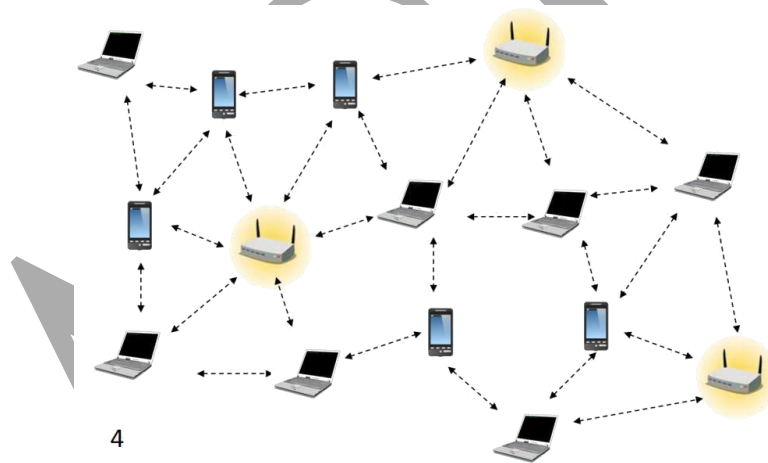


Mobile Ad hoc Networks (MANETs) : Introduction, Applications & Challenges of a MANET, Routing, Classification of Routing Algorithms, Algorithms such as DSR, AODV, DSDV, etc. , security in MANET's, Mobile Agents, Service Discovery.

Mobile Ad hoc NETWORKS (MANETs) are wireless networks which are characterized by dynamic topologies and no fixed infrastructure. Each node in a MANET is a computer that may be required to act as both a host and a router and, as much, may be required to forward packets between nodes which cannot directly communicate with one another. Each MANET node has much smaller frequency spectrum requirements than that for a node in a fixed infrastructure network. A MANET is an autonomous collection of mobile users that communicate over relatively bandwidth constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. The network is decentralized, where all network activity including discovering the topology and delivering messages must be executed by the nodes themselves, i.e., routing functionality will be incorporated into mobile nodes.



A mobile ad hoc network is a collection of wireless nodes that can dynamically be set up anywhere and anytime without using any pre-existing fixed network infrastructure.

MANET- Characteristics

- Dynamic network topology
- Bandwidth constraints and variable link capacity
- Energy constrained nodes
- Multi-hop communications
- Limited security
- Autonomous terminal
- Distributed operation
- Light-weight terminals

Need for Ad Hoc Networks

- ❖ Setting up of fixed access points and backbone infrastructure is not always viable
 - Infrastructure may not be present in a disaster area or war zone
 - Infrastructure may not be practical for short-range radios; Bluetooth (range ~ 10m)
- ❖ Ad hoc networks:
 - Do not need backbone infrastructure support
 - Are easy to deploy
 - Useful when infrastructure is absent, destroyed or impractical

Properties of MANETs

- MANET enables fast establishment of networks. When a new network is to be established, the only requirement is to provide a new set of nodes with limited wireless communication range. A node has limited capability, that is, it can connect only to the nodes which are nearby. Hence it consumes limited power.
- A MANET node has the ability to discover a neighboring node and service. Using a service discovery protocol, a node discovers the service of a nearby node and communicates to a remote node in the MANET.
- MANET nodes have peer-to-peer connectivity among themselves.
- MANET nodes have independent computational, switching (or routing), and communication capabilities.
- The wireless connectivity range in MANETs includes only nearest node connectivity.
- The failure of an intermediate node results in greater latency in communicating with the remote server.
- Limited bandwidth available between two intermediate nodes becomes a constraint for the MANET. The node may have limited power and thus computations need to be energy-efficient.
- There is no access-point requirement in MANET. Only selected access points are provided for connection to other networks or other MANETs.
- MANET nodes can be the iPods, Palm handheld computers, Smartphones, PCs, smart labels, smart sensors, and automobile-embedded systems\
- MANET nodes can use different protocols, for example, IrDA, Bluetooth, ZigBee, 802.11, GSM, and TCP/IP. MANET node performs data caching, saving, and aggregation.
- MANET mobile device nodes interact seamlessly when they move with the nearby wireless nodes, sensor nodes, and embedded devices in automobiles so that the seamless connectivity is maintained between the devices.

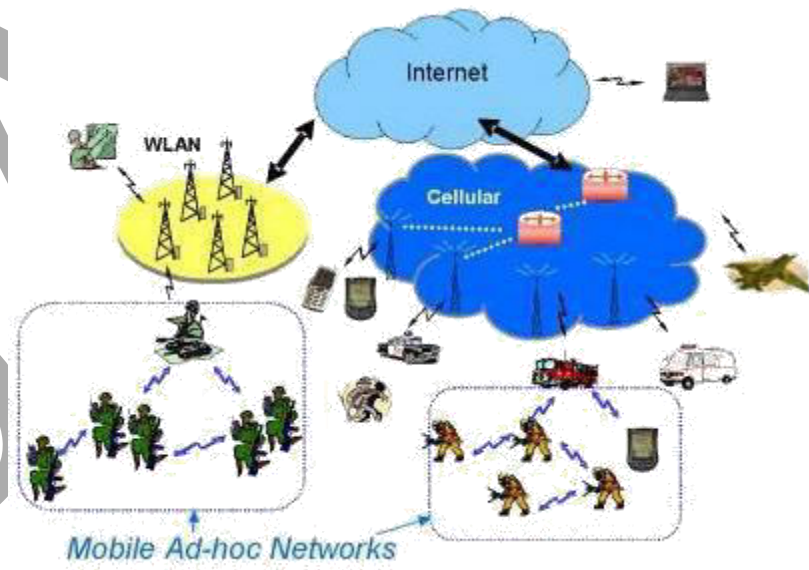
MANET challenges

To design a good wireless ad hoc network, various challenges have to be taken into account:

-
Dynamic Topology: Nodes are free to move in an arbitrary fashion resulting in the topology changing arbitrarily. This characteristic demands dynamic configuration of the network.
-
Limited security: Wireless networks are vulnerable to attack. Mobile ad hoc networks are more vulnerable as by design any node should be able to join or leave the network at any time. This requires flexibility and higher openness.
-
Limited Bandwidth: Wireless networks in general are bandwidth limited. In an ad hoc network, it is all the more so because there is no backbone to handle or multiplex higher bandwidth
-
Routing: Routing in a mobile ad hoc network is complex. This depends on many factors, including finding the routing path, selection of routers, topology, protocol etc.

Applications of MANETS

The set of applications for MANETs is diverse, ranging from small, static networks that are constrained by power sources, to large-scale, mobile, highly dynamic networks. The design of network protocols for these networks is a complex issue. Regardless of the application, MANETs need efficient distributed algorithms to determine network organization, link scheduling, and routing. Some of the main application areas of MANET's are:



- **Military battlefield**– soldiers, tanks, planes. Ad- hoc networking would allow the military to take advantage of commonplace network technology to maintain an information network between the soldiers, vehicles, and military information headquarters.

