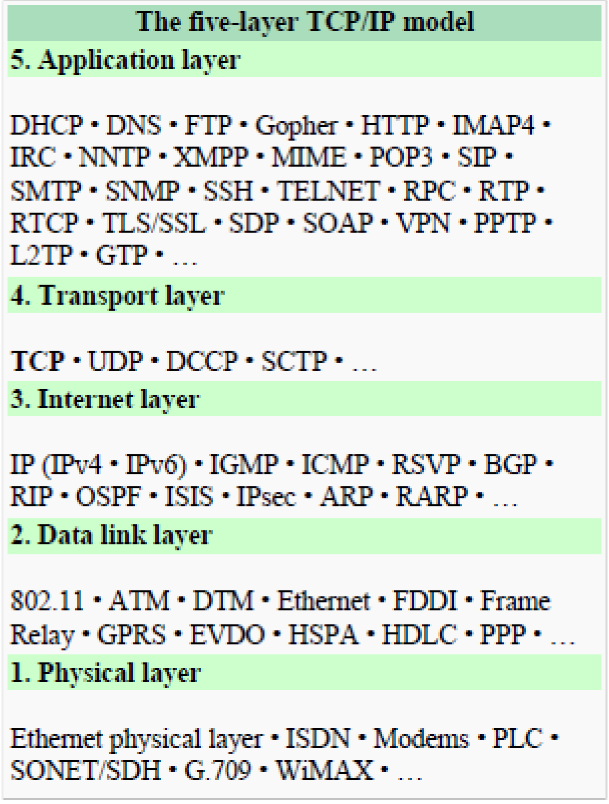
Unit-3: Mobile Transport Layer: Traditional TCP ,Indirect TCP ,Snooping TCP ,Mobile TCP, Fast retransmit/fast recovery ,Transmission /time-out freezing ,Selective retransmission, Transaction oriented TCP

Traditional TCP

The **Transmission Control Protocol** (**TCP**) is one of the core protocols of the Internet protocol suite, often simply referred to as TCP/IP. TCP is reliable, guarantees in-order delivery of data and incorporates congestion control and flow control mechanisms.



The five-layer TCP/IP model
5. Application layer
DHCP • DNS • FTP • Gopher • HTTP • IMAP4 • IRC • NNTP • XMPP • MIME • POP3 • SIP • SMTP • SNMP • SSH • TELNET • RPC • RTP • RTCP • TLS/SSL • SDP • SOAP • VPN • PPTP • L2TP • GTP • ...
4. Transport layer
TCP • UDP • DCCP • SCTP • ...
3. Internet layer
IP (IPv4 • IPv6) • IGMP • ICMP • RSVP • BGP • RIP • OSPF • ISIS • IPsec • ARP • RARP • ...
2. Data link layer
802.11 • ATM • DTM • Ethernet • FDDI • Frame Relay • GPRS • EVDO • HSPA • HDLC • PPP • ...
1. Physical layer
Ethernet physical layer • ISDN • Modems • PLC • SONET/SDH • G.709 • WiMAX • ...

TCP supports many of the Internet's most popular application protocols and resulting applications, including the World Wide Web, e-mail, File Transfer Protocol and Secure Shell. In the Internet protocol suite, TCP is the intermediate layer between the Internet layer and application layer.

The major responsibilities of TCP in an active session are to:

• **Provide reliable in-order transport of data**: to not allow losses of data.

• **Control congestions in the networks**: to not allow degradation of the network performance,

• **Control a packet flow between the transmitter and the receiver**: to not exceed the receiver's capacity.

TCP uses a number of mechanisms to achieve high performance and avoid 'congestion collapse', where network performance can fall by several orders of magnitude. These mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger collapse. There are several mechanisms of TCP that influence the efficiency of TCP in a mobile environment. Acknowledgments for data sent, or lack of acknowledgments, are used by senders to implicitly interpret network conditions between the TCP sender and receiver.

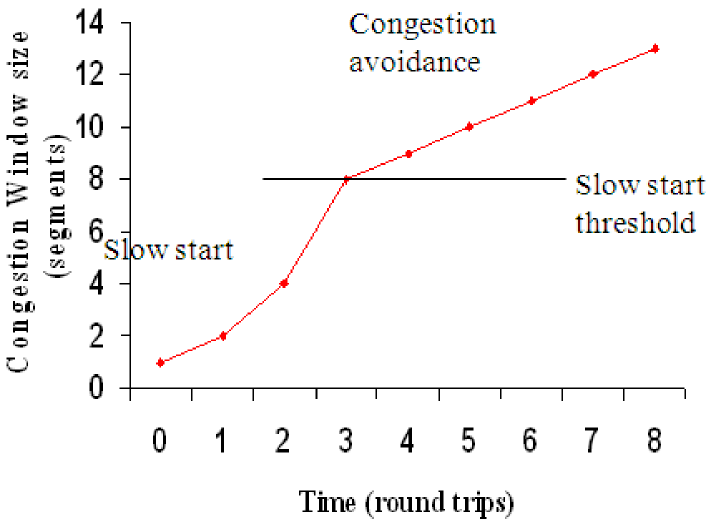
Congestion Control

A transport layer protocol such as TCP has been designed for fixed networks with fixed end- systems. Congestion may appear from time to time even in carefully designed networks. The packet buffers of a router are filled and the router cannot forward the packets fast enough because the sum of the input rates of packets destined for one output link is higher than the capacity of the output link. The only thing a router can do in this situation is to drop packets. A dropped packet is lost for the transmission, and the receiver notices a gap in the packet stream. Now the receiver does not directly tell the sender which packet is missing, but continues to acknowledge all in-sequence packets up to the missing one.

The sender notices the missing acknowledgement for the lost packet and assumes a packet loss due to congestion. Retransmitting the missing packet and continuing at full sending rate would now be unwise, as this might only increase the congestion. To mitigate congestion, TCP slows down the transmission rate dramatically. All other TCP connections experiencing the same congestion do exactly the same so the congestion is soon resolved.

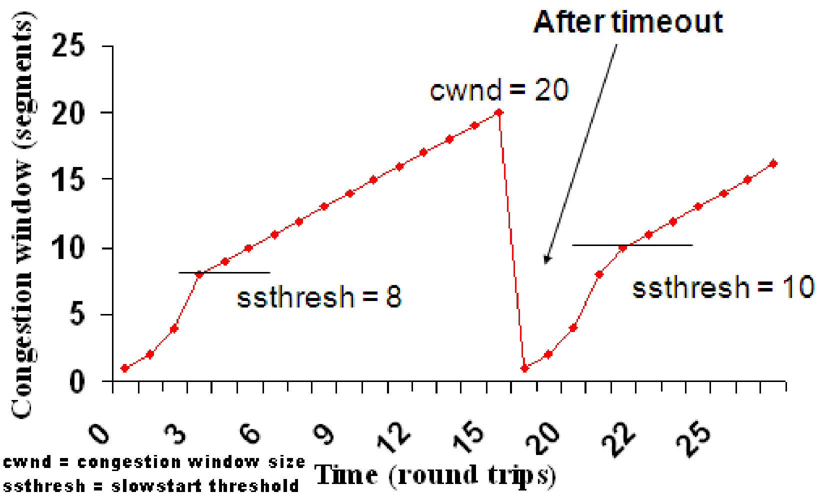
Slow start

TCP's reaction to a missing acknowledgement is quite drastic, but it is necessary to get rid of congestion quickly. The behavior TCP shows after the detection of congestion is called **slow start.** The sender always calculates a **congestion window** for a receiver. The start size of the congestion window is one segment (TCP packet). The sender sends one packet and waits for acknowledgement. If this acknowledgement arrives, the sender increases the congestion window by one, now sending two packets (congestion window = 2). This scheme doubles the congestion window every time the acknowledgements come back, which takes one round trip time (RTT). This is called the exponential growth of the congestion window in the slow start mechanism.



But doubling the congestion window is too dangerous. The exponential growth stops at the **congestion threshold**. As soon as the congestion window reaches the congestion threshold, further increase of the transmission rate is only linear by adding 1 to the congestion window each time the acknowledgements come back.

Linear increase continues until a time-out at the sender occurs due to a missing acknowledgement, or until the sender detects a gap in transmitted data because of continuous acknowledgements for the same packet. In either case the sender sets the congestion threshold to half of the current congestion window. The congestion window itself is set to one segment and the sender starts sending a single segment. The exponential growth starts once more up to the new congestion threshold, then the window grows in linear fashion.



Fast retransmit/fast recovery

The congestion threshold can be reduced because of two reasons. First one is if the sender receives continuous acknowledgements for the same packet. It informs the sender that the receiver has got all the packets upto the acknowledged packet in the sequence and also the receiver is receiving something continuously from the sender. The gap in the packet stream is not due to congestion, but a simple packet loss due to a transmission error. The sender can now retransmit the missing packet(s) before the timer expires. This behavior is called **fast retransmit**. It is an early enhancement for preventing slow-start to trigger on losses not caused by congestion. The receipt of acknowledgements shows that there is no congestion to justify a slow start. The sender can continue with the current congestion window. The sender performs a **fast recovery** from the packet loss. This mechanism can improve the efficiency of TCP dramatically. The other reason for activating slow start is a time-out due to a missing acknowledgement. TCP using fast retransmit/fast recovery interprets this congestion in the network and activates the slow start mechanism.

- The advantage of this method is its simplicity. Minor changes in the MH's software results in performance increase. No changes are required in FA or CH.
- The disadvantage of this scheme is insufficient isolation of packet losses. It mainly focuses on problems regarding Handover. Also it effects the efficiency when a CH transmits already delivered packets.

