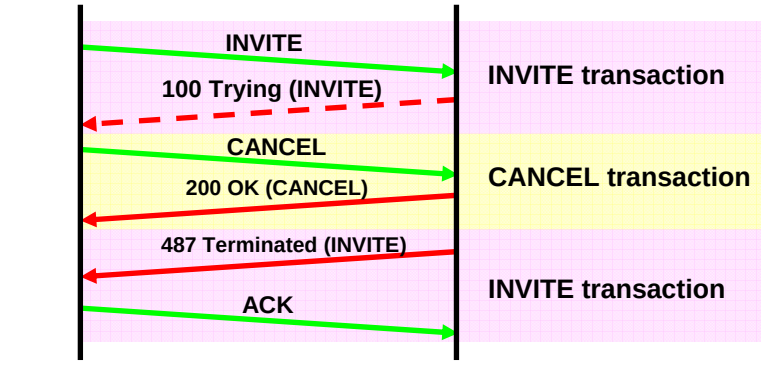
Caller Gives Up While Call Established



Atomic Transaction Processing

Transactions:

- Created by initial (i.e. not retransmitted) requests
- Terminated by final response to initial request
- Identified by Method, Request-URI, To:, From:, Call-ID:, CSeq: and topmost Via:



Transactions always complete independently!

How to Find The Callee?

- Direct calls require knowledge of callee's address
- SIP provides abstract naming scheme:

sip:user@domain

→ Define mapping from SIP URI to real locations:

- Explicit registration:
  UA registers user's name and current location
- Location service:
  Use other protocols to find potentially correct addresses

- Caller sends INVITE to any SIP server knowing about the callee's location
- Receiving server may either redirect, refuse or proxy

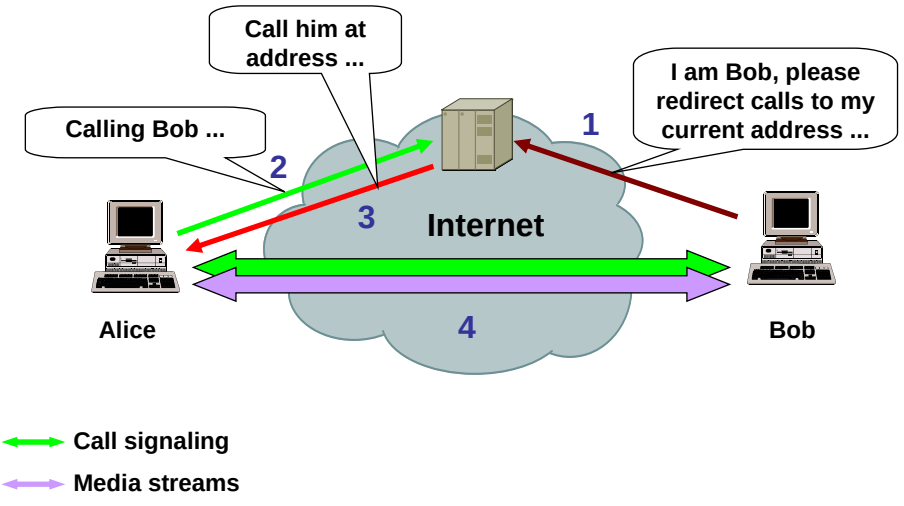
Finding the Next Hop

- UAC may use a (manually) configured outbound proxy
  - Outbound proxy may also have be learned upon registration
- If request URI contains IP address and port, message can be sent directly

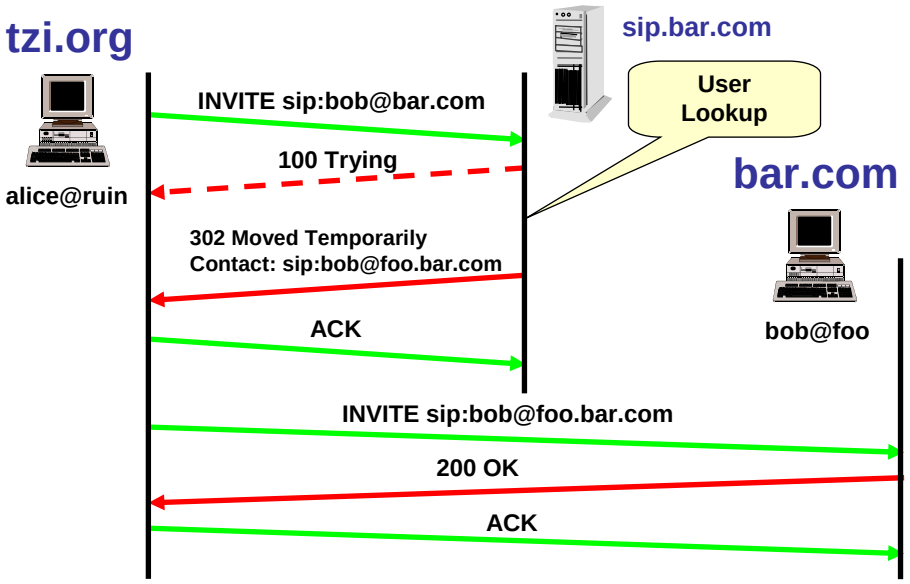
UDP/TCP 5060
TLS 5061

- Otherwise, determine next hop SIP server via DNS
  - Use NAPTR RR (SIP+D2U/D2T/D2S, SIPS+D2T/D2S) to obtain SRV records
  - Query for SRV RR: _sip._udp, _sip._tcp, _sips._tcp for all supported transport protocols
  - If entries found, try as specified in RFC 2782
    - If no entries found, use UDP for sip: URIs and TCP for sips: URIs
- Query A or AAAA records for IP address
  - For specified domain name
  - (Deprecated: For specified *sip.domain*)