- **Sensor networks** – to monitor environmental conditions over a large area
- **Local level** – Ad hoc networks can autonomously link an instant and temporary multimedia network using notebook computers or palmtop computers to spread and share information among participants at e.g. conference or classroom. Another appropriate local level application might be in home networks where devices can communicate directly to exchange information.
- **Personal Area Network (PAN)** – pervasive computing i.e. to provide flexible connectivity between personal electronic devices or home appliances. Short-range MANET can simplify the intercommunication between various mobile devices (such as a PDA, a laptop, and a cellular phone). Tedious wired cables are replaced with wireless connections. Such an ad hoc network can also extend the access to the Internet or other networks by mechanisms e.g. Wireless LAN (WLAN), GPRS, and UMTS.
- **Vehicular Ad hoc Networks** – intelligent transportation i.e. to enable real time vehicle monitoring and adaptive traffic control
- **Civilian environments** – taxi cab network, meeting rooms, sports stadiums, boats, small aircraft
- **Emergency-operations** – search and rescue, policing and fire fighting and to provide connectivity between distant devices where the network infrastructure is unavailable. Ad hoc can be used in emergency/rescue operations for disaster relief efforts, e.g. in fire, flood, or earthquake. Emergency rescue operations must take place where non-existing or damaged communications infrastructure and rapid deployment of a communication network is needed. Information is relayed from one rescue team member to another over a small hand held.

Routing in MANET's

Routing in Mobile Ad hoc networks is an important issue as these networks do not have fixed infrastructure and routing requires distributed and cooperative actions from all nodes in the network. MANET's provide point to point routing similar to Internet routing. The major difference between routing in MANET and regular internet is the route discovery mechanism. Internet routing protocols such as RIP or OSPF have relatively long converge times, which is acceptable for a wired network that has infrequent topology changes. However, a MANET has a rapid topology changes due to node mobility making the traditional internet routing protocols inappropriate. MANET-specific routing protocols have been proposed, that handle topology changes well, but they have large control overhead and are not scalable for large networks. Another major difference in the routing is the network address. In internet routing, the network address (IP address) is hierarchical containing a network ID and a computer ID on that network. In contrast, for most MANET's the network address is simply an ID of the node in the network and is not hierarchical. The routing protocol must use the entire address to decide the next hop.

Some of the fundamental differences between wired networks & ad-hoc networks are:

- Asymmetric links: - Routing information collected for one direction is of no use for the other direction. Many routing algorithms for wired networks rely on a symmetric scenario.
- Redundant links: - In wired networks, some redundancy is present to survive link failures and this redundancy is controlled by a network administrator. In ad-hoc networks, nobody controls redundancy resulting in many redundant links up to the extreme of a complete meshed topology.
- Interference: - In wired networks, links exist only where a wire exists, and connections are planned by network administrators. But, in ad-hoc networks links come and go depending on transmission characteristics, one transmission might interfere with another and nodes might overhear the transmission of other nodes.
- Dynamic topology: - The mobile nodes might move in an arbitrary manner or medium characteristics might change. This result in frequent changes in topology, so snapshots are valid only for a very short period of time. So, in ad-hoc networks, routing tables must somehow reflect these frequent changes in topology and routing algorithms have to be adopted.

Summary of the difficulties faced for routing in ad-hoc networks

- Traditional routing algorithms known from wired networks will not work efficiently or fail completely. These algorithms have not been designed with a highly dynamic topology, asymmetric links, or interference in mind.
- Routing in wireless ad-hoc networks cannot rely on layer three knowledge alone. Information from lower layers concerning connectivity or interference can help routing algorithms to find a good path.
- Centralized approaches will not really work, because it takes too long to collect the current status and disseminate it again. Within this time the topology has already changed.
- Many nodes need routing capabilities. While there might be some without, at least one router has to be within the range of each node. Algorithms have to consider the limited battery power of these nodes.
- The notion of a connection with certain characteristics cannot work properly. Ad-hoc networks will be connectionless, because it is not possible to maintain a connection in a fast changing environment and to forward data following this connection. Nodes have to make local decisions for forwarding and send packets roughly toward the final destination.
- A last alternative to forward a packet across an unknown topology is flooding. This approach always works if the load is low, but it is very inefficient. A hop counter is needed in each packet to avoid looping, and the diameter of the ad-hoc network.

Types of MANET Routing Algorithms:

1. Based on the information used to build routing tables :
 - Shortest distance algorithms: algorithms that use distance information to build routing tables.
 - Link state algorithms: algorithms that use connectivity information to build a topology graph that is used to build routing tables.
2. Based on when routing tables are built:
 - Proactive algorithms: maintain routes to destinations even if they are not needed. Some of the examples are Destination Sequenced Distance Vector (DSDV), Wireless Routing Algorithm (WRP), Global State Routing (GSR), Source-tree Adaptive Routing (STAR), Cluster-Head Gateway Switch Routing (CGSR), Topology Broadcast Reverse Path Forwarding (TBRPF), Optimized Link State Routing (OLSR) etc.
 - ❖ Always maintain routes:- Little or no delay for route determination
 - ❖ Consume bandwidth to keep routes up-to-date
 - ❖ Maintain routes which may never be used
 - ❖ Advantages: low route latency, State information, QoS guarantee related to connection set-up or other real-time requirements
 - ❖ Disadvantages: high overhead (periodic updates) and route repair depends on update frequency
 - Reactive algorithms: maintain routes to destinations only when they are needed. Examples are Dynamic Source Routing (DSR), Ad hoc-On demand distance Vector (AODV), Temporally ordered Routing Algorithm (TORA), Associativity-Based Routing (ABR) etc
 - ❖ only obtain route information when needed
 - ❖ Advantages: no overhead from periodic update, scalability as long as there is only light traffic and low mobility.
 - ❖ Disadvantages: high route latency, route caching can reduce latency
 - Hybrid algorithms: maintain routes to nearby nodes even if they are not needed and maintain routes to far away nodes only when needed. Example is Zone Routing Protocol (ZRP).

Which approach achieves a better trade-off depends on the traffic and mobility patterns.

Destination sequence distance vector (DSDV)

Destination sequence distance vector (DSDV) routing is an example of proactive algorithms and an enhancement to distance vector routing for ad-hoc networks. Distance vector routing is used as routing information protocol (RIP) in wired networks. It performs extremely poorly with certain network changes due to the count-to-infinity problem. Each node exchanges its neighbor table periodically with its neighbors. Changes at one node in the network propagate slowly through the network. The strategies to avoid this problem which are used in fixed networks do not help in the case of wireless ad-hoc networks, due to the rapidly changing topology. This might create loops or unreachable regions within the network.

DSDV adds the concept of sequence numbers to the distance vector algorithm. Each routing advertisement comes with a sequence number. Within ad-hoc networks, advertisements may propagate along many paths. Sequence numbers help to apply the advertisements in correct order. This avoids the loops that are likely with the unchanged distance vector algorithm.

Each node maintains a routing table which stores next hop, cost metric towards each destination and a **sequence number that is created by the destination itself**. Each node periodically forwards routing table to neighbors. Each node **increments and appends its sequence number** when sending its local routing table. Each route is tagged with a sequence number; routes with greater sequence numbers are preferred. Each node advertises a monotonically increasing even sequence number for itself. When a node decides that a route is **broken**, it increments the sequence number of the route and advertises it with infinite metric. Destination advertises new sequence number.

When X receives information from Y about a route to Z,



- ❖ Let destination sequence number for Z at X be $S(X)$, $S(Y)$ is sent from Y
- ❖ If $S(X) > S(Y)$, then X ignores the routing information received from Y
- ❖ If $S(X) = S(Y)$, and cost of going through Y is smaller than the route known to X, then X sets Y as the next hop to Z
- ❖ If $S(X) < S(Y)$, then X sets Y as the next hop to Z, and $S(X)$ is updated to equal $S(Y)$

Besides being loop-free at all times, DSDV has low memory requirements and a quick convergence via triggered updates. Disadvantages of DSDV are, large routing overhead, usage of only bidirectional links and suffers from count to infinity problem.