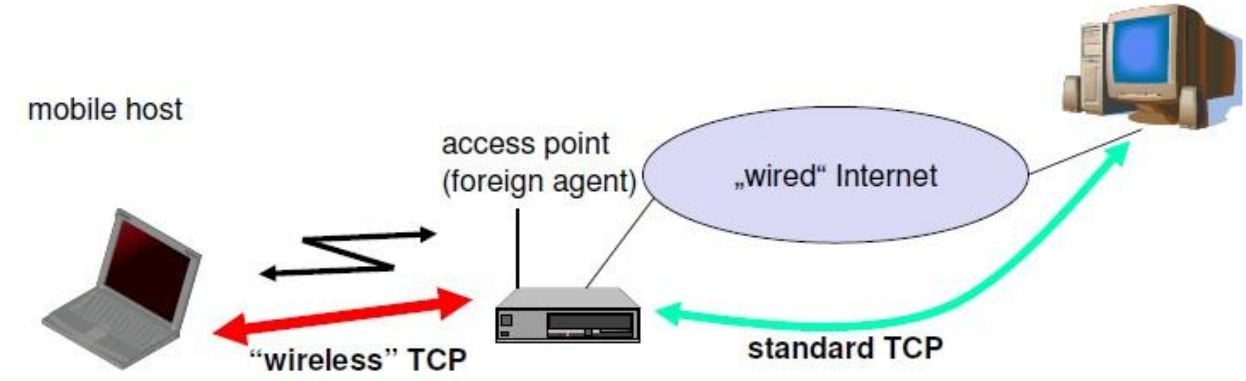
Problems with Traditional TCP in wireless environments

- Slow Start mechanism in fixed networks decreases the efficiency of TCP if used with mobile receivers or senders.
- Error rates on wireless links are orders of magnitude higher compared to fixed fiber or copper links. This makes compensation for packet loss by TCP quite difficult.
- Mobility itself can cause packet loss. There are many situations where a soft handover from one access point to another is not possible for a mobile end-system.
- Standard TCP reacts with slow start if acknowledgements are missing, which does not help in the case of transmission errors over wireless links and which does not really help during handover. This behavior results in a severe performance degradation of an unchanged TCP if used together with wireless links or mobile nodes

Classical TCP Improvements

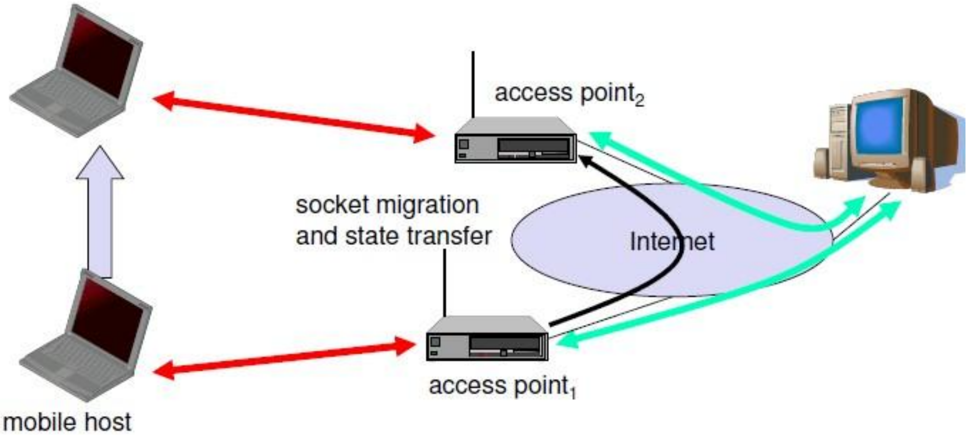
Indirect TCP (I-TCP)

Indirect TCP segments a TCP connection into a fixed part and a wireless part. The following figure shows an example with a mobile host connected via a wireless link and an access point to the 'wired' internet where the correspondent host resides.



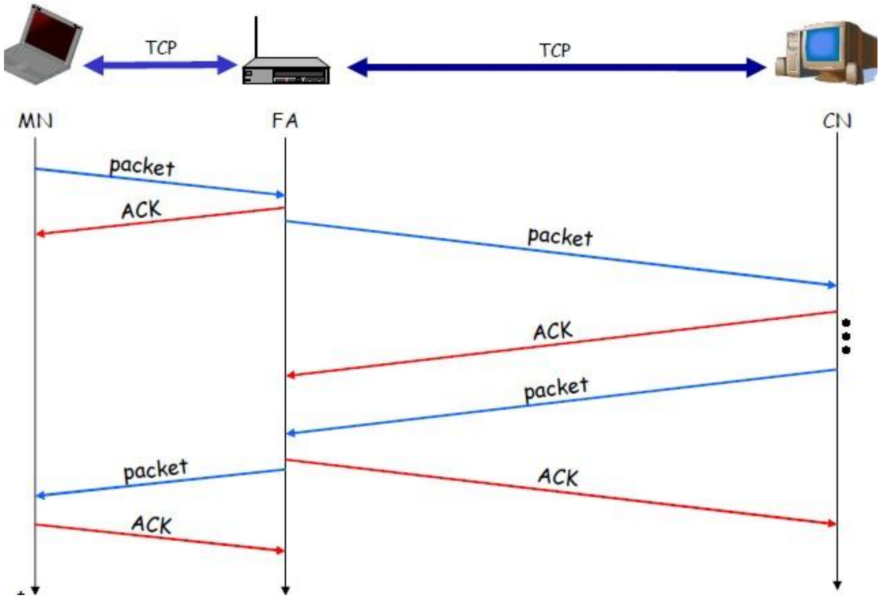
Standard TCP is used between the fixed computer and the access point. No computer in the internet recognizes any changes to TCP. Instead of the mobile host, the access point now terminates the standard TCP connection, acting as a proxy. This means that the access point is now seen as the mobile host for the fixed host and as the fixed host for the mobile host. Between the access point and the mobile host, a special TCP, adapted to wireless links, is used. However, changing TCP for the wireless link is not a requirement. A suitable place for segmenting the connection is at the foreign agent as it not only controls the mobility of the mobile host anyway and can also hand over the connection to the next foreign agent when the mobile host moves on.

The foreign agent acts as a proxy and relays all data in both directions. If CH (correspondent host) sends a packet to the MH, the FA acknowledges it and forwards it to the MH. MH acknowledges on successful reception, but this is only used by the FA. If a packet is lost on the wireless link, CH doesn't observe it and FA tries to retransmit it locally to maintain reliable data transport. If the MH sends a packet, the FA acknowledges it and forwards it to CH. If the packet is lost on the wireless link, the mobile hosts notice this much faster due to the lower round trip time and can directly retransmit the packet. Packet loss in the wired network is now handled by the foreign agent.



Socket and state migration after handover of a mobile host

During handover, the buffered packets, as well as the system state (packet sequence number, acknowledgements, ports, etc), must migrate to the new agent. No new connection may be established for the mobile host, and the correspondent host must not see any changes in connection state. Packet delivery in I-TCP is shown below:



Advantages of I-TCP

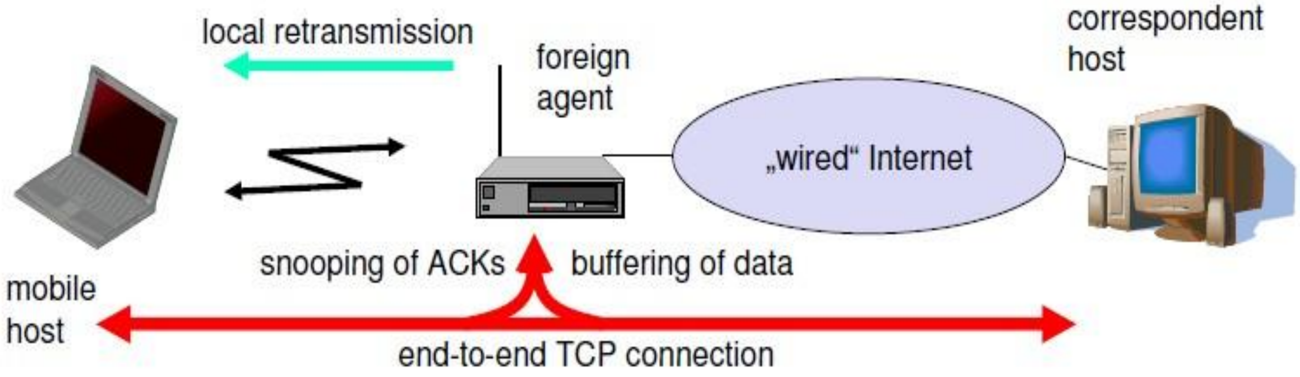
- No changes in the fixed network necessary, no changes for the hosts (TCP protocol) necessary, all current optimizations to TCP still work
- Simple to control, mobile TCP is used only for one hop between, e.g., a foreign agent and mobile host
  1. transmission errors on the wireless link do not propagate into the fixed network
  2. therefore, a very fast retransmission of packets is possible, the short delay on the mobile hop s known
- It is always dangerous to introduce new mechanisms in a huge network without knowing exactly how they behave.
  - New optimizations can be tested at the last hop, without jeopardizing the stability of the Internet.
- It is easy to use different protocols for wired and wireless networks.

Disadvantages of I-TCP

- Loss of end-to-end semantics:- an acknowledgement to a sender no longer means that a receiver really has received a packet, foreign agents might crash.
- Higher latency possible:- due to buffering of data within the foreign agent and forwarding to a new foreign agent
- Security issue:- The foreign agent must be a trusted entity

Snooping TCP

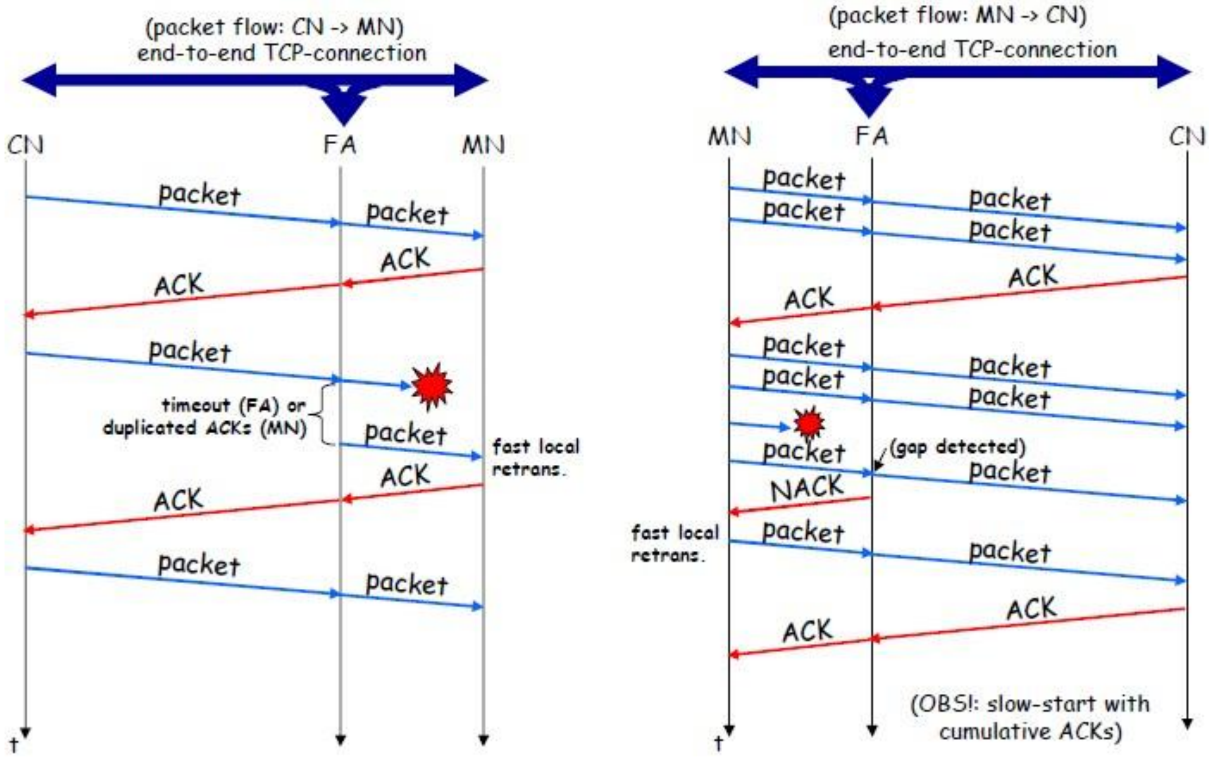
The main drawback of I-TCP is the segmentation of the single TCP connection into two TCP connections, which loses the original end-to-end TCP semantic. A new enhancement, which leaves the TCP connection intact and is completely transparent, is Snooping TCP. The main function is to buffer data close to the mobile host to perform fast local retransmission in case of packet loss.