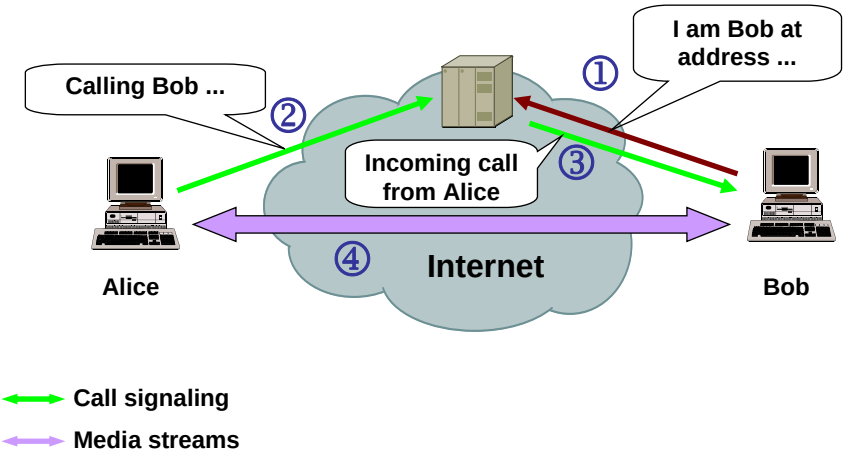
Application Scenario 2: Redirected Call



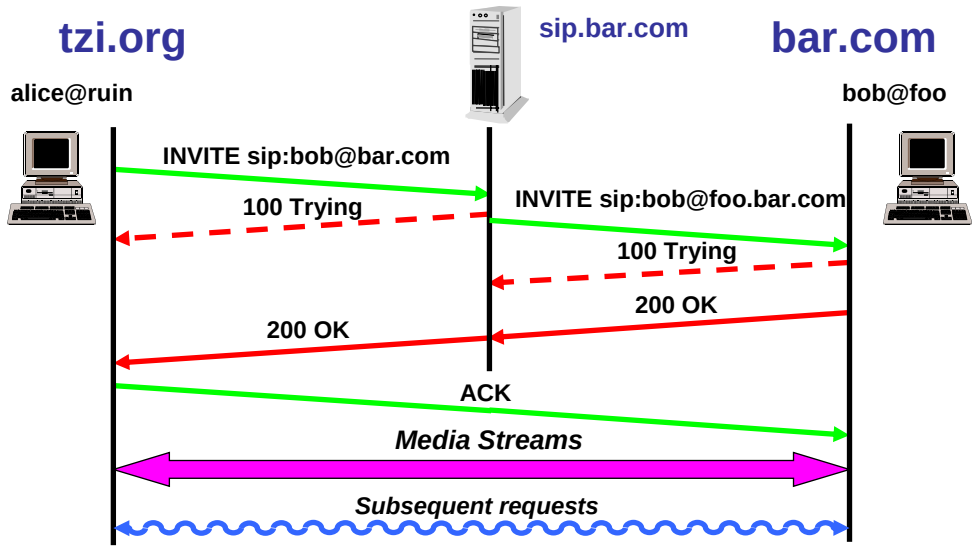
Redirected Call



Application Scenario 3: Proxied Call

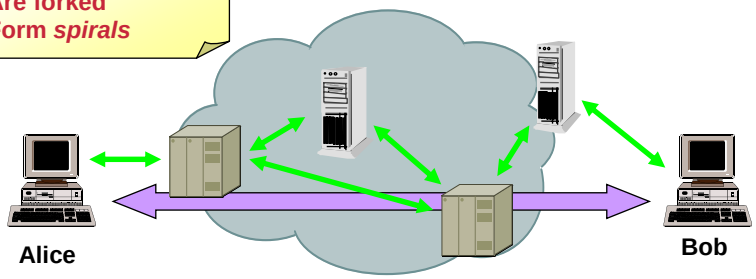


Proxied Call



Application Scenario 4: Proxied Call (Real World)

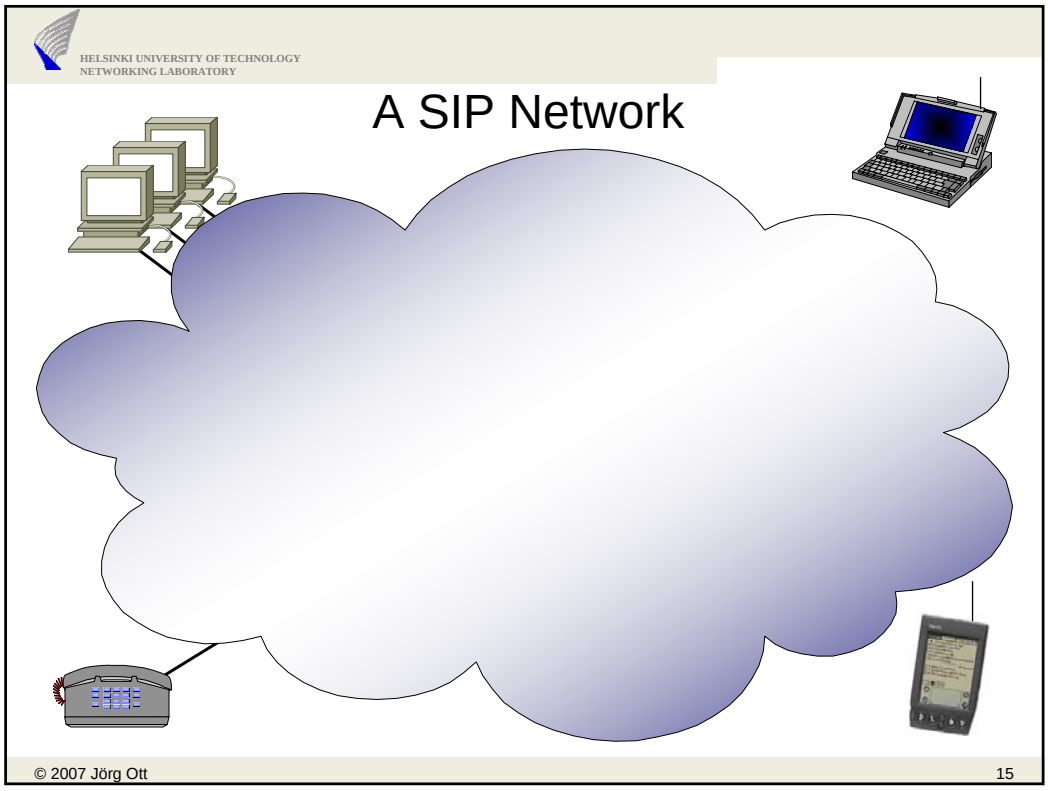
Requests typically
- Take different paths
- Are forked
- Form *spirals*