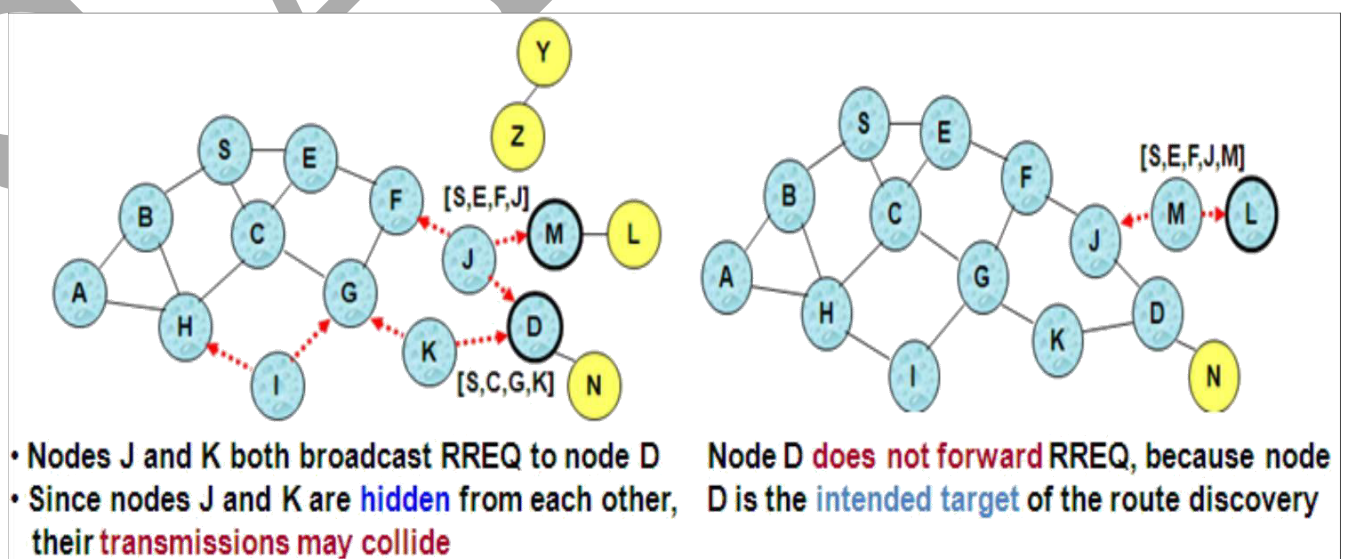
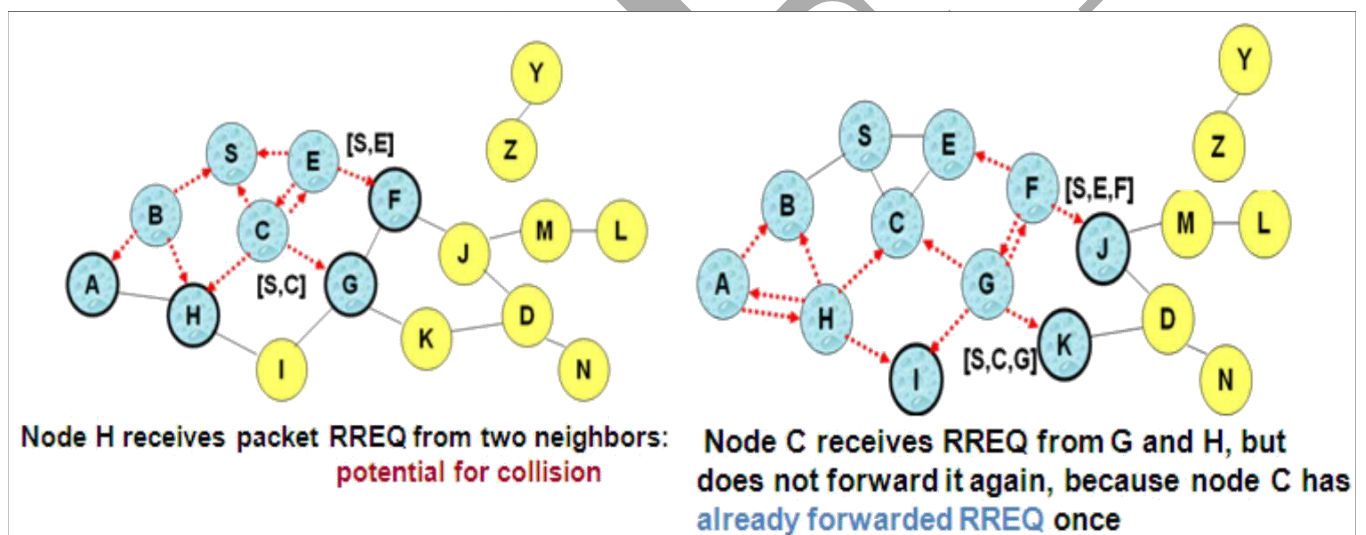
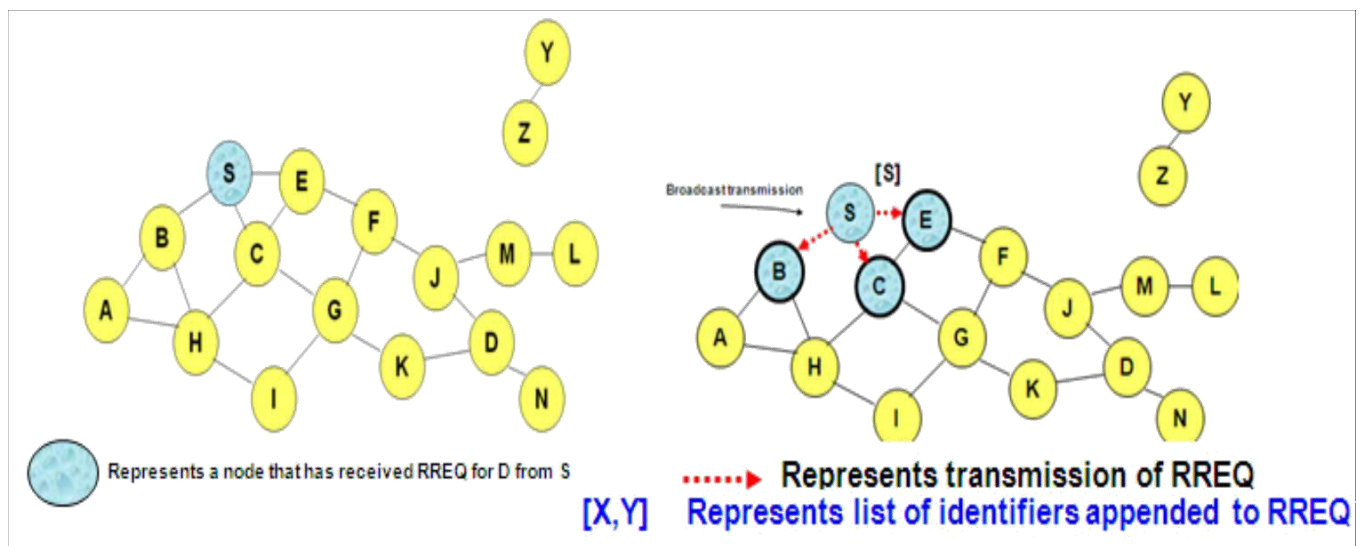
Dynamic Source Routing

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use.