Snooping TCP as a transparent TCP extension

Here, the foreign agent buffers all packets with **destination mobile host** and additionally 'snoops' the packet flow in both directions to recognize acknowledgements. The foreign agent buffers every packet until it receives an acknowledgement from the mobile host. If the FA does not receive an acknowledgement from the mobile host within a certain amount of time, either the packet or the acknowledgement has been lost. Alternatively, the foreign agent could receive a duplicate ACK which also shows the loss of a packet. Now, the FA retransmits the packet directly from the buffer thus performing a faster retransmission compared to the CH. For transparency, the FA does not acknowledge data to the CH, which would violate end-to-end semantic in case of a FA failure. The foreign agent can filter the duplicate acknowledgements to avoid unnecessary retransmissions of data from the correspondent host. If the foreign agent now crashes, the time-out of the correspondent host still works and triggers a retransmission. The foreign agent may discard duplicates of packets already retransmitted locally and acknowledged by the mobile host. This avoids unnecessary traffic on the wireless link.

For data transfer from the mobile host with **destination correspondent host**, the FA snoops into the packet stream to detect gaps in the sequence numbers of TCP. As soon as the foreign agent detects a missing packet, it returns a negative acknowledgement (NACK) to the mobile host. The mobile host can now retransmit the missing packet immediately. Reordering of packets is done automatically at the correspondent host byTCP.



Snooping TCP: Packet delivery

Advantages of snooping TCP:

- The end-to-end TCP semantic is preserved.
- Most of the enhancements are done in the foreign agent itself which keeps correspondent host unchanged.
- Handover of state is not required as soon as the mobile host moves to another foreign agent. Even though packets are present in the buffer, time out at the CH occurs and the packets are transmitted to the new COA.
- No problem arises if the new foreign agent uses the enhancement or not. If not, the approach automatically falls back to the standard solution.

Disadvantages of snooping TCP

- Snooping TCP does not isolate the behavior of the wireless link as well as I-TCP. Transmission errors may propagate till CH.
- Using negative acknowledgements between the foreign agent and the mobile host assumes additional mechanisms on the mobile host. This approach is no longer transparent for arbitrary mobile hosts.
- Snooping and buffering data may be useless if certain encryption schemes are applied end-to-end between the correspondent host and mobile host. If encryption is used above the transport layer, (eg. SSL/TLS), snooping TCP can be used.

Mobile TCP

Both I-TCP and Snooping TCP does not help much, if a mobile host gets disconnected. The **M-TCP (mobile TCP)** approach has the same goals as I-TCP and snooping TCP: to prevent the sender window from shrinking if bit errors or disconnection but not congestion cause current problems. M-TCP wants to improve overall throughput, to lower the delay, to maintain end-to-end semantics of TCP, and to provide a more efficient handover. Additionally, M-TCP is especially adapted to the problems arising from lengthy or frequent disconnections. M-TCP splits the TCP connection into two parts as I-TCP does. An unmodified TCP is used on the standard host-**supervisory host (SH)** connection, while an optimized TCP is used on the SH-MH connection.

The SH monitors all packets sent to the MH and ACKs returned from the MH. If the SH does not receive an ACK for some time, it assumes that the MH is disconnected. It then chokes the sender by setting the sender’s window size to 0. Setting the window size to 0 forces the sender to go into **persistent mode**, i.e., the state of the sender will not change no matter how long the receiver is disconnected. This means that the sender will not try to retransmit data. As soon as the SH (either the old SH or a new SH) detects connectivity again, it reopens the window of the sender to the old value. The sender can continue sending at full speed. This mechanism does not require changes to the sender’s TCP. The wireless side uses an adapted

TCP that can recover from packet loss much faster. This modified TCP does not use slow start, thus, M-TCP needs a **bandwidth manager** to implement fair sharing over the wireless link.

Advantages of M-TCP:

- It maintains the TCP end-to-end semantics. The SH does not send any ACK itself but forwards the ACKs from the MH.
- If the MH is disconnected, it avoids useless retransmissions, slow starts or breaking connections by simply shrinking the sender's window to 0.
- As no buffering is done as in I-TCP, there is no need to forward buffers to a new SH. Lost packets will be automatically retransmitted to the SH.

Disadvantages of M-TCP:

- As the SH does not act as proxy as in I-TCP, packet loss on the wireless link due to bit errors is propagated to the sender. M-TCP assumes low bit error rates, which is not always a valid assumption.
- A modified TCP on the wireless link not only requires modifications to the MH protocol software but also new network elements like the bandwidth manager.

Transmission/time-out freezing

Often, MAC layer notices connection problems even before the connection is actually interrupted from a TCP point of view and also knows the real reason for the interruption. The MAC layer can inform the TCP layer of an upcoming loss of connection or that the current interruption is not caused by congestion. TCP can now stop sending and 'freezes' the current state of its congestion window and further timers. If the MAC layer notices the upcoming interruption early enough, both the mobile and correspondent host can be informed. With a fast interruption of the wireless link, additional mechanisms in the access point are needed to inform the correspondent host of the reason for interruption. Otherwise, the correspondent host goes into slow start assuming congestion and finally breaks the connection.

As soon as the MAC layer detects connectivity again, it signals TCP that it can resume operation at exactly the same point where it had been forced to stop. For TCP time simply does not advance, so no timers expire.

Advantages:

- It offers a way to resume TCP connections even after long interruptions of the connection.
- It can be used together with encrypted data as it is independent of other TCP mechanisms such as sequence no or acknowledgements

Disadvantages:

- Lots of changes have to be made in software of MH, CH and FA.

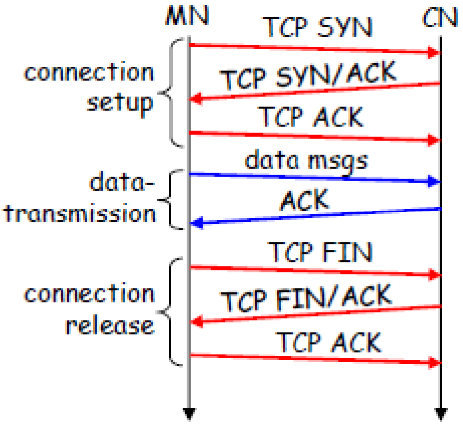
Selective retransmission

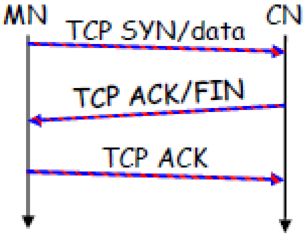
A very useful extension of TCP is the use of selective retransmission. TCP acknowledgements are cumulative, i.e., they acknowledge in-order receipt of packets up to a certain packet. A single acknowledgement confirms reception of all packets upto a certain packet. If a single packet is lost, the sender has to retransmit everything starting from the lost packet (go-back-n retransmission). This obviously wastes bandwidth, not just in the case of a mobile network, but for any network.

Using selective retransmission, TCP can indirectly request a selective retransmission of packets. The receiver can acknowledge single packets, not only trains of in-sequence packets. The sender can now determine precisely which packet is needed and can retransmit it. The **advantage** of this approach is obvious: a sender retransmits only the lost packets. This lowers bandwidth requirements and is extremely helpful in slow wireless links. The disadvantage is that a more complex software on the receiver side is needed. Also more buffer space is needed to resequence data and to wait for gaps to be filled.

Transaction-oriented TCP

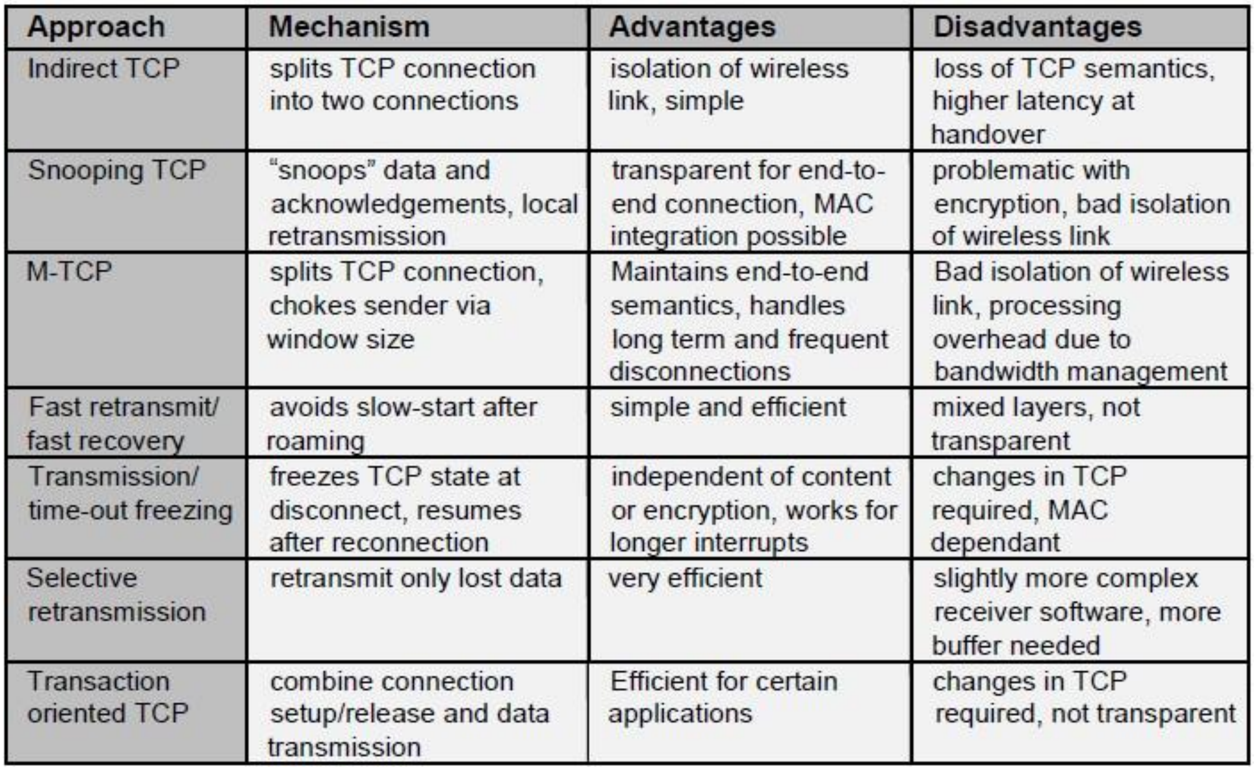
Assume an application running on the mobile host that sends a short request to a server from time to time, which responds with a short message and it requires reliable TCP transport of the packets. For it to use normal TCP, it is inefficient because of the overhead involved. Standard TCP is made up of three phases: setup, data transfer and release. First, TCP uses a three-way handshake to establish the connection. At least one additional packet is usually needed for transmission of the request, and requires three more packets to close the connection via a three-way handshake. So, for sending one data packet, TCP may need seven packets altogether. This kind of overhead is acceptable for long sessions in fixed networks, but is quite inefficient for short messages or sessions in wireless networks. This led to the development of transaction-oriented TCP (T/TCP).





