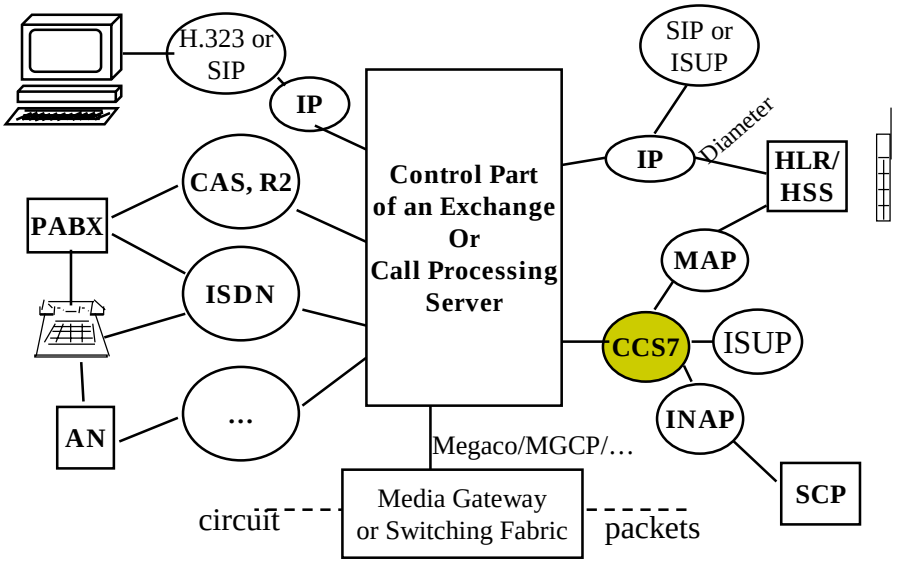
TCAP - Transaction Capabilities Application Part is used by

- ✓ **Mobile services (roaming and mobility management)**
- ✓ **Intelligent Network services**
- ✓ **Services that are independent of voice circuits (look-ahead …)**
- ✓ **O&M applications**
- ✓ **etc**

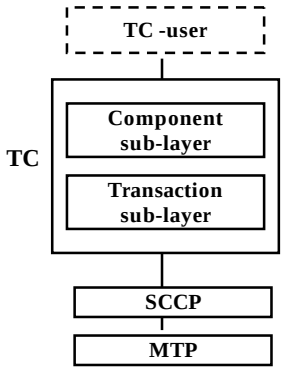
TCAP provides generic services supporting the execution of distributed transactions.
Parties in the transactions can be exchanges, service nodes, data bases etc.

TCAP offers a way to implement services that are independent of network resources.

Summary of course scope



TCAP has two sub-layers

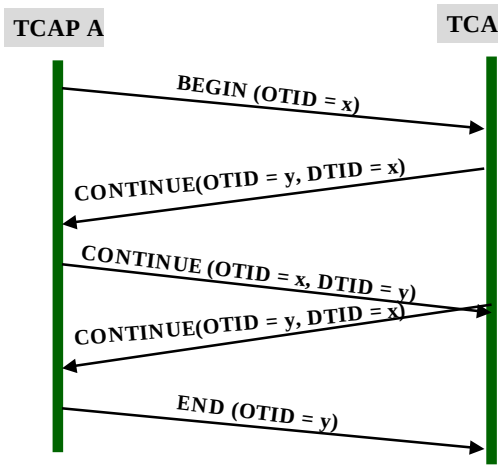


Component sub-layer: data units of the application protocol, requests and responses, dialogues: application context

Transaction sub-layer: message exchange between parties, optionally dialogues between parties.

TCAP has a lot of similarity with ROSE (Remote Operation Service Element) and ACSE (Association Control Service Element). ROSE ja ACSE are OSI layer 7 services.

A TCAP use case



Begin begins a dialogue

During the dialogue Continue -messages are sent in both directions.

End-message closes the dialogue.

OTID -identifies the dialogue/ for the sender of the transaction.

DTID -identifies dialogue/ for the object of the transaction.

TCAP supports four operation types

- ✓ **Class 1 - Both success and failure are reported**
- ✓ **Class 2 - Only failures are reported.**
- ✓ **Class 3 - Only success is reported.**
- ✓ **Class 4 - Nothing is reported**

An operation is identified by the Invoke-Id - identifier.

Indication (ind) is associated with the request (req) based on the Invoke-id.

A user may have many ongoing active operations simultaneously.