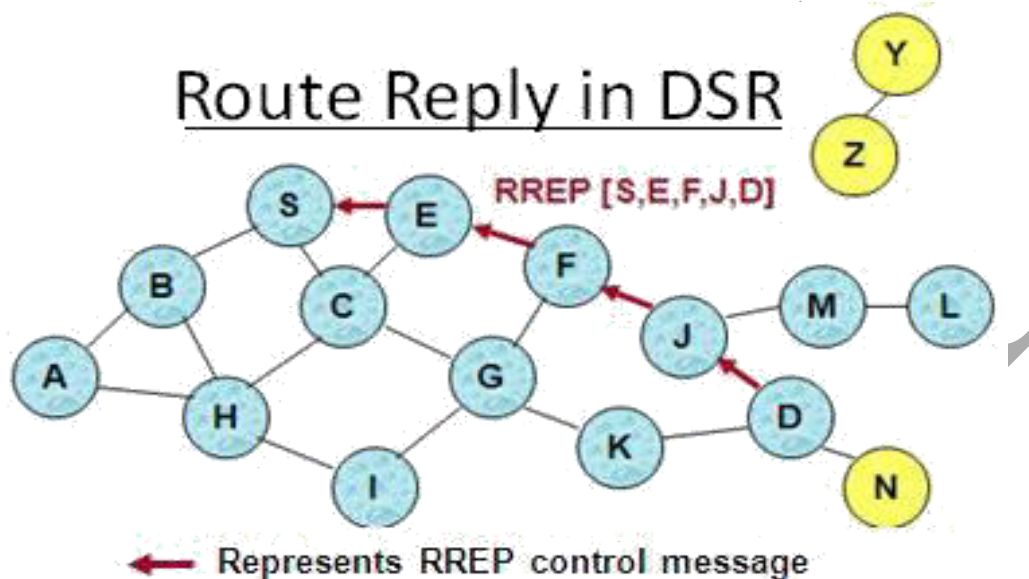
Route discovery. If the source does not have a route to the destination in its route cache, it broadcasts a route request (RREQ) message specifying the destination node for which the route is requested. The RREQ message includes a route record which specifies the sequence of nodes traversed by the message. When an intermediate node receives a RREQ, it checks to see if it is already in the route record. If it is, it drops the message. This is done to prevent routing loops. If the intermediate node had received the RREQ before, then it also drops the message. The intermediate node forwards the RREQ to the next hop according to the route specified in the header. When the destination receives the RREQ, it sends back a route reply message. If the destination has a route to the source in its route cache, then it can send a route response (RREP) message along this route. Otherwise, the RREP message can be sent along the reverse route back to the source. Intermediate nodes may also use their route cache to reply to RREQs. If an intermediate node has a route to the destination in its cache, then it can append the route to the route record in the RREQ, and send an RREP back to the source containing this route. This can help limit flooding of the RREQ. However, if the cached route is out-of-date, it can result in the source receiving stale routes.

Route maintenance. When a node detects a broken link while trying to forward a packet to the next hop, it sends a route error (RERR) message back to the source containing the link in error. When an RERR message is received, all routes containing the link in error are deleted at that node.

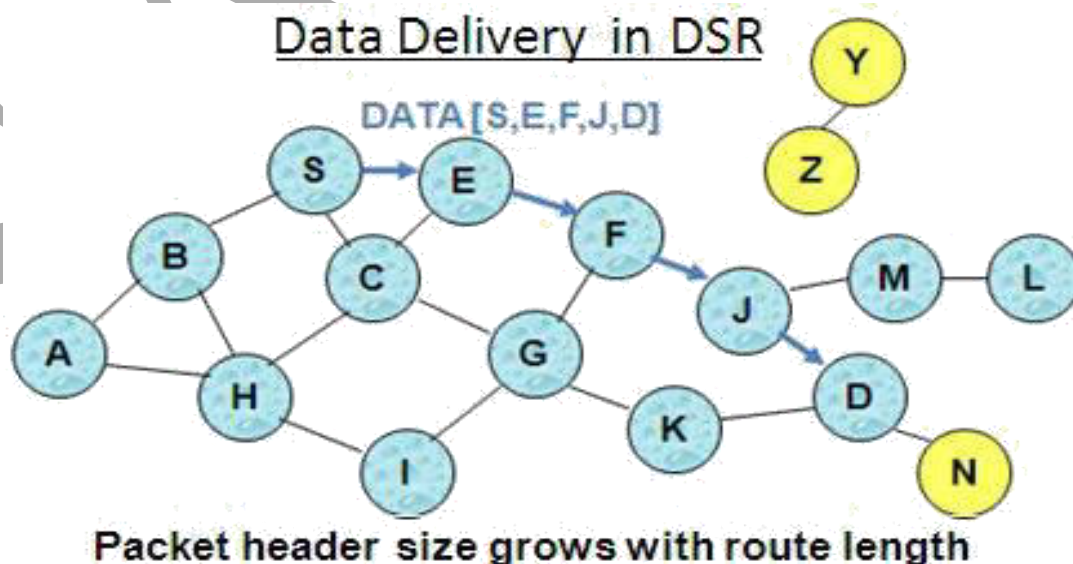
As an example, consider the following MANET, where a node S wants to send a packet to D, but does not know the route to D. So, it initiates a route discovery. Source node S floods Route Request (RREQ). Each node appends its own identifier when forwarding RREQ as shown below.



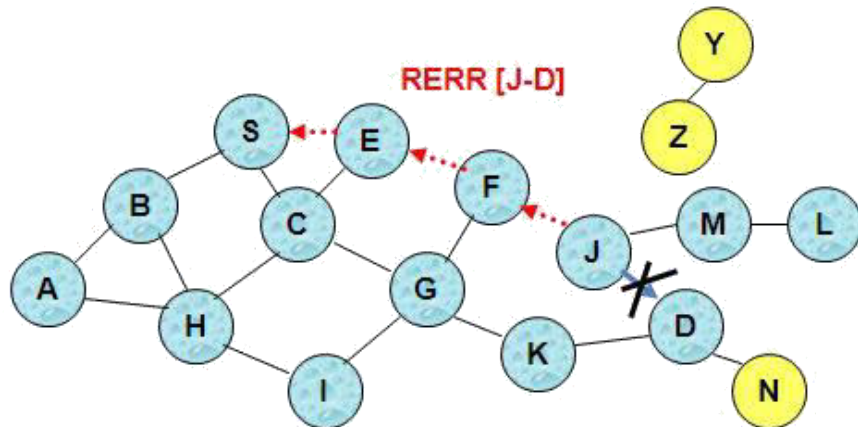
Destination D on receiving the first RREQ, sends a Route Reply (RREP). RREP is sent on a route obtained by reversing the route appended to received RREQ. RREP includes the route from S to D on which RREQ was received by node D.



Route Reply can be sent by reversing the route in Route Request (RREQ) only if links are guaranteed to be bi-directional. If Unidirectional (asymmetric) links are allowed, then RREP may need a route discovery from S to D. Node S on receiving RREP, caches the route included in the RREP. When node S sends a data packet to D, the entire route is included in the packet header {hence the name source routing}. Intermediate nodes use the source route included in a packet to determine to whom a packet should be forwarded.