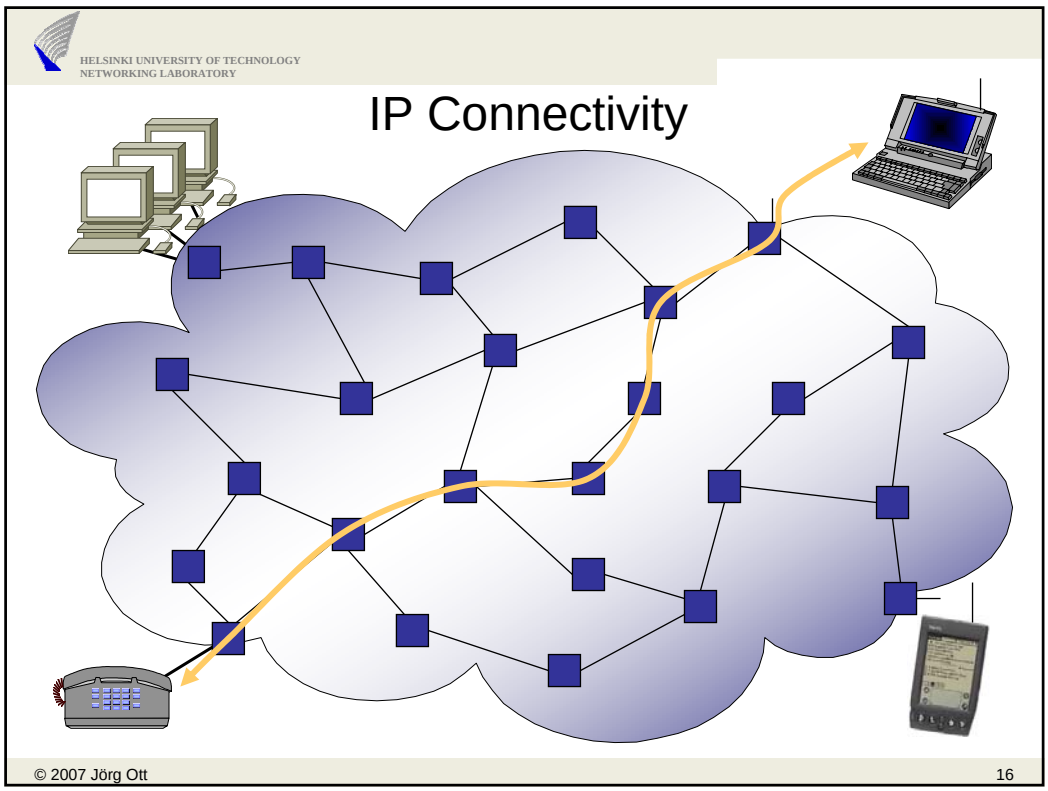


Responses always Take the reverse path of the originating request

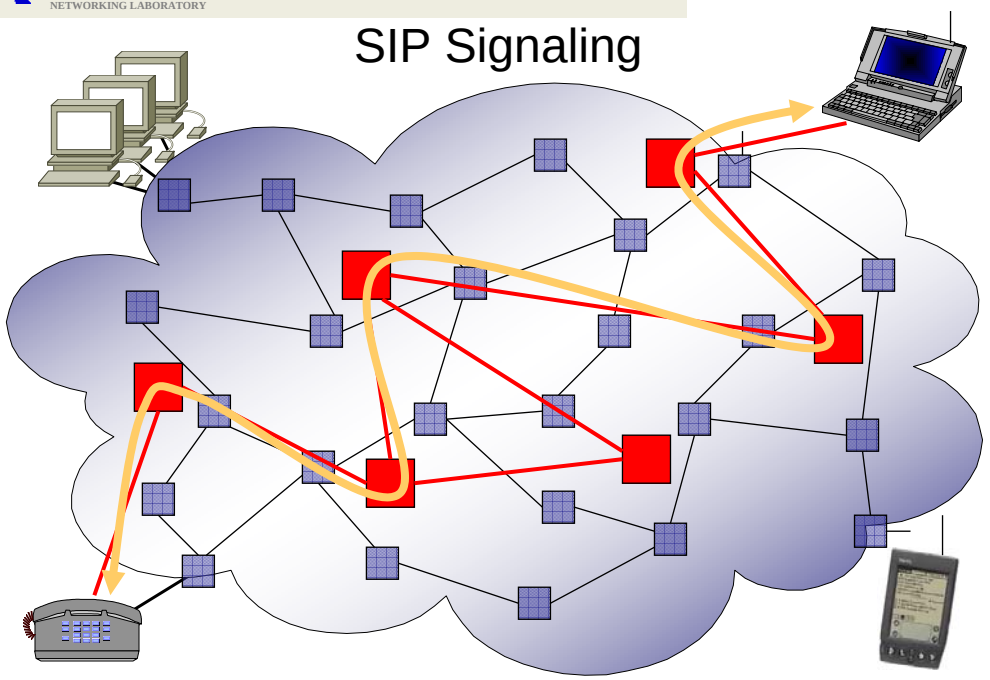
Call signaling

Media streams

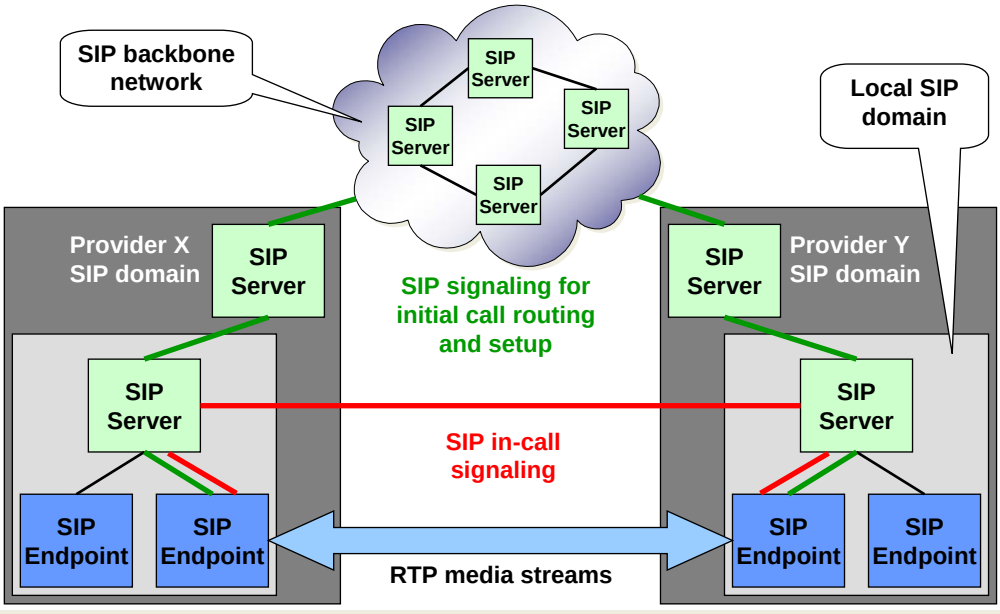




SIP Signaling



Global SIP Architecture



SIP (Proxy) Server Functionality

- Stateless vs. stateful
  - Stateless: efficient and scalable call routing (backbone)
  - Stateful: service provisioning, firewall control, ...

Some roles for proxies

- Outbound proxy
  - Perform address resolution and call routing for endpoints
  - Pre-configured for endpoint (manually, DHCP, ...)
- Backbone proxy
  - Essentially call routing functionality
- Access proxy
  - User authentication and authorization, accounting
  - Hide network internals (topology, devices, users, etc.)
- Local IP telephony server (IP PBX)
- Service creation in general...

SIP (Proxy) Server Functionality

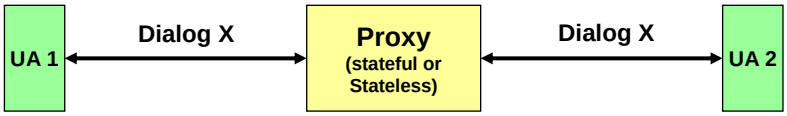
- Stateless vs. stateful
  - Stateless: efficient and scalable call routing (backbone)
  - Stateful: service provisioning, firewall control, ...

