

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

北京邮电大学
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS




SIP协议分析 — SIP协议基础架构

李替林



STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

理解协议处理的基本原理与实现模型



- SIP协议消息处理规则及基本概念
 - Transaction
 - Dialog
 - Session
- SIP协议的实现模型
 - 协议栈结构
 - 协议处理模型
- 用户管理
- 网络组网与路由
- 媒体的协商与建立

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK


协议消息处理规则及基本概念



- Transaction
 - Two-Way Handshake
 - Transaction的定义
 - Three-Way Handshake
 - Transaction的分类
 - Transaction的处理规则
- Dialog
 - Dialog的定义
 - Dialog的处理规则
- Session
 - Session的定义

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK


Transaction



- Two-Way Handshake
- Transaction的定义
- Three-Way Handshake
- Transaction的分类
- Transaction的处理规则

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

Transaction




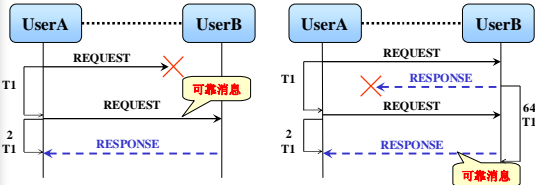
- Internet条件下，网络状况差异较大，如何保证信令可靠传输？
- 传输层保证
 - TCP数据可靠传输但并不适合信令协议
 - TCP连接建立时间长
 - TCP协议数据传递的滑动窗口机制
 - UDP协议不保证可靠传输
 - Sigtran是全新协议，应用有限
- 应用层保证
 - 消息确认
 - 超时重传

– 提供了可靠传输机制的消息：可靠消息
– 没有提供可靠传输机制的消息：不可靠消息

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

基于“请求-响应”机制的消息可靠传输

Two-Way Handshake

- 1、发出请求之后启动定时器T1，等待响应
- 2、T1超时，请求“可能”丢失，重发
- 3、重发消息的同时启动定时器2T1
- 4、T1、2T1、4T1、8T1
- 5、请求为可靠消息

- 1、发出响应之后启动定时器64T1，等待可能的重发
- 2、收到重发的请求，重发该请求的响应
- 3、64T1超时，结束
- 4、响应为可靠消息

T1取值一般500ms

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

事务 (Transaction)

- Transaction用于**应用层重发、定时和请求与响应的匹配**
- A SIP transaction occurs between a client and a server and comprises all messages from the **first request** sent from the client to the server up to a **final (non-1xx) response** sent from the server to the client
- SIP协议是基于事务 (Transaction) 的协议

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

Transaction的标识

```

INVITE sip:werner.heisenberg@munich.de SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max-Forwards: 70
To: Heisenberg <sip:werner.heisenberg@munich.de>
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:schroed5244@pc33.aol.com>
Content-Type: application/sdp
Content-Length: 159

SIP/2.0 200 OK
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:werner.heisenberg@200.201.202.203>
Content-Type: application/sdp
Content-Length: 159
  
```

消息的序号
Transaction标识

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

三次握手出现的原因

- “请求—响应”的基础是**定时器超时重发**，这需要服务侧对请求有较快的响应时间
- 会话建立请求的响应复杂，**最终响应可能不会立即返回——被叫振铃定时器30s**