J sends a route error to S along route J-F-E-S when its attempt to forward the data packet S (with route SEFJD) on J-D fails. Nodes hearing RERR update their route cache to remove link J-D



Advantages of DSR:

- Routes maintained only between nodes who need to communicate-- reduces overhead of route maintenance
- Route caching can further reduce route discovery overhead
- A single route discovery may yield many routes to the destination, due to intermediate nodes replying from local caches

Disadvantages of DSR:

- Packet header size grows with route length due to source routing
- Flood of route requests may potentially reach all nodes in the network
- Care must be taken to avoid collisions between route requests propagated by neighboring nodes -- insertion of random delays before forwarding RREQ
- Increased contention if too many route replies come back due to nodes replying using their local cache-- Route Reply *Storm* problem. Reply storm may be eased by preventing a node from sending RREP if it hears another RREP with a shorter route
- An intermediate node may send Route Reply using a stale cached route, thus polluting other caches

An optimization for DSR can be done called as Route Caching. Each node caches a new route it learns by *any means*. In the above example, When node S finds route [S,E,F,J,D] to node D, node S also learns route [S,E,F] to node F. When node K receives Route Request [S,C,G] destined for node, node K learns route [K,G,C,S] to node S. When node F forwards Route Reply RREP [S,E,F,J,D], node F learns route [F,J,D] to node D. When node E forwards Data [S,E,F,J,D] it learns route [E,F,J,D] to node D. A node may also learn a route when it overhears Data packets. Usage of Route cache can speed up route discovery and can also reduce propagation of route

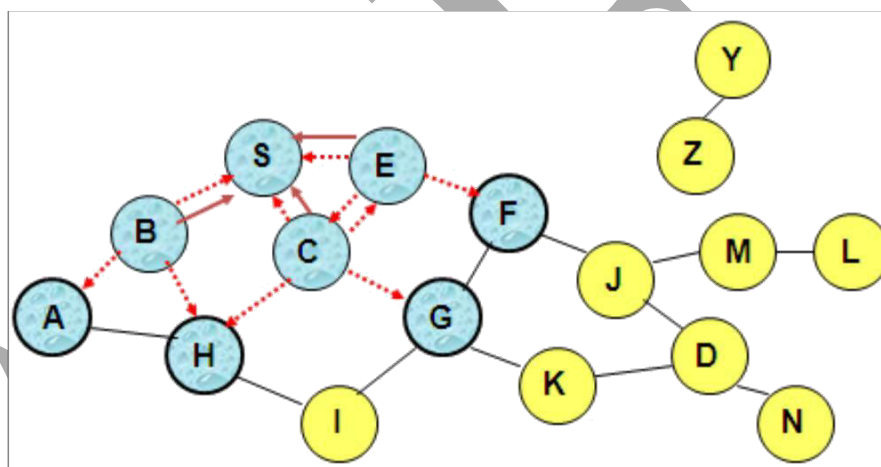
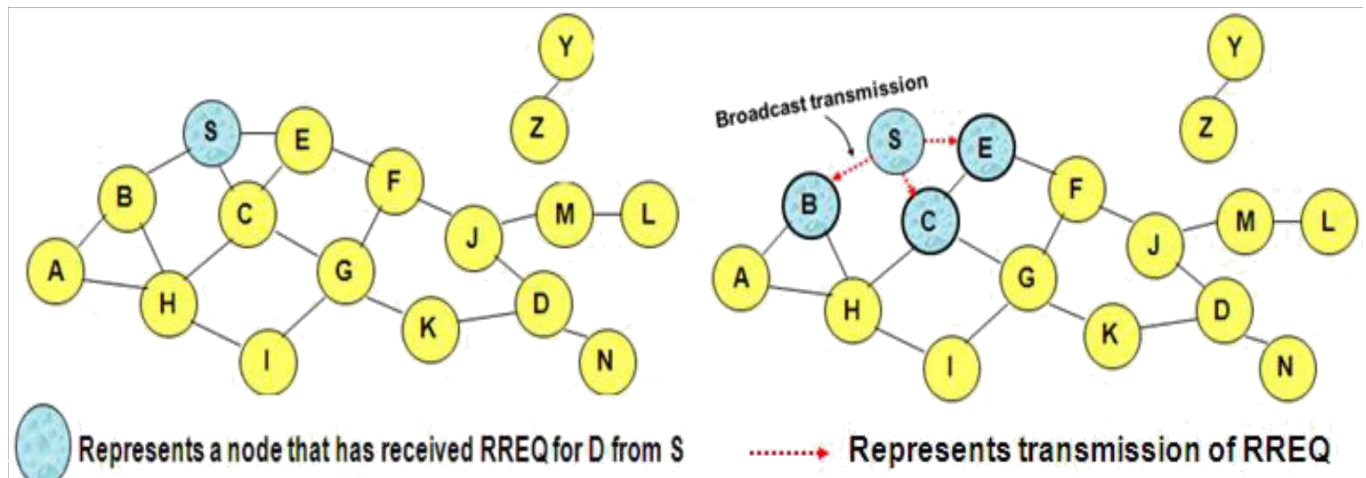
requests. The disadvantages are, stale caches can adversely affect performance. With passage of time and host mobility, cached routes may become invalid.

Ad Hoc On-Demand Distance Vector Routing (AODV)

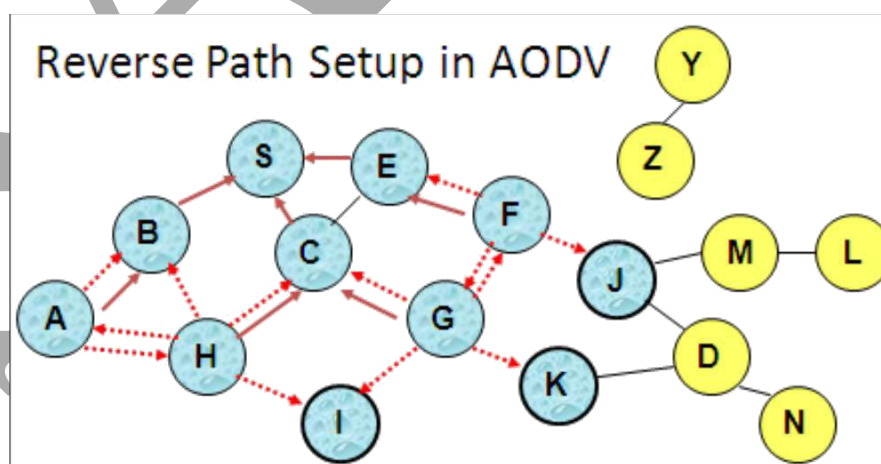
AODV is another reactive protocol as it reacts to changes and maintains only the active routes in the caches or tables for a pre-specified expiration time. Distance vector means a set of distant nodes, which defines the path to destination. AODV can be considered as a descendant of DSR and DSDV algorithms. It uses the same route discovery mechanism used by DSR. DSR includes source routes in packet headers and resulting large headers can sometimes degrade performance, particularly when data contents of a packet are small. AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes. AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate. However, as opposed to DSR, which uses source routing, AODV uses hop-by-hop routing by maintaining routing table entries at intermediate nodes.

Route Discovery. The route discovery process is initiated when a source needs a route to a destination and it does not have a route in its routing table. To initiate route discovery, the source floods the network with a RREQ packet specifying the destination for which the route is requested. When a node receives an RREQ packet, it checks to see whether it is the destination or whether it has a route to the destination. If either case is true, the node generates an RREP packet, which is sent back to the source along the reverse path. Each node along the reverse path sets up a forward pointer to the node it received the RREP from. This sets up a forward path from the source to the destination. If the node is not the destination and does not have a route to the destination, it rebroadcasts the RREQ packet. At intermediate nodes duplicate RREQ packets are discarded. When the source node receives the first RREP, it can begin sending data to the destination. To determine the relative degree out-of-datedness of routes, each entry in the node routing table and all RREQ and RREP packets are tagged with a destination sequence number. A larger destination sequence number indicates a more current (or more recent) route. Upon receiving an RREQ or RREP packet, a node updates its routing information to set up the reverse or forward path, respectively, only if the route contained in the RREQ or RREP packet is more current than its own route.