Some roles for proxies

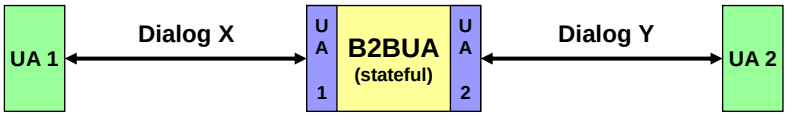
- Outbound proxy
  - Perform address resolution a
  - Pre-configured for e
- Backbone p
  - Essenti ality
- Access pr
  - User authe ication and authorization, accounting
  - Hide network internals (topology, devices, users, etc.)
- Local IP telephony server (IP PBX)
- Service creation in general...

More elaborate functions provided by Back-to-Back User Agents (B2BUAs)

Proxy vs. B2BUA



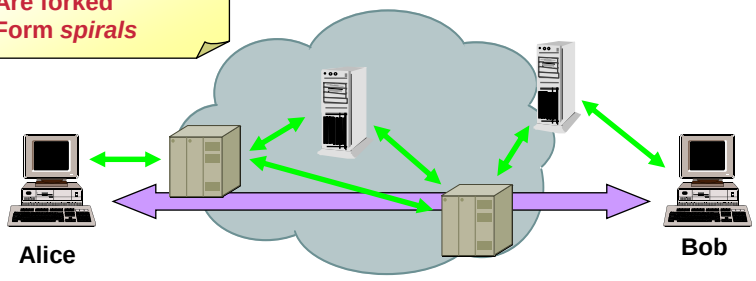
Proxies only route and forward requests on behalf of UAs, they do not get active by themselves. They may generate responses and react to responses as part of processing a request.



B2BUAs terminate dialogs. They may create the illusion of an end-to-end dialog by coupling two dialogs and forwarding messages transparently between UAs. But they are a party to both dialogs.

Application Scenario 4: Proxied Call (Real World)

Requests typically
- **Take different paths**
- **Are forked**
- **Form *spirals***

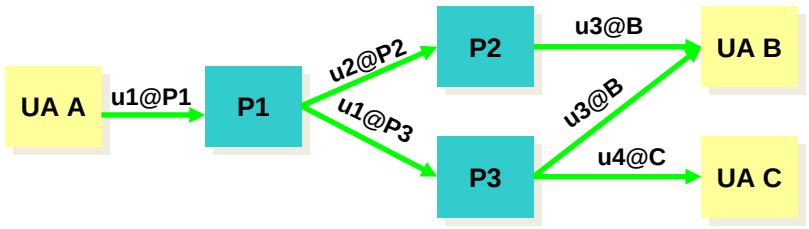


Responses always Take the reverse path of the originating request

Call signaling

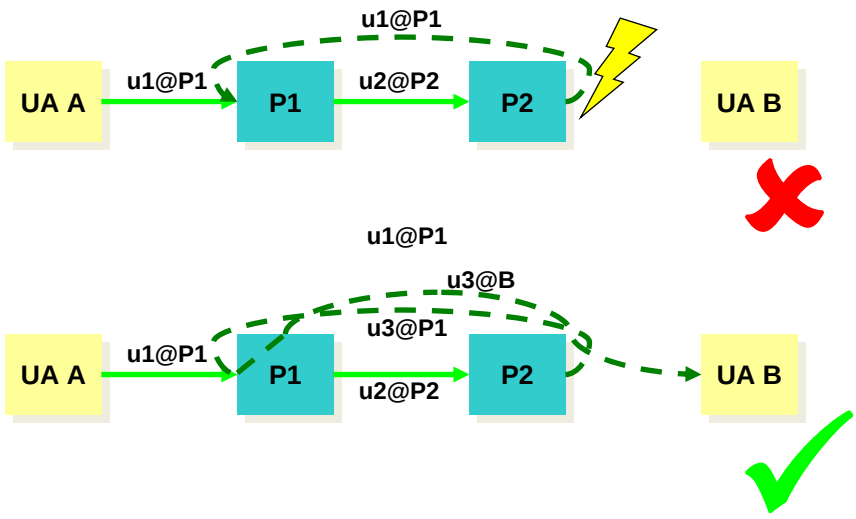
Media streams

Request Routing



- Every server determines next hop
- Several destinations may be tried in parallel
  - Mark outgoing messages with branch tag
  - Use z9hG4bK + unique id to indicate unique branch tag
- Multiple requests can arrive at a single UAS
  - UAS tags To: header to identify user presence
  - RFC3261: From: tags mandatory for request merging at UAS

Loops vs. Spirals

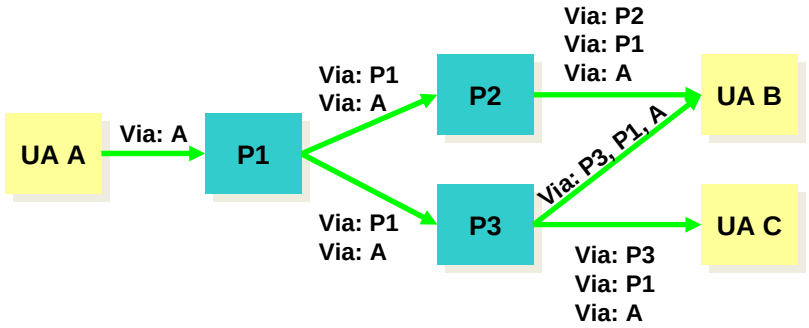


Response Path

- Proxies may collect state for pending transactions
  - TCP connections
  - Accounting information
  - Parallel search

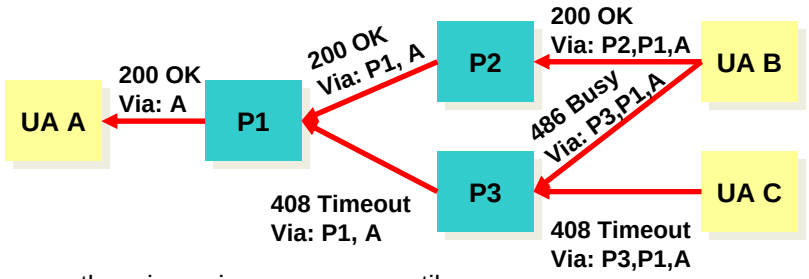
  → need response to clean up
- Insert Via: header when forwarding request
- Send response along reverse path of originating request
- Subsequent requests may take different path!
  - Unless record routing is used