T/TCP can combine packets for connection establishment and connection release with user data packets. This can reduce the number of packets down to two instead of seven. The obvious **advantage** for certain applications is the reduction in the overhead which standard TCP has for connection setup and connection release. Disadvantage is that it requires changes in the software in mobile host and all correspondent hosts. This solution does not hide mobility anymore. Also, T/TCP exhibits several security problems.

Classical Enhancements to TCP for mobility: A comparison



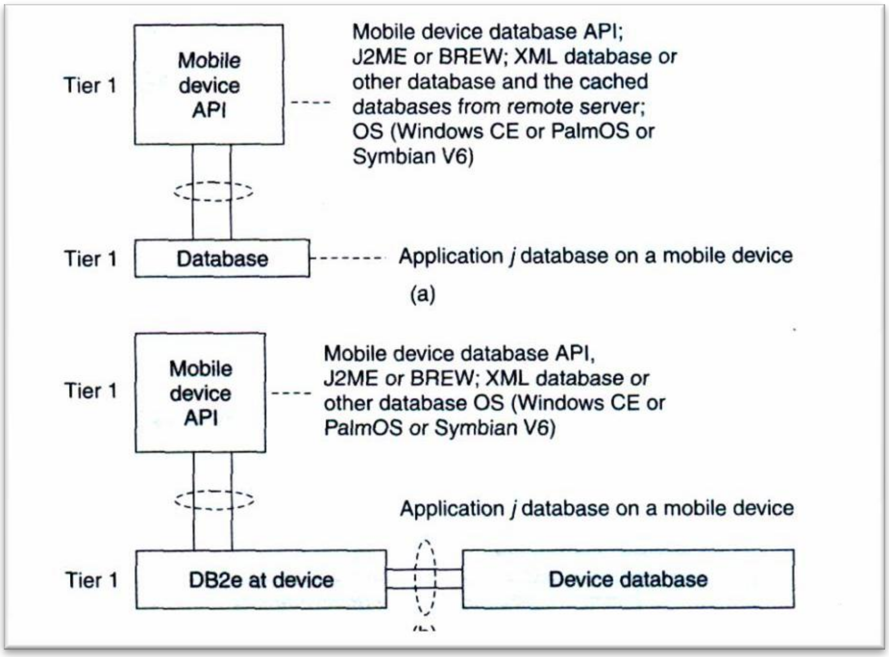
Approach	Mechanism	Advantages	Disadvantages
Indirect TCP	splits TCP connection into two connections	isolation of wireless link, simple	loss of TCP semantics, higher latency at handover
Snooping TCP	"snoops" data and acknowledgements, local retransmission	transparent for end-to-end connection, MAC integration possible	problematic with encryption, bad isolation of wireless link
M-TCP	splits TCP connection, chokes sender via window size	Maintains end-to-end semantics, handles long term and frequent disconnections	Bad isolation of wireless link, processing overhead due to bandwidth management
Fast retransmit/ fast recovery	avoids slow-start after roaming	simple and efficient	mixed layers, not transparent
Transmission/ time-out freezing	freezes TCP state at disconnect, resumes after reconnection	independent of content or encryption, works for longer interrupts	changes in TCP required, MAC dependant
Selective retransmission	retransmit only lost data	very efficient	slightly more complex receiver software, more buffer needed
Transaction oriented TCP	combine connection setup/release and data transmission	Efficient for certain applications	changes in TCP required, not transparent

Database issues: Hoarding techniques, caching invalidation mechanisms. client server computing with adapt ion , power-aware and context-aware computing, transactional models, query processing, recovery, and quality of service issues

A database is a collection of systematically stored records or information. Databases store data in a particular logical manner. A mobile device is not always connected to the server or network; neither does the device retrieve data from a server or a network for each computation. Rather, the device caches some specific data, which may be required for future computations, during the interval in which the device is connected to the server or network. Caching entails saving a copy of select data or a part of a database from a connected system with a large database. The cached data is hoarded in the mobile device database. Hoarding of the cached data in the database ensures that even when the device is not connected to the network, the data required from the database is available for computing.

Database Hoarding

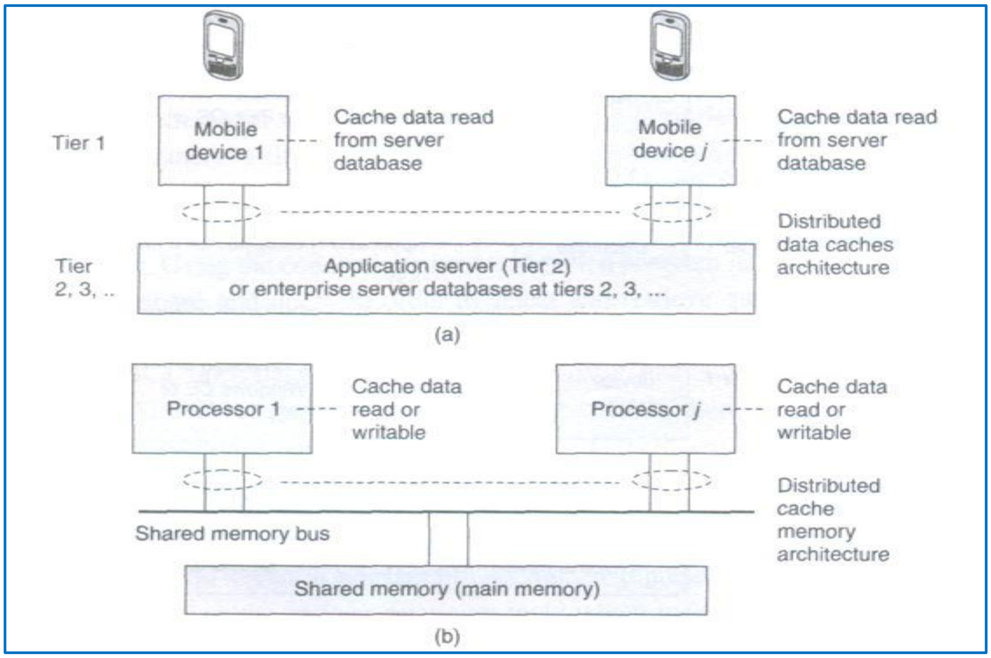
Database hoarding may be done at the application tier itself. The following figure shows a simple architecture in which a mobile device API directly retrieves the data from a database. It also shows another simple architecture in which a mobile device API directly retrieves the data from a database through a program, for ex: IBM DB2 Everyplace (DB2e).


Tier 1
Mobile device API
Mobile device database API; J2ME or BREW; XML database or other database and the cached databases from remote server; OS (Windows CE or PalmOS or Symbian V6)
Tier 1
Database
Application / database on a mobile device
(a)
Tier 1
Mobile device API
Mobile device database API, J2ME or BREW; XML database or other database OS (Windows CE or PalmOS or Symbian V6)
Application / database on a mobile device
Tier 1
DB2e at device
Device database

(a) API at mobile device sending queries and retrieving data from local database (Tier 1)
(b) API at mobile device retrieving data from database using DB2e (Tier 1)

Both the two architectures belong to the class of one-tier database architecture because the databases are specific to a mobile device, not meant to be distributed to multiple devices, not synchronized with the new updates, are stored at the device itself. Some examples are downloaded ringtones, music etc. **IBM DB2 Everyplace (DB2e)** is a relational database engine which has been designed to reside at the device. It supports J2ME and most mobile device operating systems. DB2e synchronizes with DB2 databases at the synchronization, application, or enterprise server

The database architecture shown below is for two-tier or multi-tier databases. Here, the databases reside at the remote servers and the copies of these databases are cached at the client tiers. This is known as client-server computing architecture.


Tier 1
Mobile device 1
Cache data read from server database
Mobile device *j*
Cache data read from server database
Distributed data caches architecture
Tier 2, 3, ..
Application server (Tier 2) or enterprise server databases at tiers 2, 3, ...
(a)
Processor 1
Cache data read or writable
Processor *j*
Cache data read or writable
Distributed cache memory architecture
Shared memory bus
Shared memory (main memory)
(b)

(a) Distributed data caches in mobile devices
(b) Similar architecture for a distributed cache memory in multiprocessor systems

A cache is a list or database of items or records stored at the device. Databases are hoarded at the application or enterprise tier, where the database server uses business logic and connectivity for retrieving the data and then transmitting it to the device. The server provides and updates local copies of the database at each mobile device connected to it. The computing API at the mobile device (first tier) uses the cached local copy. At first tier (tier 1), the API uses the cached data records using the computing architecture as explained above. From tier 2 or tier 3, the server retrieves and transmits the data records to tier 1 using business logic and synchronizes the local copies at the device. These local copies function as device caches.

The advantage of hoarding is that there is no access latency (delay in retrieving the queried record from the server over wireless mobile networks). The client device API has instantaneous data access to hoarded or cached data. After a device caches the data distributed by the server, the data is hoarded at the device. The disadvantage of hoarding is that the consistency of the cached data with the database at the server needs to be maintained.

Data Caching

Hoarded copies of the databases at the servers are distributed or transmitted to the mobile devices from the enterprise servers or application databases. The copies cached at the devices are equivalent to the cache memories at the processors in a multiprocessor system with a shared main memory and copies of the main memory data stored at different locations.

Cache Access Protocols: A client device caches the pushed (disseminated) data records from a server. Caching of the pushed data leads to a reduced access interval as compared to the pull (on-demand) mode of data fetching. Caching of data records can be-based on pushed 'hot records' (the most needed database records at the client device). Also, caching can be based on the ratio of two parameters—access probability (at the device) and pushing rates (from the server) for each record. This method is called cost-based data replacement or caching.

Pre-fetching: Pre-fetching is another alternative to caching of disseminated data. The process of pre-fetching entails requesting for and pulling records that may be required later. The client devicecan pre-fetch instead of caching from the pushed records keeping future needs in view. Pre- fetching reduces server load. Further, the cost of cache-misses can thus be reduced. The term 'cost of cache-misses' refers to the time taken in accessing a record at the server in case that record is not found in the device database when required by the deviceAPI.