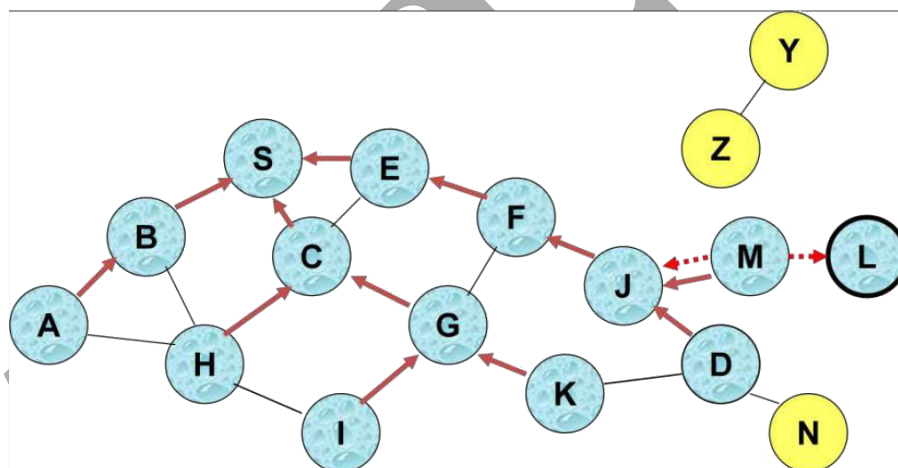
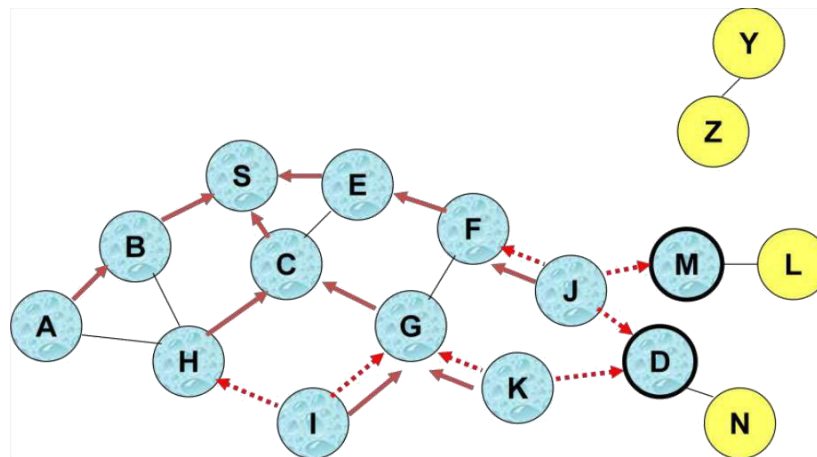
Route Maintenance. When a node detects a broken link while attempting to forward a packet to the next hop, it generates a RERR packet that is sent to all sources using the broken link. The RERR packet erases all routes using the link along the way. If a source receives a RERR packet and a route to the destination is still required, it initiates a new route discovery process. Routes are also deleted from the routing table if they are unused for a certain amount of time.



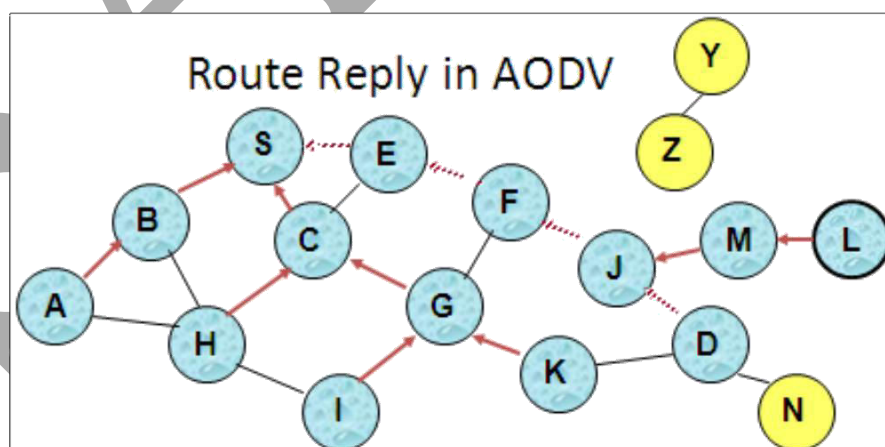
Represents links on Reverse Path



Node C receives RREQ from G and H, but does not forward it again, because node C has **already forwarded RREQ** once

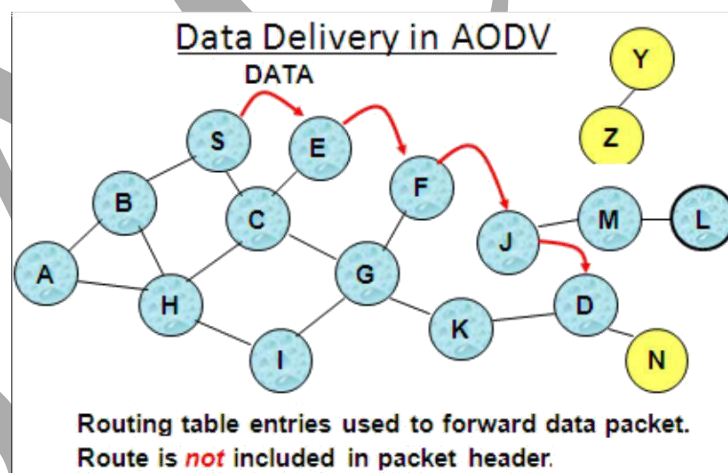
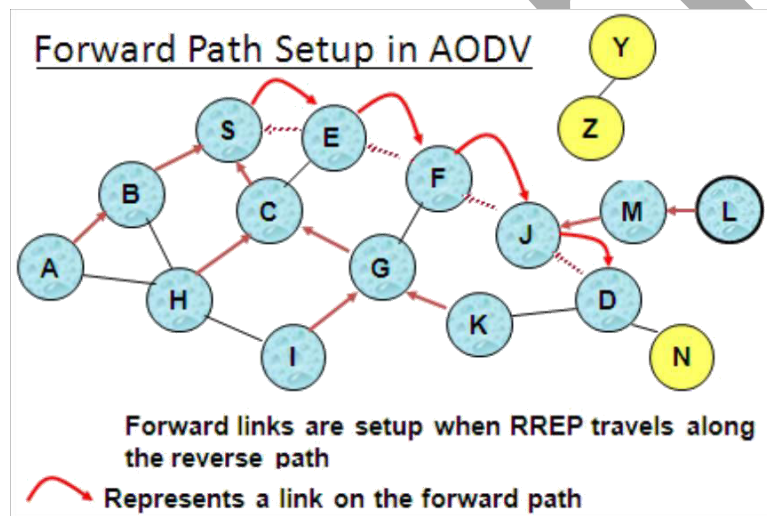


Node D **does not forward** RREQ, because node D is the **intended target** of the RREQ



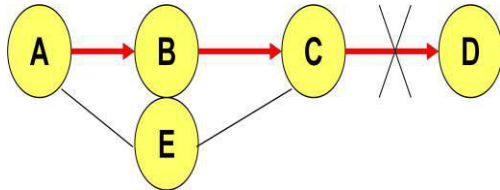
..... Represents links on path taken by RREP

An intermediate node (not the destination) may also send a Route Reply (RREP) provided that it knows a more recent path than the one previously known to sender S. To determine whether the path known to an intermediate node is more recent, *destination sequence numbers* are used. The likelihood that an intermediate node will send a Route Reply when using AODV is not as high as DSR. A new Route Request by node S for a destination is assigned a higher destination sequence number. An intermediate node which knows a route, but with a smaller sequence number, cannot send Route Reply



When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message. Node X increments the destination sequence number for D cached at node X. The incremented sequence number *N* is included in the RERR. When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*. When node D receives the route request with destination sequence number *N*, node D will set its sequence number to *N*, unless it is already larger than *N*.