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

三次握手 Three-Way Handshake

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

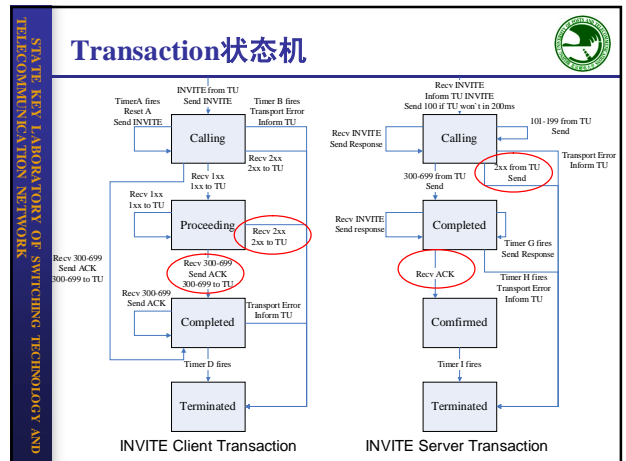
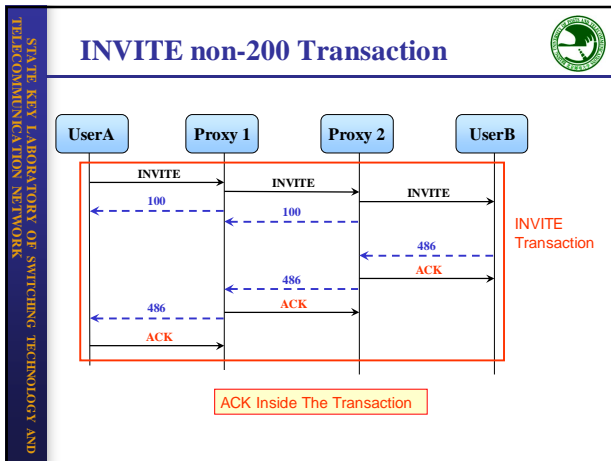
Transaction的分类

- INVITE Transaction
 - INVITE请求创建的Transaction
 - 对应于INVITE三次握手处理规则
 - INVITE Transaction包含200OK响应的ACK
- Non-INVITE Transaction
 - 非INVITE请求创建的Transaction
 - 对应于除了INVITE之外的使用两次握手处理规则的Transaction
- Regular Transaction:
 - A regular transaction is any transaction with a method other than INVITE, ACK, or CANCEL
 - 对应于标准的“请求—响应”处理规则
 - Regular Transaction一定是Non-INVITE Transaction
 - Non-INVITE Transaction不一定是Regular Transaction

STATE KEY LABORATORY OF SWITCHING TECHNOLOGY AND TELECOMMUNICATION NETWORK

INVITE Transaction

INVITE Transaction
ACK Outside The Transaction



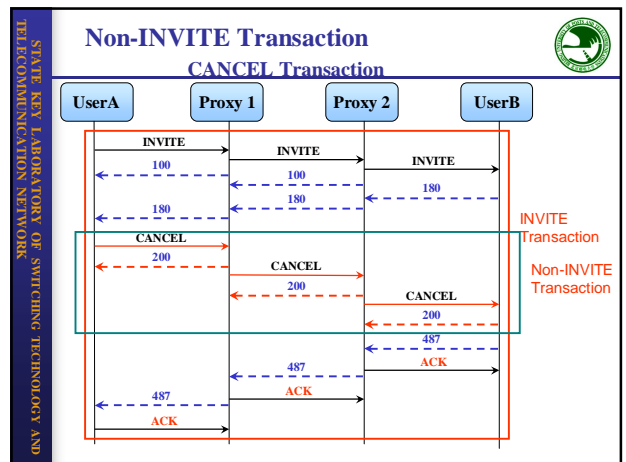
Transaction中ACK的处理

Question:

- INVITE Client Transaction收到3xx~6xx响应之后，发送ACK，跳转到Completed状态
- INVITE Server Transaction发送3xx~6xx响应之后，跳转到Completed状态，收到ACK之后跳转到Confirmed状态，但并没有将ACK上报给TU
 - ACK属于非2xx响应的INVITE Transaction，因此由CT维护ACK的发送
 - 但是ST收到ACK之后为什么不报ACK?
- INVITE Client Transaction收到2xx响应之后，跳转到Terminated状态，没有管ACK
- INVITE Server Transaction发送2xx响应之后，跳转到Terminated状态，没有管ACK
 - ACK不属于2xx响应的INVITE Transaction，因此CT、ST并不维护ACK的发送与接收
 - 但是既然ACK不能创建Transaction，INVITE-2xx Transaction不维护，又应该由谁维护呢?