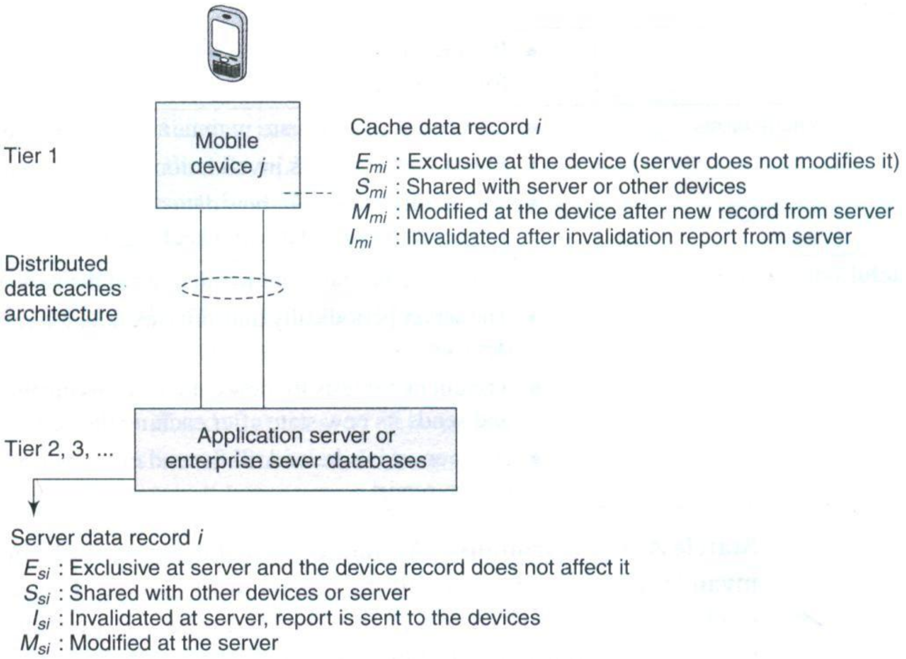
Caching Invalidation Mechanisms

A cached record at the client device may be invalidated. This may be due to expiry or modification of the record at the database server. Cache invalidation is a process by which a cached data item or record becomes invalid and thus unusable because of modification, expiry, or invalidation at another computing system or server. Cache invalidation mechanisms are used to synchronize the data at other processors whenever the cache-data is written (modified) by a processor in a multiprocessor system, cache invalidation mechanisms are also active in the case of mobile devices having distributed copies from the server.

A cache consists of several records. Each record is called a cache-line, copies of which can be stored at other devices or servers. The cache at the mobile devices or server databases at any given time can be assigned one of four possible tags indicating its state—*modified* (after rewriting), *exclusive, shared,* and *invalidated* (after expiry or when new data becomes available) at any given instance. These four states are indicated by the letters M, E, S, and I, respectively (MESI). The states indicated by the various tags are as follows:

a) The ***E*** tag indicates the *exclusive* state which means that the data record is for internal use and cannot be used by any other device or server.
b) The ***S*** tag indicates the *shared* state which indicates that the data record can be used by others.
c) The ***M*** tag indicates the *modified* state which means that the device cache
d) The ***I*** tag indicates the *invalidated state* which means that the server database no longer has a copy of the record which was shared and used for computations earlier.

The following figure shows the four possible states of a data record *i* at any instant in the server database and its copy at the cache of the mobile device *j*.


Tier 1
Mobile device *j*
Cache data record *i*
E_{mi} : Exclusive at the device (server does not modifies it)
S_{mi} : Shared with server or other devices
M_{mi} : Modified at the device after new record from server
I_{mi} : Invalidated after invalidation report from server
Distributed data caches architecture
Tier 2, 3, ...
Application server or enterprise sever databases
Server data record *i*
E_{si} : Exclusive at server and the device record does not affect it
S_{si} : Shared with other devices or server
I_{si} : Invalidated at server, report is sent to the devices
M_{si} : Modified at the server

Four possible states (*M, E, S,* or *I*) of a data record *I* at any instance at the server database and device *j* cache

Another important factor for cache maintenance in a mobile environment is *cache consistency* (also called *cache coherence).* This requires a mechanism to ensure that a database record is identical at the server as well as at the device caches and that only the valid cache records are used for computations.

UNIT- III

Cache invalidation mechanisms in mobile devices are triggered or initiated by the server. There are four possible invalidation mechanisms – Stateless asynchronous, stateless synchronous, stateful asynchronous and stateful synchronous.

Stateless Asynchronous: A stateless mechanism entails broadcasting of the invalidation of the cache to all the clients of the server. The server does not keep track of the records stored at the device caches. It just uniformly broadcasts invalidation reports to all clients irrespective of whether the device cache holds that particular record or not. The term 'asynchronous' indicates that the invalidation information for an item is sent as soon as its value changes. The server does not keep the information of the present state (whether E_{mi}, M_{mi}, S_{mi}, or I_{mi}) of a data-record in cache for broadcasting later. The server advertises the invalidation information only. The client can either request for a modified copy of the record or cache the relevant record when data is pushed from the server. The server advertises as and when the corresponding data-record at the server is invalidated and modified (deleted or replaced).

The advantage of the asynchronous approach is that there are no frequent, unnecessary transfers of data reports, thus making the mechanism more bandwidth efficient. The disadvantages of this approach are—(a) every client device gets an invalidation report, whether that client requires that copy or not and (b) client devices presume that as long as there is no invalidation report, the copy is valid for use in computations. Therefore, even when there is link failure, the devices may be using the invalidated data and the server is unaware of state changes at the clients after it sends the invalidation report.

Stateless Synchronous This is also a stateless mode, i.e., the server has no information regarding the present state of data records at the device caches and broadcasts to all client devices. However, unlike the asynchronous mechanism, here the server advertises invalidation information at periodic intervals as well as whenever the corresponding data-record at server is invalidated or modified. This method ensures synchronization because even if the in-between period report is not detected by the device due to a link failure, the device expects the period-end report of invalidation and if that is not received at the end of the period, then the device sends a request for the same (deleted or replaced). In case the client device does not get the periodic report due to link failure, it requests the server to send the report.

The advantage of the synchronous approach is that the client devices receive periodic information regarding invalidity (and thus validity) of the data caches. The periodic invalidation reports lead to greater reliability of cached data as update requests for invalid data can be sent to the server by the device-client. This also helps the server and devices maintain cache consistency through periodical exchanges. The disadvantages of this mode of cache invalidation are—(a) unnecessary transfers of data invalidation reports take place, (b) every client device gets an advertised invalidation report periodically, irrespective of whether that client has a copy of the invalidated data or not, and (c) during the period between two invalidation reports, the client devices assume that, as long as there is no invalidation report, the copy is valid for use in computations. Therefore, when there are link failures, the devices use data which has been invalidated in the in-between period and the server is unaware of state changes at the clients after it sends the invalidation report.

Stateful Asynchronous The stateful asynchronous mechanism is also referred to as the AS (asynchronous stateful) scheme. The term 'stateful' indicates that the cache invalidation reports are sent only to the affected client devices and not broadcasted to all. The server stores the information regarding the present state (a record I can have its state as E_{mi}, M_{mi}, S_{mi}, or I_{mi}) of each data-record at the client device caches. This state information is stored in the home location cache (HLC) at the server. The HLC is maintained by an HA (home agent) software. This is similar to the HLR at the MSC in a mobile network. The client device informs the HA of the state of each record to enable storage of the same at the HLC. The server transmits the invalidation information as and when the records are invalidated and it transmits only to the device-clients which are affected by the invalidation of data. Based on the invalidation information, these device-clients then request the server for new or modified data to replace the invalidated data. After the data records transmitted by the server modify the client device cache, the device sends information about the new state to the server so that the record of the cache-states at the server is also modified.

The advantage of the stateful asynchronous approach is that the server keeps track of the state of cached data at the client device. This enables the server to synchronize with the state of records at the device cache and keep the HLC updated. The stateful asynchronous mode is also advantageous in that only the affected clients receive the invalidation reports and other devices are not flooded with irrelevant reports. The disadvantage of the AS scheme is that the client devices presume that, as long as there is no invalidation report, the copy is valid for use in computations. Therefore, when there is a link failure, then the devices use invalidated data.

Stateful Synchronous: The server keeps the information of the present state (E_{mi}, M_{mi}, S_{mi}, or I_{mi}) of data-records at the client-caches. The server stores the cache record state at the home location cache (HLC) using the home agent (HA). The server transmits the invalidation information at periodic intervals to the clients and whenever the data-record relevant to the client is invalidated or modified (deleted or replaced) at the server. This method ensures synchronization because even if the in-between period report is not detected by the device due to a link failure, the device expects the period-end report of invalidation and if it is not received at the end of the period, then the device requests for the same.

The advantage of the stateful synchronous approach is that there are reports identifying invalidity (and thus, indirectly, of validity) of data caches at periodic intervals and that the server also periodically updates the client-cache states stored in the HLC. This enables to synchronize with the client device when invalid data gets modified and becomes valid. Moreover, since the invalidation report is sent periodically, if a device does not receive an invalidation report after the specified period of time, it can request the server to send the report. Each client can thus be periodically updated of any modifications at the server. When the invalidation report is not received after the designated period and a link failure is found at the device, the device does not use the invalidated data. Instead it requests the server for an invalidation update. The disadvantage of the stateful synchronous approach is the high bandwidth requirement to enable periodic transmission of invalidation reports to each device and updating requests from each client device.

Data Cache Maintenance in Mobile Environments

Assume that a device needs a data-record during an application. A request must be sent to the server for the data record (this mechanism is called pulling). The time taken for the application software to access a particular record is known as *access latency.* Caching and hoarding the record at the device reduces access latency to zero. Therefore, data cache maintenance is necessary in a mobile environment to overcome access latency.

Data cache inconsistency means that data records cached for applications are not invalidated at the device when modified at the server but not modified at the device. Data cache consistency can be maintained by the three methods given below:

I. *Cache invalidation mechanism* (server-initiated case): the server sends invalidation reports on invalidation of records (asynchronous) or at regular intervals (synchronous).

II. *Polling mechanism* (client-initiated case): Polling means checking from the server, the state of data record whether the record is in the valid, invalid, modified, or exclusive state. Each cached record copy is polled whenever required by the application software during computation. If the record is found to be modified or invalidated, then the device requests for the modified data and replaces the earlier cached record copy.

III. *Time-to-live mechanism* (client-initiated case): Each cached record is assigned a TTL (time-to-live). The TTL assignment is adaptive (adjustable) previous update intervals of that record. After the end of the TTL, the cached record copy is polled. If it is modified, then the device requests the server to replace the invalid cached record with the modified data. When TTL is set to 0, the TTL mechanism is equivalent to the polling mechanism.

Web Cache Maintenance in Mobile Environments

The mobile devices or their servers can be connected to a web server (e.g., traffic information server or train information server). Web cache at the device stores the web server data and maintains it in a manner similar to the cache maintenance for server data described above. If an application running at the device needs a data record from the web which is not at the web cache, then there is access latency. Web cache maintenance is necessary in a mobile environment to overcome access latency in downloading from websites due to disconnections. Web cache consistency can be maintained by two methods. These are:

I. Time-to-live (TTL) mechanism (client-initiated case): The method is identical to the one discussed for data cache maintenance.

II. Power-aware computing mechanism (client-initiated case): Each web cache maintained at the device can also store the CRC (cyclic redundancy check) bits. Assume that there are N cached bits and *n* CRC bits. N is much greater than *n.* Similarly at the server, *n* CRC bits are stored. As long as there is consistency between the server and device records, the CRC bits at both are identical. Whenever any of the records cached at the server is modified, the corresponding CRC bits at the server are also modified. After the TTL expires or on-demand for the web cache records by the client API, the cached record CRC is polled and obtained from the website server. If the *n* CRC bits at server are found to be modified and the change is found to be much higher than a given threshold (i.e., a significant change), then the modified part of the website hypertext or database is retrieved by the client device for use by the API. However, if the change is minor, then the API uses the previous cache. Since *N* » *n,* the power dissipated in the web cache maintenance method (in which invalidation reports and all invalidated record bits are transmitted) is much greater than that in the present method (in which the device polls for the significant change in the CRC bits at server and the records are transmitted only when there is a significant change in the CRC bits).