Sequence numbers are used in AODV to avoid using old/broken routes and to determine which route is newer. Also, it prevents formation of loops.



Assume that A does not know about failure of link C-D because RERR sent by C is lost.

Now C performs a route discovery for D. Node A receives the RREQ (say, via path C-E-A)

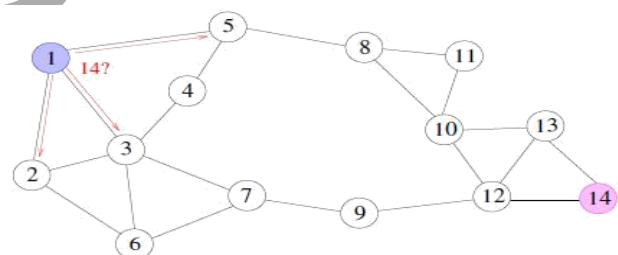
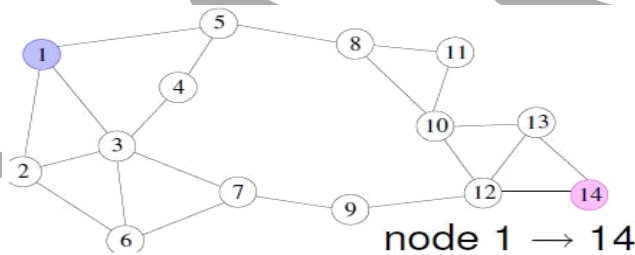
Node A will reply since A knows a route to D via node B resulting

in a loop (for instance, C-E-A-B-C)

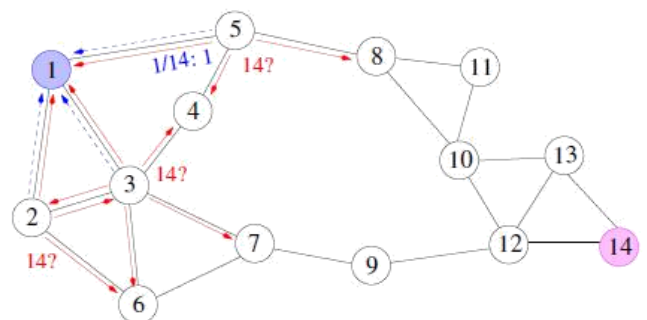
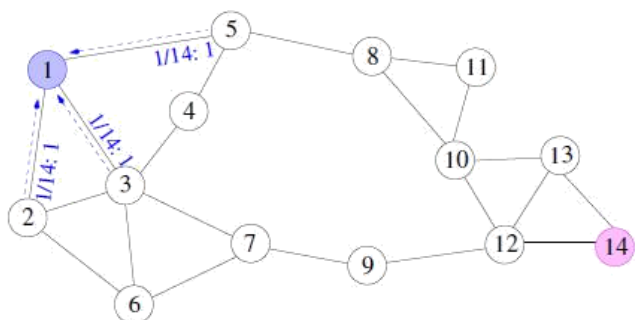
Neighboring nodes periodically exchange hello message and absence of hello message indicates a link failure. When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a **RERR message**. Node X increments the destination sequence number for D cached at node X. The incremented sequence number *N* is included in the RERR. When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*. When node D receives the route request with destination sequence number *N*, node D will set its sequence number to *N*, unless it is already larger than *N*.

Another example for AODV protocol:

Assume node-1 want to send a msg to node-14 and does not know the route. So, it broadcasts (floods) route request message, shown in red.



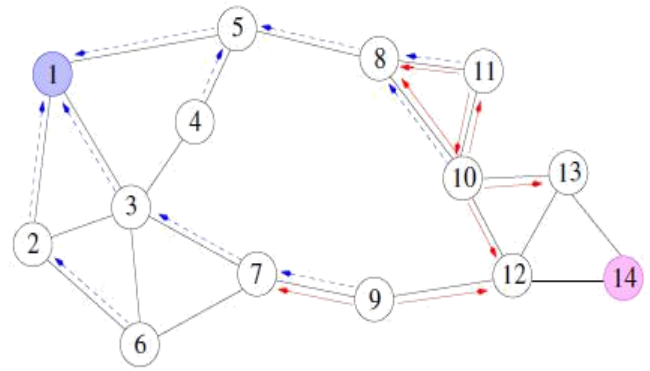
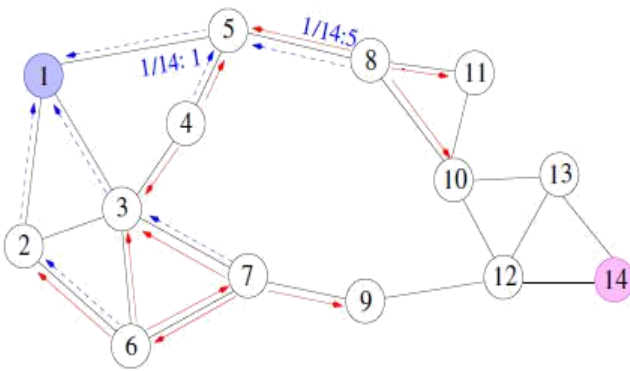
Node from which RREQ was received defines a reverse route to the source. (reverse routing table entries shown in blue).



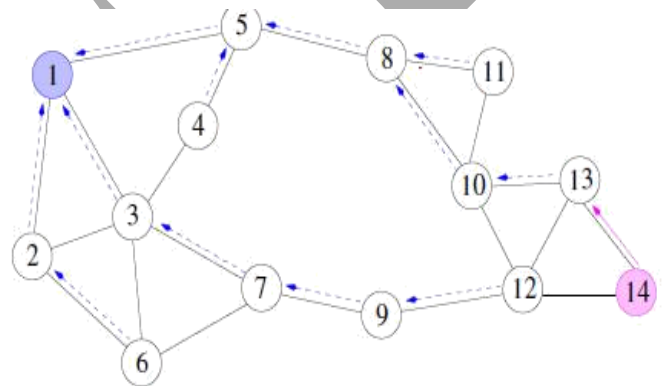
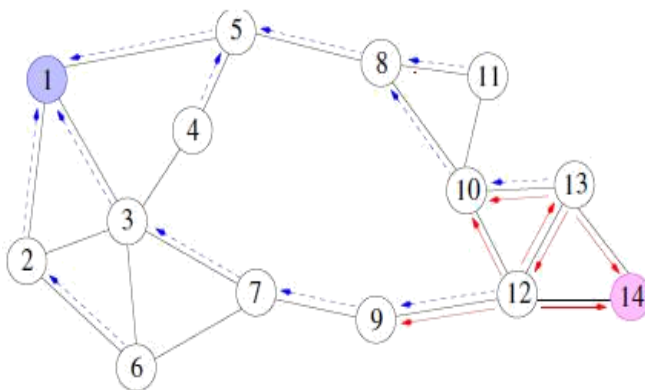
The route request is flooded through the network. Destination managed sequence number, ID prevent looping. Also, flooding is expensive and creates broadcast collision problem.