Creation of Via-path



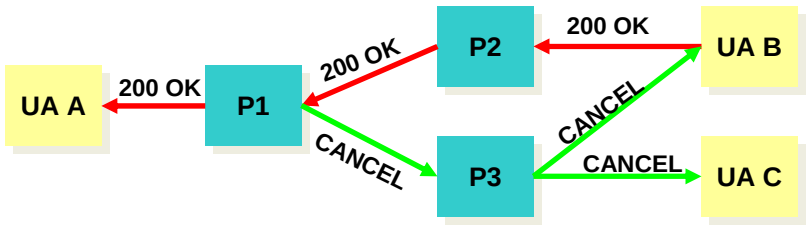
- Send outgoing responses to first Via-entry
- UAS drops responses with different first Via entry
- Add branch-tag to distinguish different search paths

Response Merging



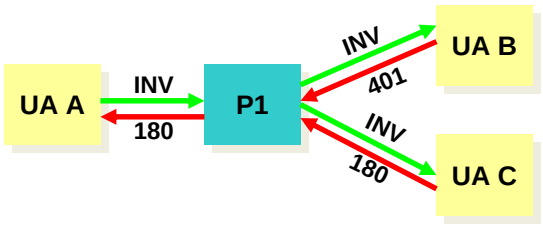
- Proxy gathers incoming responses until
  - No requests are pending, or
  - Request timers have expired
  - A user-initiated final response arrives (6xx or 2xx)
- "Best" response is returned to caller, others may be discarded or aggregated ("repairable errors")
- OK-responses are returned whenever received
  - INVITE requests may get multiple 200 OK responses, others just one

Cancelling requests



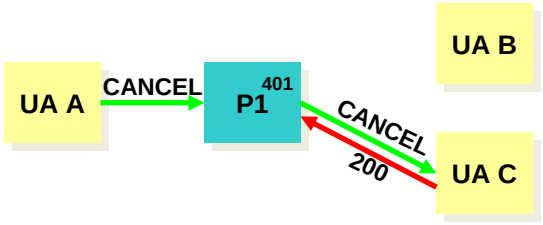
- Cancel a previous request
- Does not affect complete transactions
  - UAS sends 487 Transaction does not exist
  - Otherwise no impact on UAS state
- Used by forking proxies to prune search tree if OK response arrived (see P1)

Heterogeneous Error Response Forking Problem (HERFP) 1



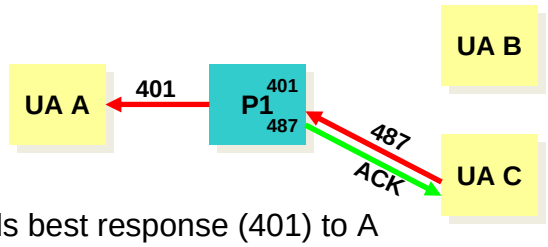
- P1 forks INVITE request
  - UA B stays silent and solicits “401 Unauthorized”
  - UA C begins Ringing (180)
- Provisional response 180 is forwarded upstream
  - UAC sees ringing indication
- No handling for 401 response possible at this time

Heterogeneous Error Response Forking Problem (HERFP) 2



- No answer from C hence A hangs up → CANCEL

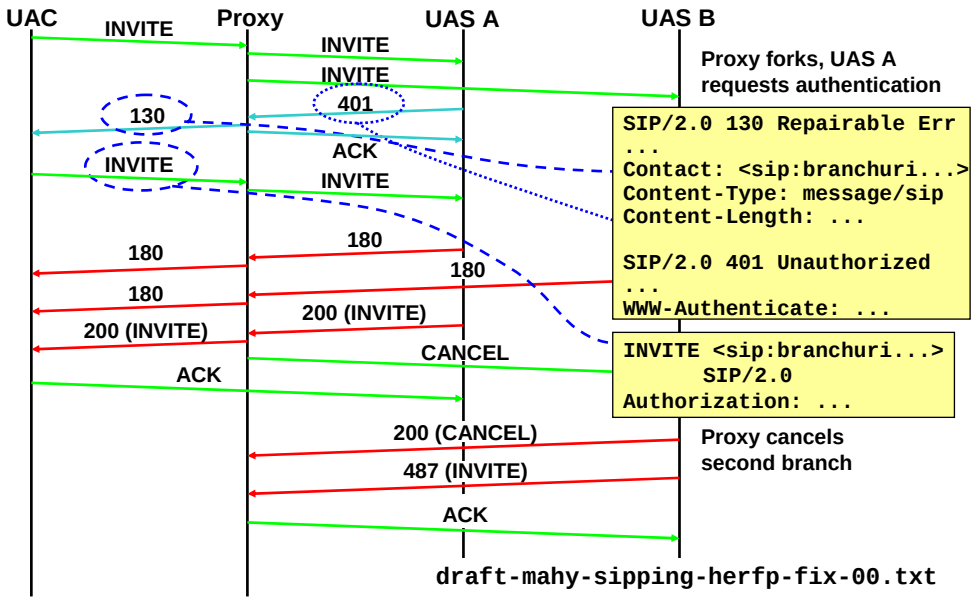
Heterogeneous Error Response Forking Problem (HERFP) 3



- P1 forwards best response (401) to A
  - Resubmit INVITE with appropriate credentials for B
- Drawbacks:
  - C will ring again
  - large call-setup delay
  - Forked non-INVITE must be idempotent for sequential search

→ Heterogeneous error responses cause problems

Solutions to HERFP? (1)



Solutions to HERFP? (2)

- Use 130 Repairable Errors response
  - UAC issues a DECLINE method if the error cannot be fixed

  `draft-mahy-sipping-herfp-fix-00.txt`
- Use "FIX" request back to UAC (instead of 130)

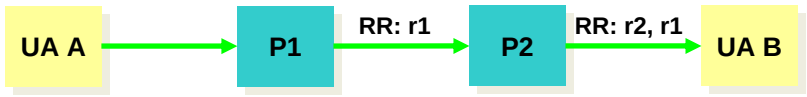
  `draft-jbemmel-sipping-herfp-solution-00.txt`
- Alternative: avoid complexity
  - Problems comes from forking -> avoid forking
  - If multiple targets are found, return 300 class response
    - Redirect the client to multiple targets
    - Protect targets by means of internal mapping

  `draft-roach-sip-herfp-avoidance-00.txt`

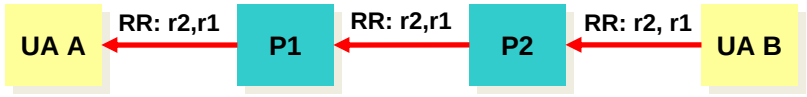
Record-Route

- Proxies may depend on seeing every request of established dialog
  - Update state information for accounting, call distribution, …
  - Firewalls and NATs
- Record-Route and Route headers for request routing
  - Prepend globally reachable request URI with proxy's address to Record-Route entries
  - Receiving UAS copies contents into response
  - Route header may be initialized with pre-existing route set
- Subsequent requests for this call will contain *source route* created from initial Record-Route header
  - Route only created during dialog establishment (e.g., INVITE)
  - Unchanged throughout dialog duration (only target URI may change)