Operations are identified and chained using the Invoke-Id

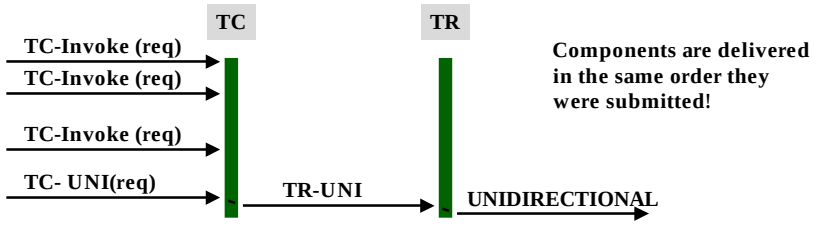
- ✓ **Operation is identified by the Invoke-Id.**
- ✓ **Indication (ind) is associated with the request (req) based on the Invoke-id.**
- ✓ **The Response can be a new operation request that is chained to the previous operation request using a link-identifier.**
- ✓ **A user may have many simultaneous operations.**

The result of an operation sent to a remote system can be

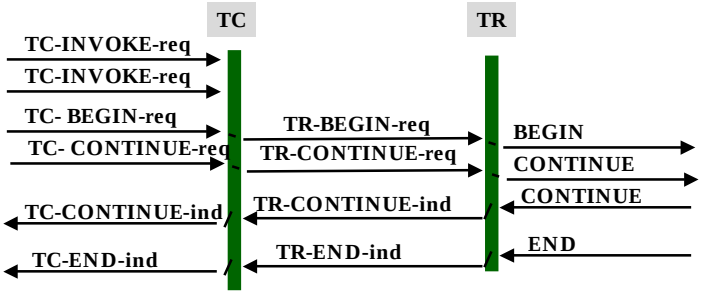
- ✓ **Result: Operation succeeded.**
  - › **The result can also be segmented (chained)**
- ✓ **Error: Operation failed.**
- ✓ **Reject: Execution of the operation is not possible.**
- ✓ **Before sending the result, the remote system can send an arbitrary number of linked operations.**

Non-structured dialogue transfers one or more components

- ✓ **TC-user can send many components in Class 4 operations by a UNIDIRECTIONAL message.**
- ✓ **Components with the same dialogue -id can be sent in one message.**
- ✓ **Control over sequencing of operations is left to the application.**

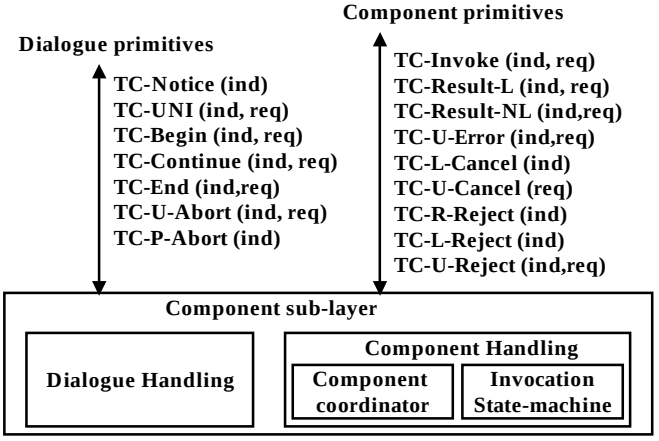


A Structured dialogue has a beginning, information transfer, ending or abort



- **Begin causes a *transaction identifier* to be reserved.**
- **The remote system can either continue the transaction or close it.**
- **Continue - messages are exchanged in a full-duplex mode.**
- **Closing options:**
  - **based on pre-arrangement independently**
  - **normally by the End-message or "abnormally" by an Abort message**

The Component sub-layer is split into dialogue handling and component handling



Component handling primitives are

TC_INVOKE - Invocation of an operation which may be linked to another operation

TC_RESULT_L - Only result or last part of segmented result of a successful operation

TC_RESULT_NL - non-last part of segmented result

TC_U_ERROR - reply to a previously invoked op that failed

TC_L_CANCEL - informs user of local timeout

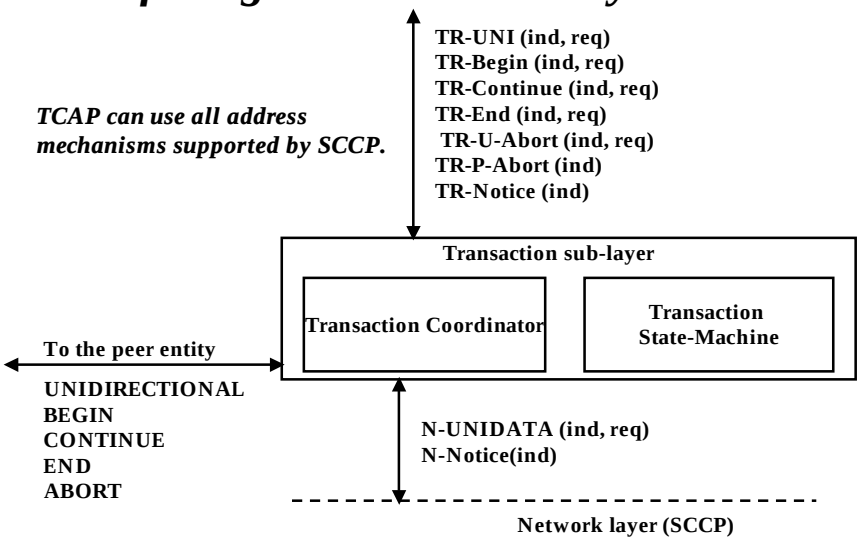
TC_U_CANCEL - Causes local termination of op on TC_user request