Answer:

- ACK请求不能创建新的Transaction（没有响应）
- 在非2xx最终响应场景下
 - ACK是“Hop-Hop”逐跳处理的
 - TU并不关心ACK的内容
 - ACK由INVITE Transaction处理维护，不上报给TU
- 在2xx最终响应场景下
 - ACK是“End-End”处理的
 - TU需要关心ACK的内容，TU需要通过ACK维护2xx的应用层重发
 - ACK完全由TU维护，Transaction参数完全重新构建，不经过Transaction层，直接发送给Transport层



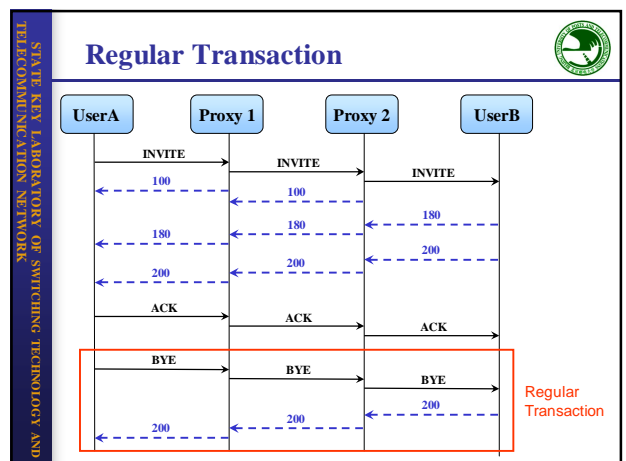
Transaction中CANCEL的处理

Question:

- INVITE Client Transaction和INVITE Server Transaction中都没有CANCEL的处理
 - CANCEL将建立独立于INVITE Transaction的自己的CANCEL Transaction
 - 那么CANCEL又应该由谁处理?
 - 如何才能取消INVITE Transaction呢?

Answer:

- CANCEL会创建一个新的Transaction，使用Non-INVITE Transaction状态机
- CANCEL Transaction和INVITE Transaction将由TU维护
- TU将依据INVITE Transaction的参数构造CANCEL消息，并发送给Transaction Layer
 - Transaction Layer建立CANCEL Client Transaction
 - 依据CSeq消息头的方法参数不同区分INVITE Client Transaction和CANCEL Client Transaction
- TU将依据CANCEL Transaction参数匹配需要被取消的INVITE Transaction
 - CSeq消息头的方法参数设置为“INVITE”
 - TU如果匹配到了INVITE Server Transaction，则向该INVITE Server Transaction发送487响应，驱动INVITE Server Transaction释放
 - TU如果匹配到了INVITE Server Transaction，则向发起取消的CANCEL Server Transaction发送200响应，驱动CANCEL Server Transaction释放
 - CANCEL是“Hop-Hop”逐跳处理的
- 注意，此时可能还没有建立Dialog，不能依据Dialog标识判断CANCEL和INVITE之间的关系
- 因此，CANCEL的是INVITE Transaction，而不是Dialog



Ack与Cancel的Transaction标识

- Ack的Transaction标识
 - 在收到非2xx最终响应的时候，ACK属于本INVITE Transaction
 - Transaction标识与INVITE Transaction一致 (Via的branch一致)
 - CSeq: 1 ACK (CSeq ID与INVITE一致)
 - 在收到2xx最终响应的时候，ACK不属于本INVITE Transaction
 - ACK并不创建新的Transaction，但会由应用层分配Transaction参数
 - Transaction标识与INVITE Transaction不一致 (Via的branch不一致)
 - CSeq: 1 ACK (CSeq ID与INVITE一致)
- CANCEL的Transaction标识
 - CANCEL请求创建Transaction
 - 请求的处理
 - CANCEL建立独立的Transaction
 - 但CANCEL与INVITE具有相同的Transaction标识 (Via branch一致)
 - CSeq: 1 CANCEL (CSeq ID与INVITE一致)
 - Transaction Layer通过 CSeq: 1 CANCEL 方法名区分INVITE和CANCEL分别建立的Transaction

Transaction与逻辑功能实体

- Transaction划分为 Client Transaction (CT)和Server Transaction (ST) C/S结构
- Stateful Proxy同时包含CT和ST
- Stateless Proxy不包含Transaction
- User Agent在特定时刻只能包含CT或ST中的一种

Dialog

Dialog的定义

Dialog的处理规则

Dialog

- Dialog用于维护UA之间的状态
- A dialog is a **peer-to-peer** SIP relationship between two UAs that persists for some time.
- A dialog is **established** by SIP messages, such as a **2xx response to an INVITE** request.
- **Dialogs are created through the generation of non-failure responses to requests with specific methods**
- the **BYE** method **terminates** a session and the dialog associated with it

Dialog的标识

A dialog is identified by a **call identifier**, **local tag**, and a **remote tag**.