Constructing the Route

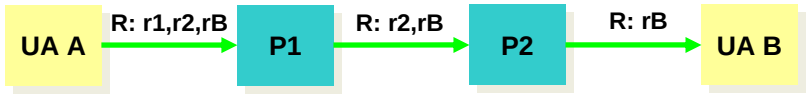


- UA A sends request for UA B to P1
- P1 and P2 add request URIs to Record-Route header
- UA B stores recorded route for later use



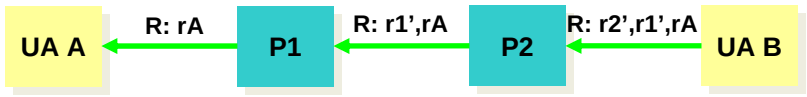
- UA B literally copies record route of request into response

Using a Recorded Route



- **UA A creates new request to B with *reversed* contents of Record-Route from received response in Route-header**
- **UA A appends B's contact from former response**
- **Every proxy forwards request according to Route**

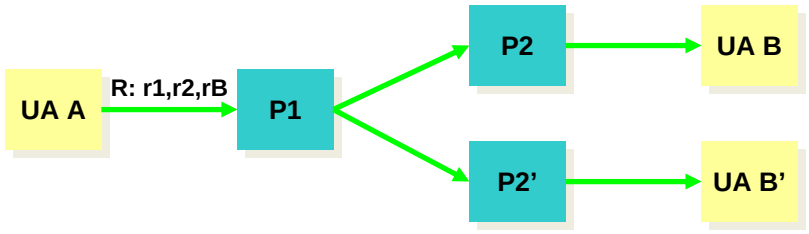
Significant change from RFC 2543



- **UA B does not reverse ordering**
- **Request URIs for proxies are changed to UA A's contact**

Pre-loaded Routes

- **Requests may contain pre-existing Route set**
  → default outbound proxy, CSCFs in 3GPP
  - **Configured manually or learned automatically**
  - **For transactions within a dialog, pre-pended to Record-Route set**
- **Loose source routing:**
  Forwarding proxy may ignore topmost Route entry
- **Route entry rewriting:**
  In a response Proxy may rewrite its own Route URI
- **Deal with forked routes (due to DNS lookups):**

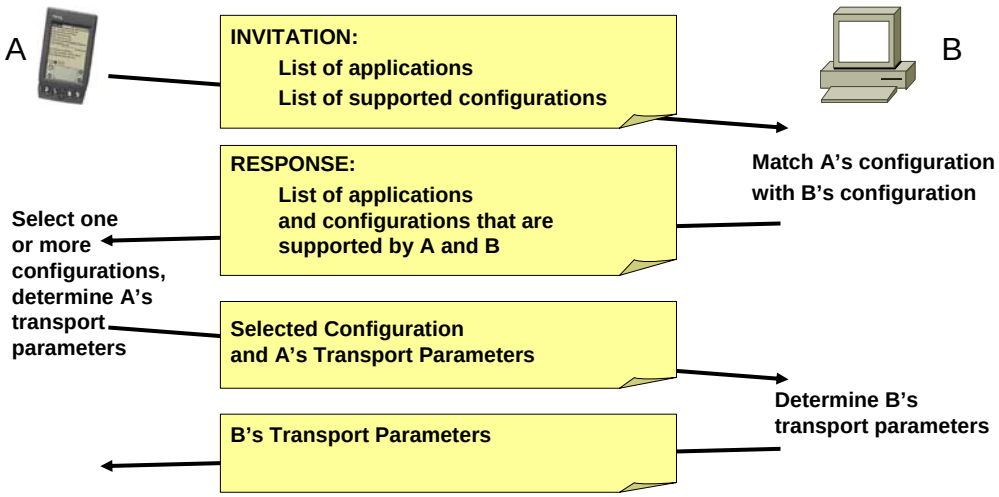


Media Session Setup Between Endpoints

Session Initiation

- Distribute conference configuration
  - Applications
    - Media types, media format parameters
  - Transport Parameters
    - IP addresses, transport protocols, protocol parameters
- Negotiate Parameters!
  - Heterogeneous end systems
    - Different hardware and software capabilities
  - User preferences
- SDP provides syntax mechanisms to express parameters
  - Procedural model for initiation required

Recap: Invitation Conceptual Model



SIP Capability Negotiation with Offer/Answer (RFC 3264)

- SDP: Session Description Protocol, RFC 2327
  - Used to rudimentarily describe media sessions of a call

SDP Example

Length of Time represented by Media in a single Packet

(In SIP: address where originator wants to receive data)

```
v=0
o=llynch 3117798688 3117798739 IN IP4 128.223.214.23
s=UO Presents KWAX Classical Radio
i=University of Oregon sponsored classical radio station KWAX-FM
u=http://darkwing.uoregon.edu/~uocomm/
e=UO Multicasters multicast@lists.uoregon.edu
p=Lucy Lynch (University of Oregon) (541) 346-1774
t=0 0
a=tool:sdr v2.4a6
a=type:test
m=audio 30554 RTP/AVP 0
c=IN IP4 224.2.246.13/127
a=ptime:40
```

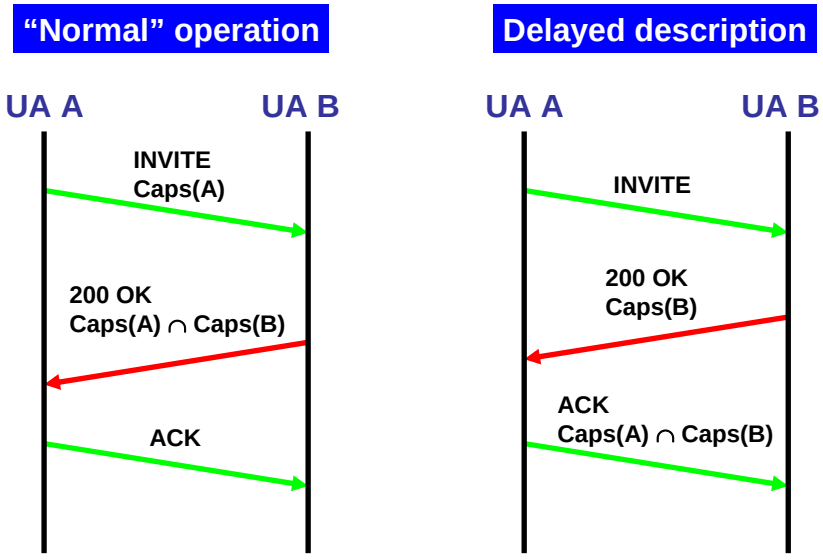
Session Level

Media Level

SIP Capability Negotiation with Offer/Answer (RFC 3264)