TC_L_REJECT - local reject by Component sub-layer to TC_user

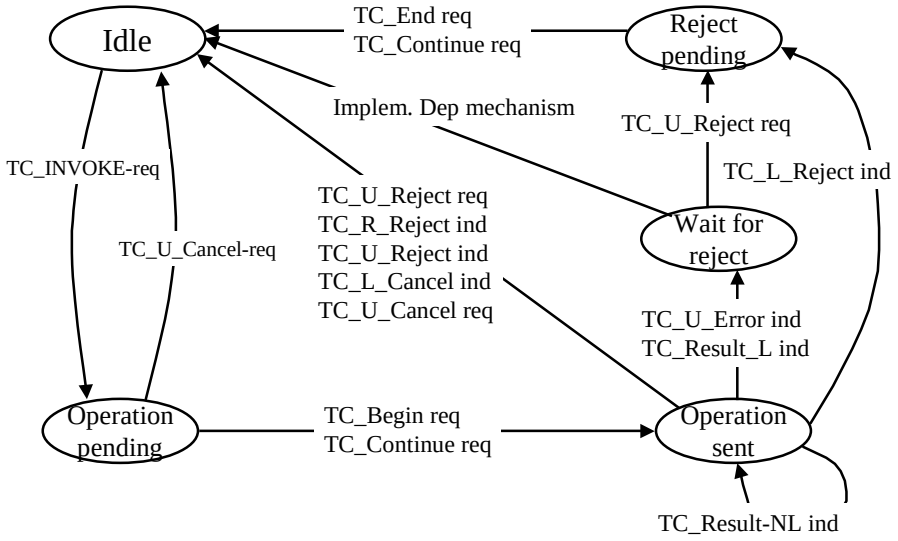
TC_R_REJECT - remote reject by remote component sub-layer

TC_U_REJECT - Rejection by TC_user indicating malformation

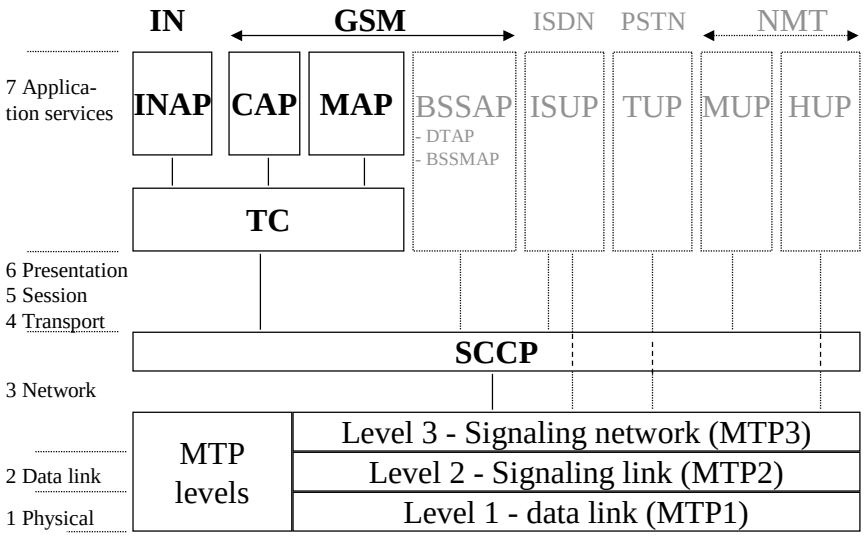
Transaction sub-layer handles the interfacing to the network layer



State transition Diagram for Class 1 Operations



Most important users of TCAP are..



TCAP added value is

- ✓ **Decoupling the actions and states of an application from communication states for managing the flow of information with the remote end**
- ✓ **Takes care of managing the communication with the peer – let's the application concentrate on essential matters**
  - › **four classes of service**
  - › **report on success tells the application that the remote end has done its job for sure**
  - › **report on failures speeds up recovery (but an application can not really rely on getting the report on every failure!)**
  - › **or alternatively can let the application take care of all acknowledgements**