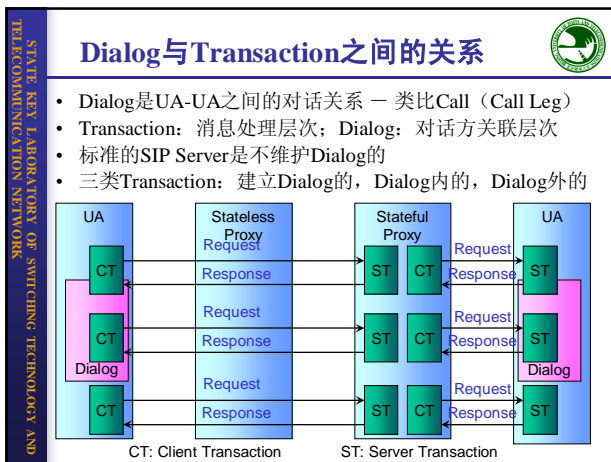
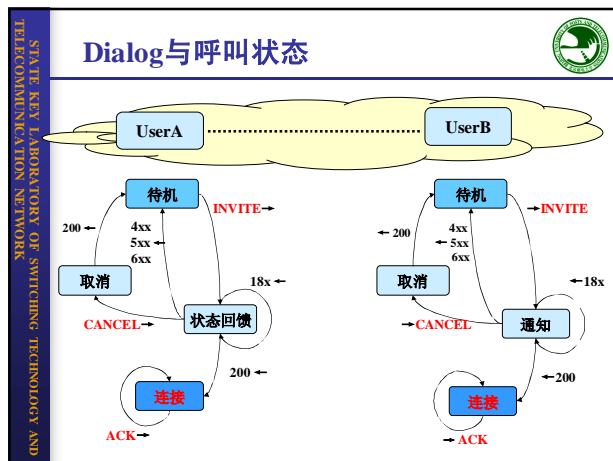
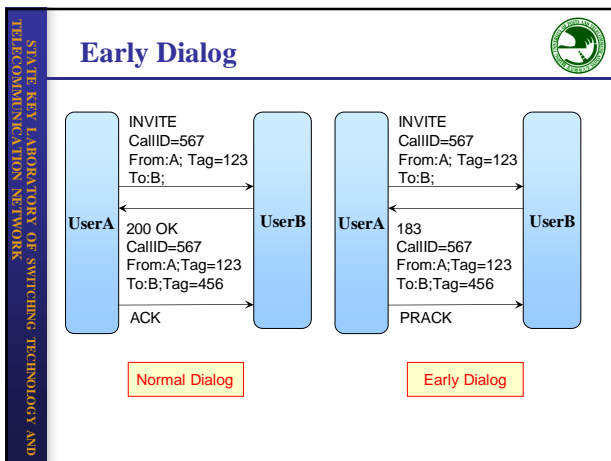
```

INVITE sip:werner.heisenberg@munich.de SIP/2.0
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
Max-Forwards: 70
To: Heisenberg <sip:werner.heisenberg@munich.de>
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:schroed5244@pc33.aol.com>
Content-Type: application/sdp
Content-Length: 159

SIP/2.0 200 OK
Via: SIP/2.0/UDP 100.101.102.103:5060;branch=z9hG4bKmp17a
To: Heisenberg <sip:werner.heisenberg@munich.de>;tag=314159
From: E. Schroedinger <sip:schroed5244@aol.com>;tag=42
Call-ID: 10@100.101.102.103
CSeq: 1 INVITE
Contact: <sip:werner.heisenberg@200.201.102.203>
Content-Type: application/sdp
Content-Length: 159
  