- SDP: Session Description Protocol, RFC 2327
  - Used to rudimentarily describe media sessions of a call
- Caller includes SDP capability description in INVITE
  - Time information may be set to “t=0 0” or omitted
  - For RTP/AVT, use of *rtpmap* mappings is encouraged
- For each media stream (*m*-part of SDP message), callee returns own configuration in response
  - Indicate destination address in *c*=-field
  - Indicate port and selected media parameters in *m*=-field
  - Set port to zero to suppress media streams
- UA may return user’s capability set in 200 OK response when receiving an OPTIONS request

Media Negotiation During Call Setup

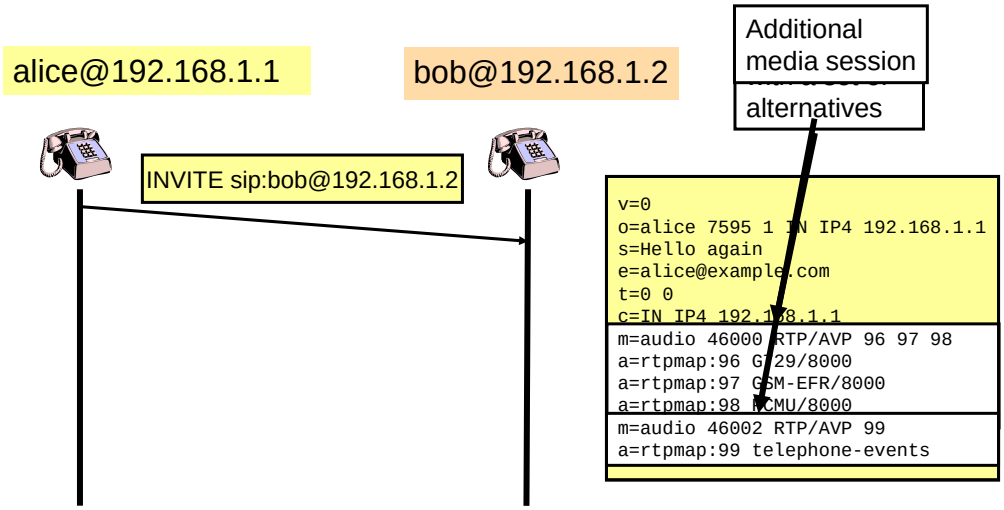


Changing Session Parameters

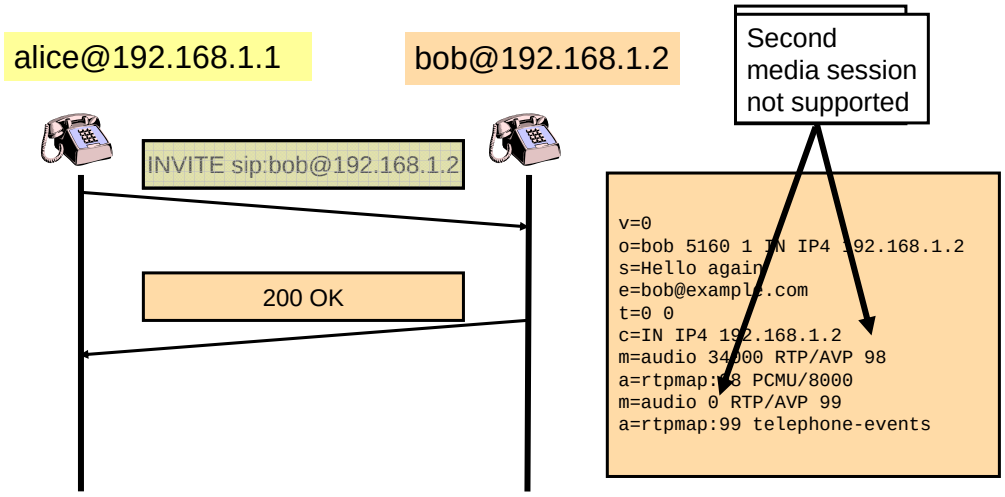
- Either party of a call may send a re-INVITE that contains a new session description
  - Use other connection address, port
  - Add/remove codecs
  - Append new media streams at the message's end
- Recipient re-aligns session description with current values
  - Change media parameters
  - Delete media streams (port has zero-value)
  - Add new streams

→Gateways may use re-INVITE when session parameters are unknown during call setup

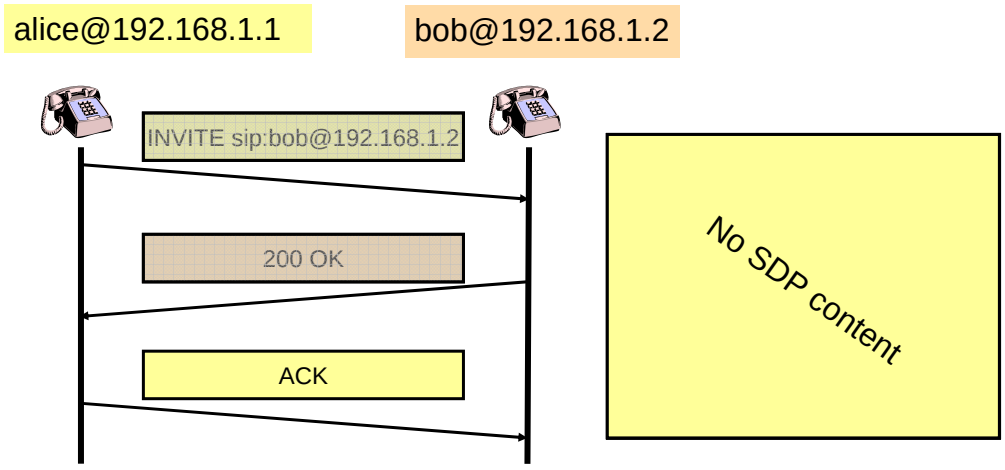
SDP in SIP Applications



SDP in SIP Applications



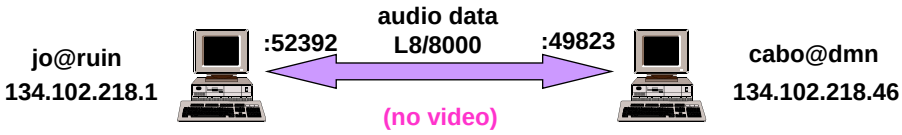
SDP in SIP Applications



Example SDP Alignment

v=0 o=jo 7849 2873246 IN IP4 ruin.inf… s=SIP call t=0 0 c=IN IP4 134.102.218.1 m=audio 52392 RTP/AVP 98 99 a=rtpmap:98 L8/8000 a=rtpmap:99 L16/8000 m=video 59485 RTP/AVP 31 a=rtpmap:31 H261/90000	v=0 o=cabo 82347 283498 IN IP4 dmn.inf… s=SIP call t=0 0 c=IN IP4 134.102.218.46 m=audio 49823 RTP/AVP 98 a=rtpmap:98 L8/8000 m=video 0 RTP/AVP 31