Client-Server Computing

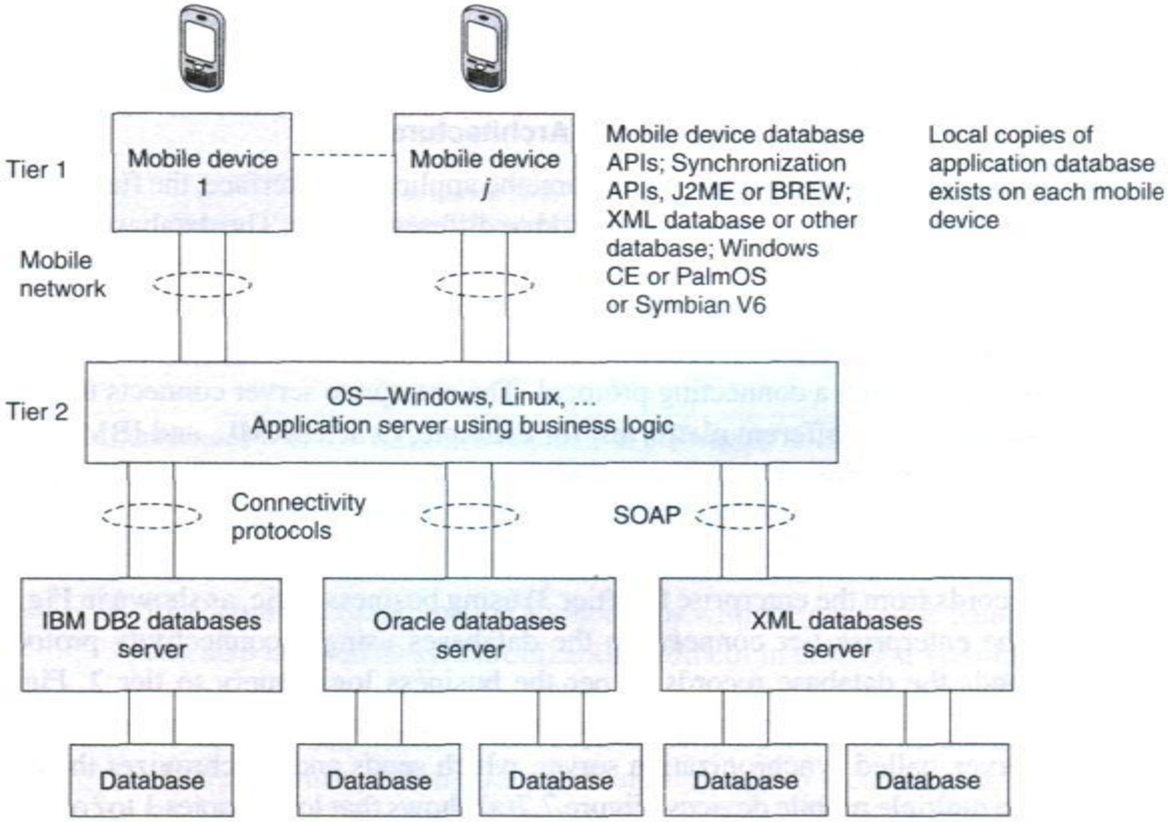
Client-server computing is a distributed computing architecture, in which there are two types of nodes, i.e., the clients and the servers. A server is defined as a computing system, which responds to requests from one or more clients. A client is defined as a computing system, which requests the server for a resource or for executing a task. The client can either access the data records at the server or it can cache these records at the client device. The data can be accessed either on client request or through broadcasts or distribution from the server.

The client and the server can be on the same computing system or on different computing systems. Client-server computing can have N-tier architecture *(N=* 1, 2 ...). When the client and the server are on the same computing system then the number of tiers, *N* = 1. When the client and the server are on different computing systems on the network, then *N* = 2. A command interchange protocol (e.g., HTTP) is used for obtaining the client requests at the server or the server responses at the client.

The following subsections describe client-server computing in 2, 3, or N-tier architectures. Each tier connects to the other with a connecting, synchronizing, data, or command interchange protocol.

UNIT- III

Two-tier Client-Server Architecture


Tier 1
Mobile device 1
Mobile device j
Mobile device database APIs; Synchronization APIs, J2ME or BREW; XML database or other database; Windows CE or PalmOS or Symbian V6
Local copies of application database exists on each mobile device
Mobile network
Tier 2
OS—Windows, Linux, ...
Application server using business logic
Connectivity protocols
SOAP
IBM DB2 databases server
Oracle databases server
XML databases server
Database
Database
Database
Database
Database

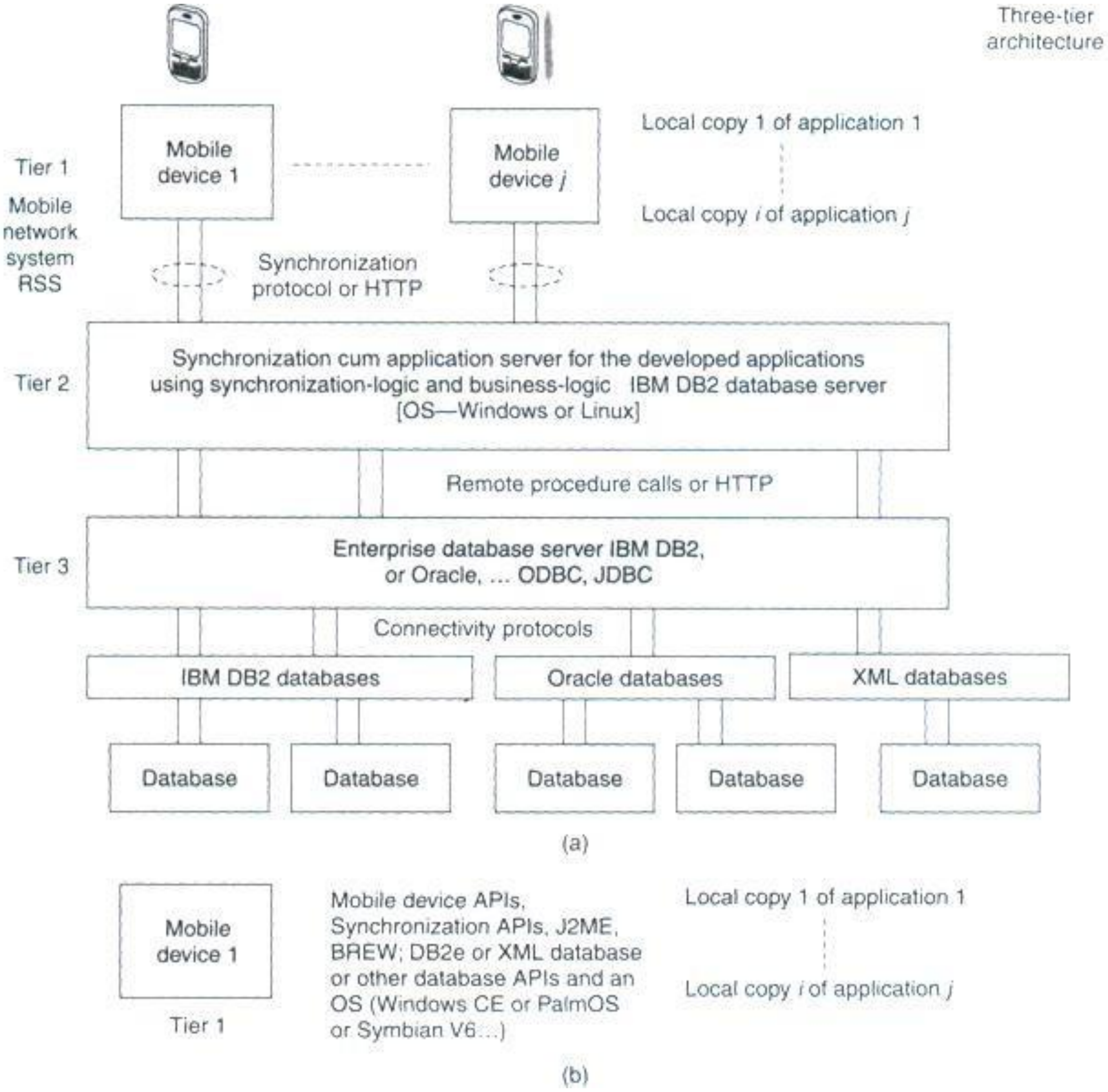
Multimedia file server in two-tier client-server computing architecture (local copies 1 to j of image and voice hoarding at the mobile devices)

The following figure shows the application server at the second tier. The data records are retrieved using business logic and a synchronization server in the application server synchronizes with the local copies at the mobile devices. Synchronization means that when copies of records at the server-end are modified, the copies cached at the client devices should also be accordingly modified. The APIs are designed independent of hardware and software platforms as far as possible as different devices may have different platforms.

Three-tier Client-Server Architecture

In a three-tier computing architecture, the application interface, the functional logic, and the database are maintained at three different layers. The database is associated with the enterprise server tier (tier 3) and only local copies of the database exist at mobile devices. The database connects to the enterprise server through a connecting protocol. The enterprise server connects the complete databases on different platforms, for example, Oracle, XML, and IBM DB2.

UNIT- III


Three-tier architecture
Tier 1
Mobile network system RSS
Mobile device 1
Mobile device *j*
Local copy 1 of application 1
Local copy *i* of application *j*
Synchronization protocol or HTTP
Tier 2
Synchronization cum application server for the developed applications using synchronization-logic and business-logic IBM DB2 database server [OS—Windows or Linux]
Remote procedure calls or HTTP
Tier 3
Enterprise database server IBM DB2, or Oracle, ... ODBC, JDBC
Connectivity protocols
IBM DB2 databases
Oracle databases
XML databases
Database
Database
Database
Database
Database
(a)
Mobile device 1
Tier 1
Mobile device APIs, Synchronization APIs, J2ME, BREW; DB2e or XML database or other database APIs and an OS (Windows CE or PalmOS or Symbian V6...)
Local copy 1 of application 1
Local copy *i* of application *j*
(b)