```

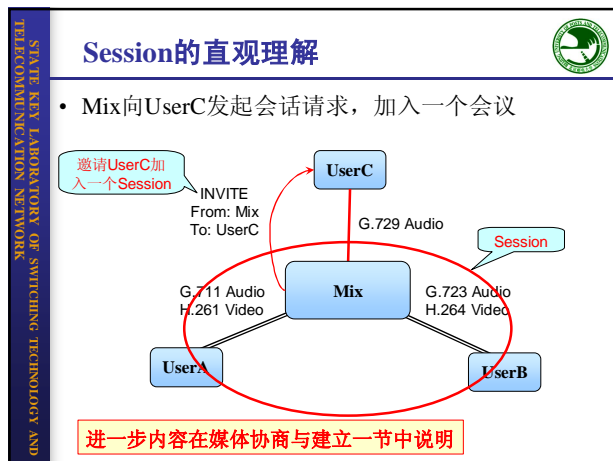
} 共同标识一个 Dialog

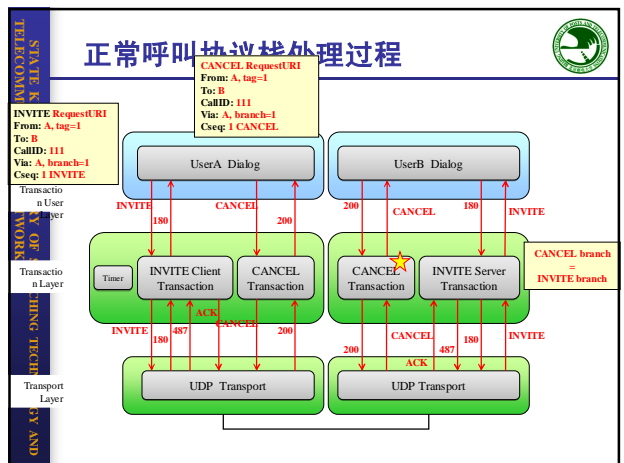
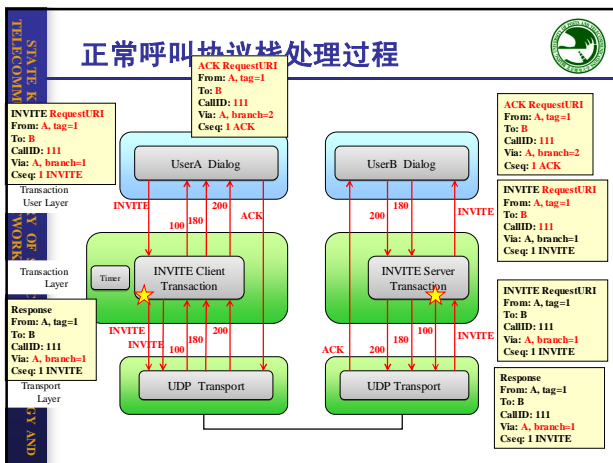
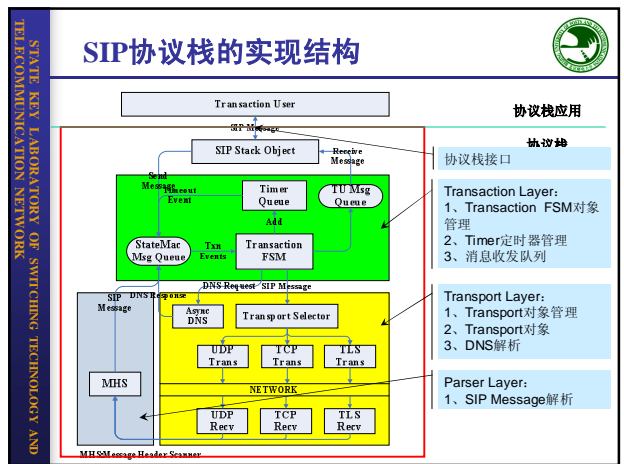
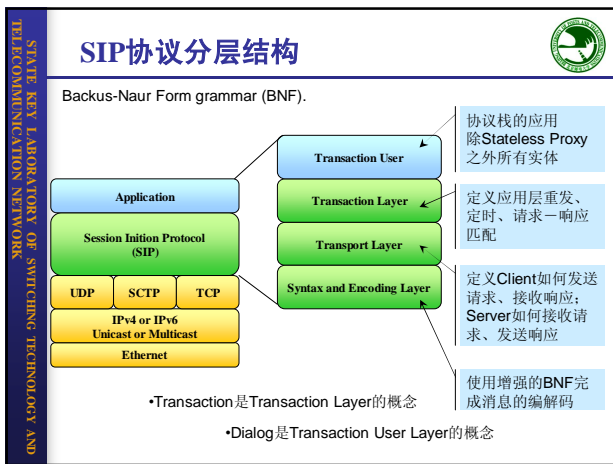
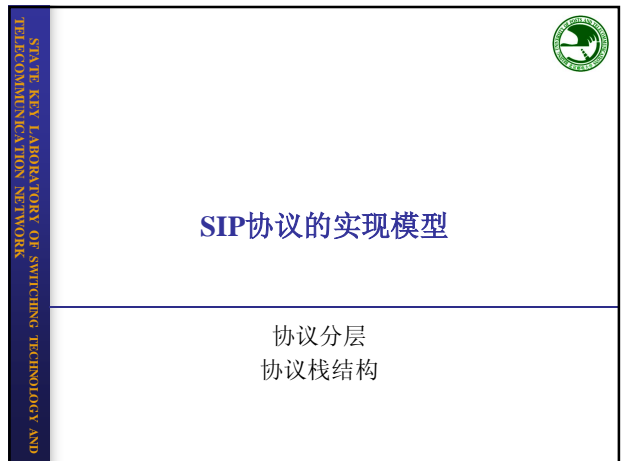
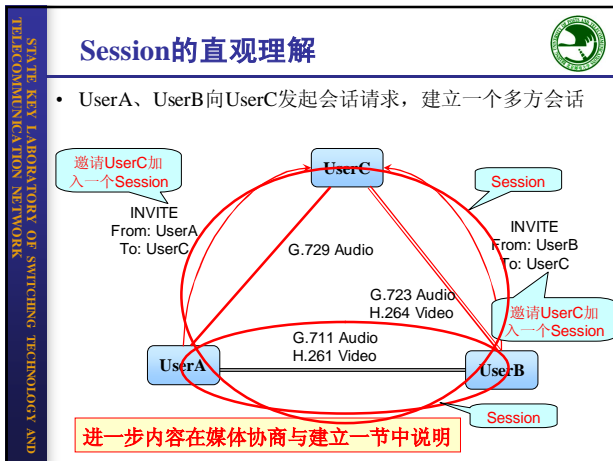
Dialog的建立与删除