Resulting configuration:



Media and Codec Selection

- Offer can contain multiple media streams
  - Answerer selects those suitable for itself
  - Indicates choice by setting port=0 for unsupported media
  - Media line matching/identification by order of appearance
- Offer can provide multiple codecs for a media stream
  - Ordered by preference
  - Offerer commits to support all codecs (one at a time)
  - Answerer should generate list of codecs for each stream, maintaining payload type mapping
  - New codecs may be added
- One of N codec selection
  - Offer multiple codecs, but cannot change dynamically
  - Offerer sends codec list “with reservation”
  - Answerer sends back subset
  - Offerer “locks” one codec for session
  - Implemented with `a=inactive` media level attribute…

Send/Receive Only

- Media streams may be unidirectional
  - Indicated by *a=sendonly*, *a=recvonly*
- Attributes are interpreted from sender's view
- sendonly
  - Recipient of SDP description should not send data
  - Connection address indicates where to send RTCP receiver reports
  - Multicast session: recipient sends to specified address
- recvonly
  - Sender lists supported codecs
  - Receiver chooses the subset he intends to use
  - Multicast session: recipient listens on specified address
- inactive
  - To pause a media stream (rather than deleting it)

Grouping of m= lines in SDP

- Observation:
  - Multiple m= lines in SDP have no relationship to each other
    - Independent media streams
    - usually different media types
- Problem:
  - Want to express synchronization relationship
    - Lip synchronization
  - Concept of "flows" that consist of several media streams
    - Streams encoded in several formats
    - May be streamed from different hosts/ports
    - Useful application in some IP telephony scenarios

Example for Lip Synchronization

Stream 1 and 2 should be synchronized.

```
v=0
o=Laura 289083124 289083124 IN IP4 one.example.com
t=0 0
c=IN IP4 224.2.17.12/127
a=group:LS 1 2
m=audio 30000 RTP/AVP 0
a=mid:1
m=video 30002 RTP/AVP 31
a=mid:2
m=audio 30004 RTP/AVP 0
i=This media stream contains the Spanish translation
a=mid:3
```


Simple Capability Declaration in SDP

- Observation:
  - Capability negotiation/declaration in SDP too limited
  - Session description describe both session parameters and capabilities without clear distinction
  - Simultaneous capability restrictions cannot be expressed
    - *"Supporting multiple codecs for one media type, but only one per session"*
- Simcap:
  add SDP attributes to explicitly express capabilities

Simcap Example

```
v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=
c=IN IP4 128.96.41.1
t=0 0
m=audio 3456 RTP/AVP 18 96
a=rtpmap:96 telephone-event
a=fmtp:96 0-15,32-35
a=sqn: 0
a=cdsc: 1 audio RTP/AVP 0 18 96
a=cpar: a=fmtp:96 0-16,32-35
a=cdsc: 4 image udptl t38
a=cdsc: 5 image tcp t38
```

Sender is willing to receive and send G.729 (18) and telephone-events.

Additionally, it declares the following capabilities:

- **PCMU-Audio (0)**
- **telephone-events (different events)**
- **Fax-Relay over UDP and TCP**

Simcap Example

```
v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=
c=IN IP4 128.96.41.1
t=0 0
m=audio 3456 RTP/AVP 18 96
a=rtpmap:96 telephone-event
a=fmtp:96 0-15,32-35
a=sqn: 0
a=cdsc: 1 audio RTP/AVP 0 18 96
a=cpar: a=fmtp:96 0-16,32-35
a=cdsc: 4 image udptl t38
a=cdsc: 5 image tcp t38
```

Semantics:

- **a=sqn: declares a sequence number**
- **a=cdsc: declare one or more capabilities**
- **a=cpar: additional parameters for a declaration**

SDP Extensions: There is more…

- Using alternative network addresses for media streams
  - To support IPv4 and IPv6
- Labeling media sections (beyond informational i= lines)
  - To enable referring to media streams from outside SDP
- Precondition signaling for media streams
  - Security, QoS, connectivity, connection establishment
- Establishing connection-oriented media streams (e.g. TCP)
  - Peers need to signal who shall initiate a connection and when to keep / release / replace a connection
- Key management (fixing k=)
  - End-to-end key negotiation
  - End-to-end key distribution (via a protected channel)
- Many simple attributes
  - RTCP bandwidth, RTCP port number, SSM source addresses, …
- And support for further media types
  - Multicast file distribution, application sharing, …

Changing Session Parameters

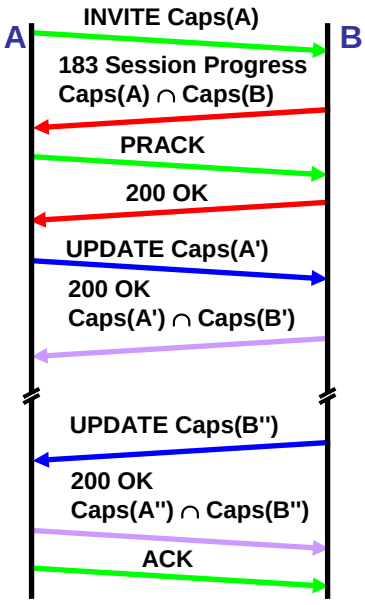
- Either party of a call may send a re-INVITE that contains a new session description
  - Use other connection address, port
  - Add/remove codecs
  - Append new media streams at the message's end
- Recipient re-aligns session description with current values
  - Change media parameters
  - Delete media streams (port has zero-value)
  - Add new streams

→ Gateways may use re-INVITE when session parameters are unknown during call setup

UPDATEs for Session State (RFC 3311)

- UPDATE as generic mechanism to modify session state
  - Dialog state vs. session state
  - Prequisite: Early dialog established
    - i.e. first non-100 response with To-tag received
- Allows updating session state before 200 OK
  - Negotiation and redirection of early media, putting media “on hold”
  - Requesting and informing about
    - resource reservation, security associations, connectivity
  - Implemented via session state changes without dialog state changes
- UPDATE and the Offer/Answer-Model
  - Multiple O/A cycles possible
  - No overlapping offers allowed
    - Race conditions signaled by 491 error response
  - O/A in 183, PRACK, etc. allowed

Use of UPDATE in Offer/Answer Model



- Offerer sends session description
- Answerer must match against local capabilities
  - Provides temporary answer in 183
    - E.g. for ringback tone
  - Acknowledge by PRACK for reliability
- UPDATE request during early dialog:
  - establish dialog state first (final or reliable provisional response)
  - No effect on dialog state
- No offer allowed as long as pending offers exist
- Any party may send offer