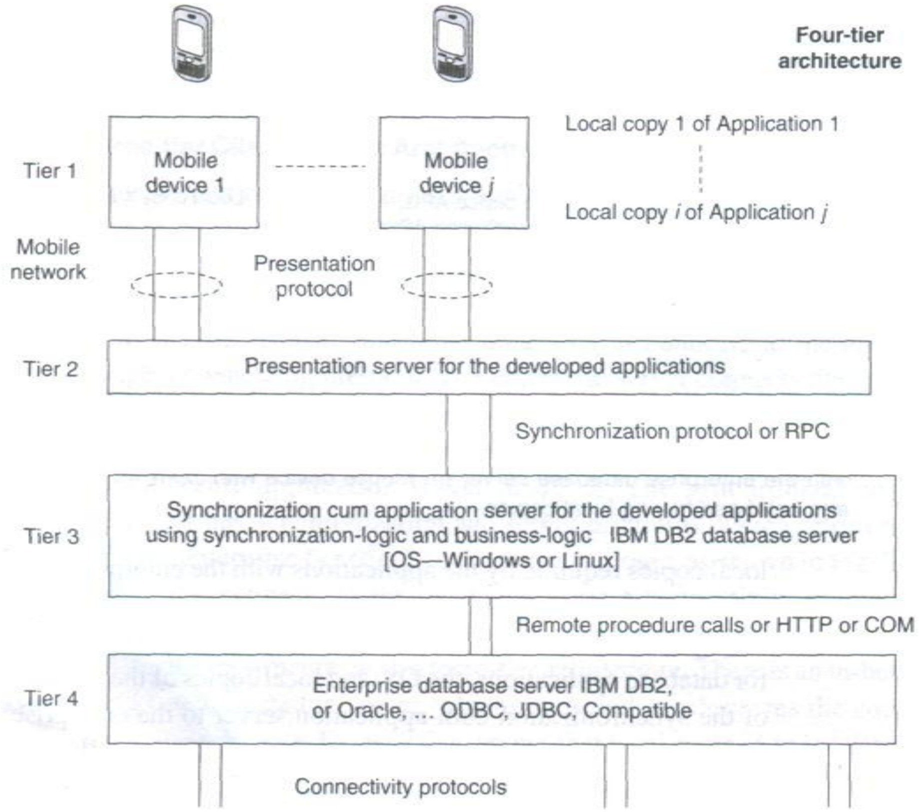
(a) Local copies 1 to j of database hoarded at the mobile devices using an enterprise database connection synchronization server, which synchronizes the required local copies for application with the enterprise database server (b) Mobile device with J2ME or BREW platform, APIs an OS and database having local copies

Data records at tier 3 are sent to tier 1 as shown in the figure through a synchronization-cum-application server at tier 2. The synchronization-cum-application server has synchronization and server programs, which retrieves data records from the enterprise tier (tier 3) using business logic. There is an in-between server, called synchronization server, which sends and synchronizes the copies at the multiple mobile devices. The figure shows that local copies 1 to j of databases are hoarded at the mobile devices for the applications 1 to j.

UNIT- III

N-tier Client-Server Architecture

When *N* is greater than 3, then the database is presented at the client through in-between layers. For example, the following figure shows a four-tier architecture in which a client device connects to a data-presentation server at tier 2.


Four-tier architecture
Tier 1
Mobile device 1
Mobile device *j*
Local copy 1 of Application 1
Local copy *i* of Application *j*
Mobile network
Presentation protocol
Tier 2
Presentation server for the developed applications
Synchronization protocol or RPC
Tier 3
Synchronization cum application server for the developed applications using synchronization-logic and business-logic IBM DB2 database server [OS—Windows or Linux]
Remote procedure calls or HTTP or COM
Tier 4
Enterprise database server IBM DB2, or Oracle, ... ODBC, JDBC, Compatible
Connectivity protocols

4-tier architecture in which a client device connects to a data-presentation server

The presentation server then connects to the application server tier 3. The application server can connect to the database using the connectivity protocol and to the multimedia server using Java or XML API at tier 4. The total number of tiers can be counted by adding 2 to the number of in-between servers between the database and the client device. The presentation, application, and enterprise servers can establish connectivity using RPC, Java RMI, JNDI, or HOP. These servers may also use HTTP or HTTPS in case the server at a *tier j* connects to tier j+1 using the Internet.

Client-Server Computing with Adaptation

The data formats of data transmitted from the synchronization server and those required for the device database and device APIs are different in different cases, there are two adapters at a mobile device—an adapter for standard data format for synchronization at the mobile device and another adapter for the backend database copy, which is in a different data format for the API at the mobile device. An adapter is software to get data in one format or data governed by one protocol and convert it to another format or to data governed by another protocol.

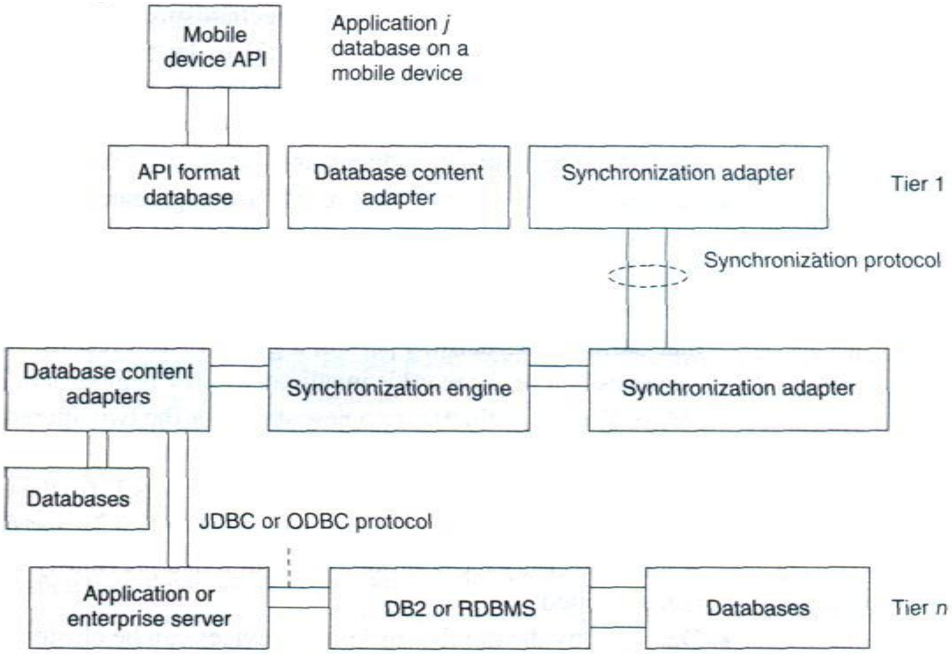


Figure shows an API, database, and adapters at a mobile device and the adapters at the synchronization, application, or enterprise servers. Here the adapters are an addition used for interchange between standard data formats and data formats for the API.

Context-aware Computing

The context of a mobile device represents the circumstances, situations, applications, or physical environment under which the device is being used. For example, let us assume that a mobile phone is operating in a busy, congested area. If the device is *aware* of the surrounding noises, then during the conversation, it can raise the speaker volume by itself and when the user leaves that area, the device can again reduce the volume. Also, if there is intermittent loss of connectivity during the conversation, the device can introduce background noises by itself so that the user does not feel discomfort due to intermittent periods of silence. This is one example in which the computing system is aware of the surrounding physical context in which the conversation is taking place.

A context-aware computing system is one which has user, device, and application interfaces such that, using these, the system remains aware of the past and present surrounding situations, circumstances, or actions such as the present mobile network, surrounding devices or systems, changes in the state of the connecting network, physical parameters such as present time of the day, presently remaining memory and battery power, presently available nearest connectivity, past sequence of actions of the device user, past sequence of application or applications, and previously cached data records, and takes these into account during computations.

UNIT- III

Context

The term 'context' refers to the interrelated conditions in which a collection of elements, records, components, or entities exists or occurs. Each message, data record, element, or entity has a meaning. But when these are considered along with the conditions that relate them to each other and to the environment, then they have a wider meaning. Understanding of the context in which a device is meant to operate, results in better, more efficient computing strategies.

Structural Context: To explain what is meant by structural context let us consider a few examples of records with structural arrangement. The fields *name, address, experience,* and *achievements* of a person have an individual meaning. However, when put together to form a resume, these fields acquire a significance beyond their individual meanings. This significance comes from the fact that they are now arranged in a structure which indicates an interrelationship between them. The structure of the resume includes the records and their interrelationship and thus defines a context for these records. Whereby, the records have a new meaning in the context of the resume (which is a structure). Contexts such as the context of the resume of an individual are called structural contexts. The context in such cases comes from the structure or format in which the records in a database are organized.

Consider another example, this time that of a line in a telephone directory. It has a sequence of records including a name, an address, and a 10-digit number. Each record has an individual meaning. But a collection of these records shows an interrelationship and thus defines a context, i.e., a telephone directory.

Implicit and Explicit Contexts Context may be implicit or explicit. Implicit context provides for omissions by leaving out unimportant details, takes independent world-views, and performs alterations in order to cope with incompatible protocols, interfaces, or APIs by transparently changing the messages. Implicit context uses history to examine call history, to manage omissions, or to determine recipients and performs contextual message alterations. Consider the context *'Contacts'* which has a set of contacts. The name, e-mail ID, and telephone number are implicit in a *contact* in the context Contacts. When a computing device uses a contact to call a number using a name record, the system takes independent view and uses the telephone number implicitly and deploys CDMA or GSM protocols for connecting to the mobile network implicitly. Context CDMA is implicit in defining the records 'Contact'. When a computing system uses a contact to send an e-mail using a name record, the use of the e-mail ID record is implicit to the system and the use of SMTP (simple mail transfer protocol) or other mail sending protocol is also implicit. Name gets automatically altered to e-mail ID when the context is sending of e-mail. The implicit context also copes with incompatible interfaces, for example, mail sending and receiving software handling data in different formats. Consider the context *document.* In *document* context, the contact or personal information is an extrinsic context. In context to processing of a document, the existence of document author contact information is extrinsic. The *contacts* context is imported into the *document* context to establish interrelationship between *document* and *contact.*

Context-aware Computing

Context-aware computing leads to application-aware computing. This is so because the APIs are part of the context (implicit or explicit contexts). For example, if context is a contact, the phone-talk application will adapt itself to use of the telephone number from the 'contact' and to the use of GSM or CDMA communication.

Use of context in computing helps in reducing possibility of errors. It helps in reducing the ambiguity in the action(s). It helps in deciding the expected system response on computations. For example, if *name* is input in personal biodata context, then the *address, experience,* and *achievements,* which correspond to that name, are also required for computations. This is because all four are related and needed in biodata context. When *name* is input in telephone directory context, then the *address* and phone number, which correspond to that name, are also required for computations. This is because all three are related in context to telephone directory. The *name* in two different contexts (personal biodata and telephone directory) during computations needs computations to perform different actions.

Context Types in Context-aware Computing

The five types of contexts that are important in context-aware computing are-physical context, computing context, user context, temporal context, and structural context.

- *Physical Context*: The context can be that of the physical environment. The parameters for defining a physical context are service disconnection, light level, noise level, and signal strength. For example, if there is service disconnection during a conversation, the mobile device can sense the change in the physical conditions and it interleaves background noise so that the listener does not feel the effects of the disconnection. Also, the mobile device can sense the light levels, so during daytime the display brightness is increased and during night time or in poor light conditions, the device display brightness is reduced. The physical context changes and the device display is adjusted accordingly.
- *Computing Context:* The context in a context-aware computing environment may also be computing context. Computing context is defined by interrelationships and conditions of the network connectivity protocol in use (Bluetooth, ZigBee, GSM, GPRS, or CDMA), bandwidth, and available resources. Examples of resources are keypad, display unit, printer, and cradle. A cradle is the unit on which the mobile device lies in order to connect to a computer in the vicinity. Consider a mobile device lying on a cradle. It discovers the computing context and uses ActiveSync to synchronize and download from the computer. When a mobile device lies in the

vicinity of a computer with a Bluetooth interface, it discovers another computing context resource and uses wireless Bluetooth for connecting to the computer. When it functions independently and connects to a mobile network, it discovers another computing context and uses a GSM, CDMA, GPRS, or EDGE connection. The response of the system is as per the computing context, i.e., the network connectivity protocol.