Session

- A multimedia session is a **set of multimedia senders and receivers** and the **data streams** flowing from senders to receivers (RFC 2327 SDP)
 - A multimedia conference is an example of a multimedia session
 - A session as defined for SDP can comprise one or more RTP sessions
- As defined, a callee can be invited several times, by different calls, to the same session.
- If SDP is used, a session is defined by the concatenation of the SDP user name, session id, network type, address type, and address elements in the origin field
- INVITE是在邀请什么?
 - 加入一个Session

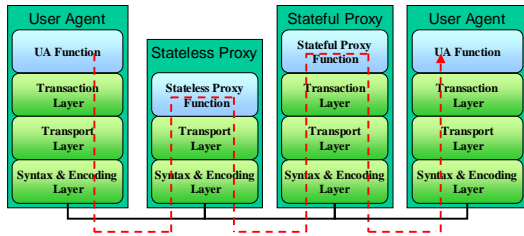




SIP协议分层与功能实体



- 逻辑设备与物理设备
- 应用可以变换角色，可以变换使用的层次
- Stateless Proxy 不包含Transaction Layer



协议基本架构总结



- 应用层保证会话可靠建立机制：三次握手
- 协议的核心：Transaction
- 协议的用户层概念：Dialog
- SIP邀请建立的Session概念
- SIP协议分层
- SIP协议栈结构

思考题



- 使用三次握手的原因
- 如何理解Transaction
- 如何理解Dialog
- 如何理解Session
- SIP协议分层与基本核心架构的关系