- *User Context:* The user context is defined user location, user profiles, and persons near the user. Reza B 'Far defines user-interfaces context states as follows—'within the realm of user interfaces, we can define context as the sum of the relationships between the user interface components, the condition of the user, the primary intent of the system, and all of the other elements that allow users and computing systems to communicate.
- *Temporal Context:* Temporal context defines the interrelation between time and the occurrence of an event or action. A group of interface components have an intrinsic or extrinsic temporal context. For example, assume that at an instant the user presses the switch for *dial* in a mobile device. At the next instant the device seeks a *number* as an input. Then user will consider it in the context of dialling and input the number to be dialled. Now, assume that at another time the user presses the switch to *add* a contact in the mobile device. The device again prompts the user to enter a *number* as an input. The user will consider it in context of the number to be added in the *contacts and* stored in the device for future use. The device then seeks the name of the contact as the input. Response of the system in such cases is as per the temporal context. The context for the VUI (voice user interface) elements also defines a temporal context (depending upon the instances and sequences in which these occur).
- *Structural Context:* Structural context defines a sequence and structure formed by the elements or records. Graphic user interface (GUI) elements have structural context. Structural context may also be extrinsic for some other type of context. Interrelation among the GUI elements depends on structural positions on the display screen. When time is the context, then the hour and minute elements.

Transaction Models

A transaction is the execution of interrelated instructions in a sequence for a specific operation on a database. Database transaction models must maintain data integrity and must enforce a set of rules called ACID rules. These rules are as follows:

❖ ***Atomicity****:* All operations of a transaction must be complete. In case, a transaction cannot be completed; it must be undone (rolled back). Operations in a transaction are assumed to be one indivisible unit (atomic unit).

- ***Consistency**:* A transaction must be such that it preserves the integrity constraints and follows the declared consistency rules for a given database. Consistency means the data is not in a contradictory state after the transaction.
- ***Isolation**:* If two transactions are carried out simultaneously, there should not be any interference between the two. Further, any intermediate results in a transaction should be invisible to any other transaction.
- ***Durability:*** After a transaction is completed, it must persist and cannot be aborted or discarded. For example, in a transaction entailing transfer of a balance from account A to account B, once the transfer is completed and finished there should be no roll back.

Consider a base class library included in Microsoft.NET. It has a set of computer software components called ADO.NET (ActiveX Data Objects in .NET). These can be used to access the data and data services including for access and modifying the data stored in relational database systems. The ADO.NET transaction model permits three transaction commands:

1. `BeginTransaction`**: It is used to begin a transaction. Any operation after** `BeginTransaction` **is assumed to be a part of the transaction till the** `CommitTransaction` **command or the** `RollbackTransaction` **command. An example of a command is as follows:**

   ```
   connectionA.open();
   ```

   ```
   transA = connectionA.BeginTransaction();
   ```

   Here `connectionA` **and** `transA` **are two distinct objects.**

2. `Commit`**: It is used to commit the transaction operations that were carried out after the** `BeginTransaction` **command and up to this command. An example of this is**

   ```
   transA.Commit();
   ```

   All statements between `BeginTransaction` **and** `commit` **must execute automatically.**

3. `Rollback`**: It is used to rollback the transaction in case an exception is generated after the** `BeginTransaction` **command is executed.**

A DBMS may provide for auto-commit mode. *Auto-commit mode* means the transaction finished automatically even if an error occurs in between.

Query Processing

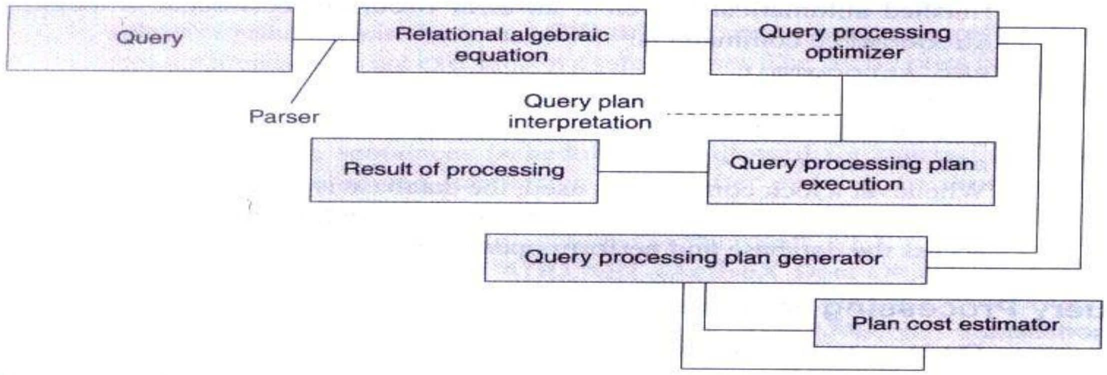
Query processing means making a correct as well as efficient execution strategy by *query decomposition* and *query-optimization*. A relational-algebraic equation defines a set of operations needed during query processing. Either of the two equivalent relational-algebraic equations given below can be used.

$$\pi_{\text{cName, cTelNum}} (\sigma_{\text{Contacts.firstChar = "R"}} (\sigma_{\text{Contacts.cTelNum = DialledNumbers.dTelNum}} (\text{Contacts}) \times \text{DialledNumbers}))$$

This means first select a column Contacts.cTelNum in a row in Contacts in which Contacts.cTelNum column equals a column DialledNumbers.dTelNum by crosschecking and matching the records of a column in Contacts with all the rows of DialledNumbers. Then in the second step select the row in which Contacts. firstChar = "R" and the selected cTelNum exists. Then in the third step project cName and CTelNum.

$$\pi_{\text{cName, cTelNum}} (\sigma_{\text{Contacts.firstChar = "R"} \wedge \text{Contacts.cTelNum = DialledNumbers.dTelNum}} (\text{Contacts} \times \text{DialledNumbers}))$$

This means that in first series of step, crosscheck all rows of Contacts and DialledNumbers and select, after AND operation, the rows in which Contacts.firstchar = "R" and Contacts.cTelNum = DialledNumbers.dTelNum. Then in the next step project cName and cTelNum form the selected records.


Query
Relational algebraic equation
Query processing optimizer
Parser
Query plan interpretation
Result of processing
Query processing plan execution
Query processing plan generator
Plan cost estimator

Query processing architecture

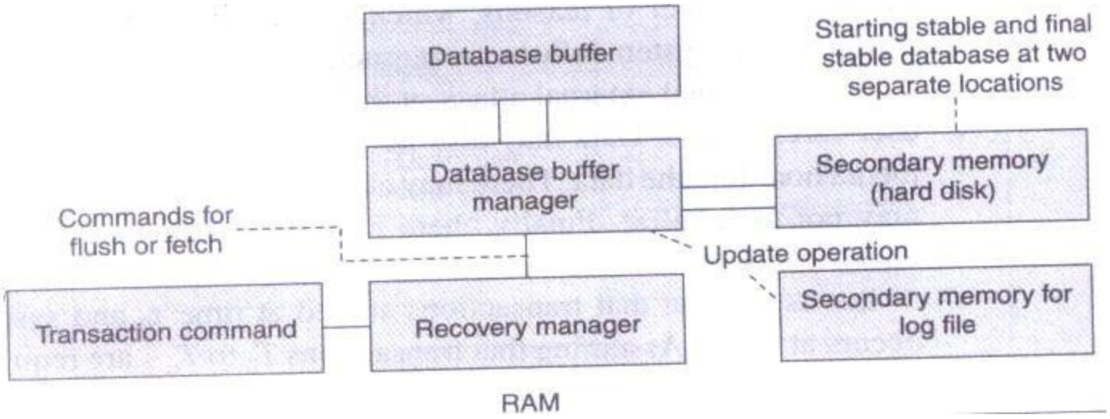
Π represents the projection operation, σ the *selection* operation, and Λ, the AND operation. It is clear that the second set of operations in query processing is less efficient than the first. Query decomposition of the first set gives efficiency. Decomposition is done by (i) analysis, (ii) conjunctive and disjunctive normalization, and (iii) semantic analysis.

Efficient processing of queries needs optimization of steps for query processing. Optimization can be based on cost (number of micro-operations in processing) by evaluating the costs of sets of equivalent expressions. Optimization can also be based on a heuristic approach consisting of the following steps: perform the selection steps and projection steps as early as possible and eliminate duplicate operations.

The query optimizer employs (a) query processing plan generator and (b) query processing cost estimator to provide an efficient plan for query processing.

Data Recovery Process

Data is non-recoverable in case of media failure, intentional attack on the database and transactions logging data, or physical media destruction. However, data recovery is possible in other cases. Figure below shows recovery management architecture. It uses a recovery manager, which ensures atomicity and durability. Atomicity ensures that an uncommitted but started transaction aborts on failure and aborted transactions are logged in log file. Durability ensures that a committed transaction is not affected by failure and is recovered. Stable state databases at the start and at the end of transactions reside in secondary storage. Transaction commands are sent to the recovery manager, which sends fetch commands to the database manager. The database manager processes the queries during the transaction and uses a database buffer. The recovery manager also sends the flush commands to transfer the committed transactions and database buffer data to the secondary storage. The recovery manager detects the results of operations. It recovers lost operations from the secondary storage. Recovery is by detecting the data lost during the transaction.


Database buffer
Starting stable and final stable database at two separate locations
Database buffer manager
Secondary memory (hard disk)
Commands for flush or fetch
Update operation
Secondary memory for log file
Transaction command
Recovery manager
RAM

Recovery Management Architecture

The recovery manager uses a log file, which logs actions in the following manner:

1. Each instruction for a transaction for update (insertion, deletion, replacement, and addition) must be logged.
2. Database read instructions are not logged
3. Log files are stored at a different storage medium.
4. Log entries are flushed out after the final stable state database is stored.

Each logged entry contains the following fields.

- transaction type (begin, commit, or rollback transaction)
- transaction ID
- operation-type
- object on which the operation is performed
- pre-operation and post-operation values of the object.

A procedure called the Aries algorithm is also used for recovering lost data. The basic steps of the algorithm are:

I. Analyse from last checkpoint and identify all dirty records (written again after operation restarted) in the buffer.
II. Redo all buffered operations logged in the update log to finish and make final page.
III. Undo all write operations and restore pre-transaction values.

The recovery models used in data recovery processes are as follows:

I. The *full recovery model* creates back up of the database and incremental backup of the changes. All transactions are logged from the last backup taken for the database.
II. The *bulk logged recovery model* entails logging and taking backup of bulk data record operations but not the full logging and backup. Size of bulk logging is kept to the minimum required. This improves performance. We can recover the database to the point of failure by restoring the database with the bulk transaction log file backup. This is unlike the full recovery model in which all operations are logged.
III. The *simple recovery model* prepares full backups but the incremental changes are not logged. We can recover the database to the most recent backup of the given database.