SACRED

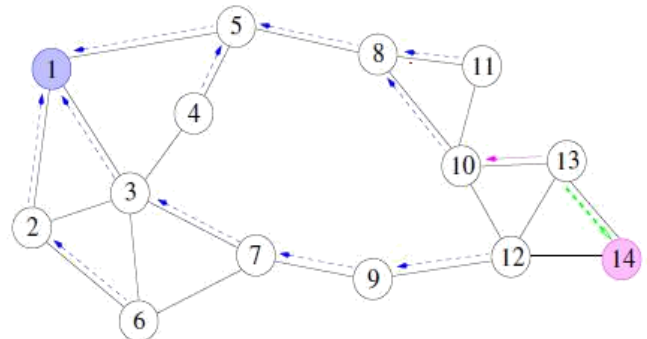
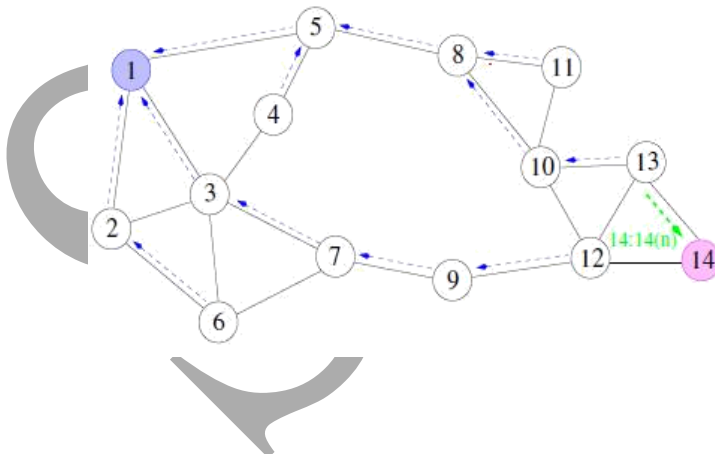
Mobile Ad Hoc Networks (MANETs)



Route request arrives at the destination node-14. Upon receiving, destination sends route reply by setting a sequence number (shown in pink)

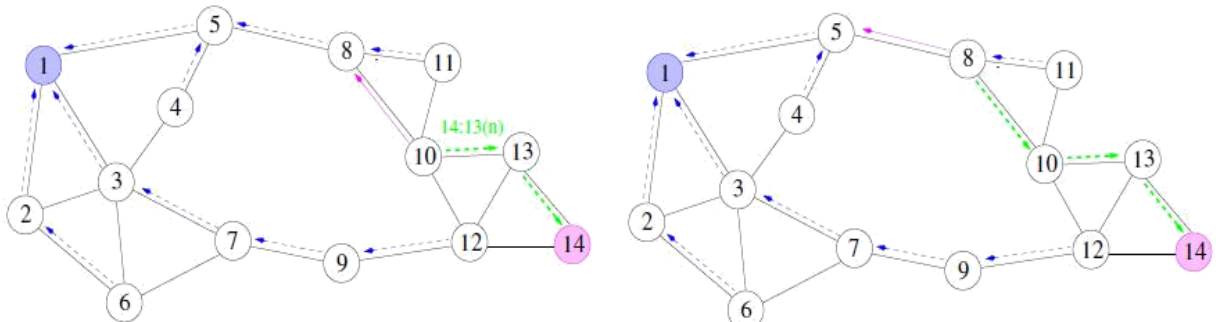


Routing table now contains forward route to the destination. Route reply follows reverse route back to the source.

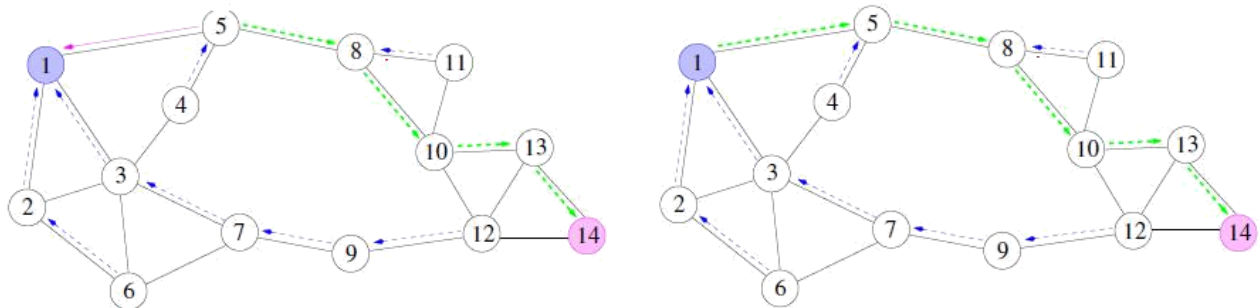


Mobile Ad Hoc Networks (MANETs)

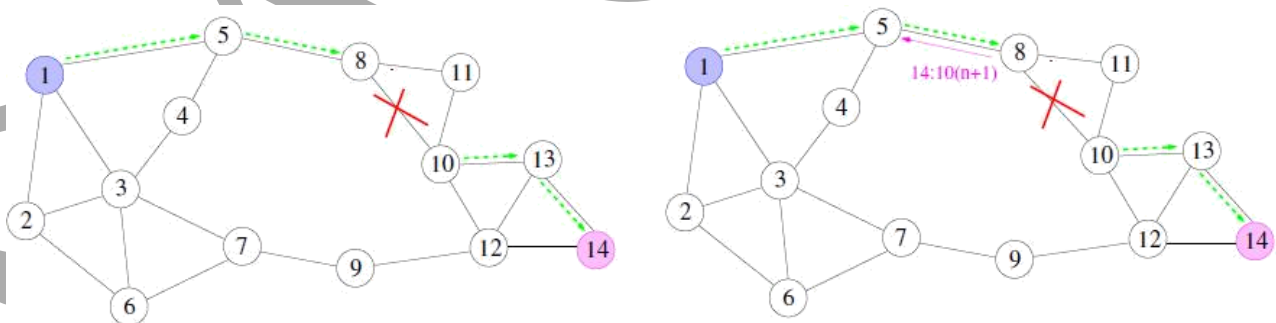
The route reply sets the forward table entries on its way back to the source.



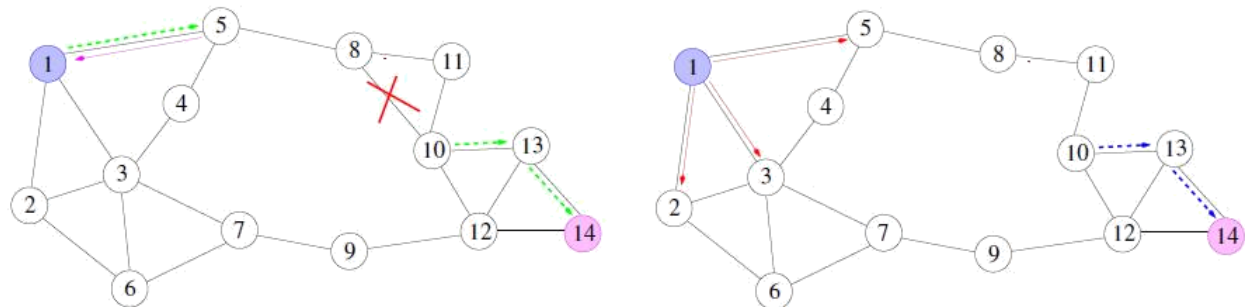
Once the route reply reaches the source, it adopts the destination sequence number. Traffic flows along the forward route. Forward route is refreshed and the reverse routes get timed out.



Suppose there has been a failure in one of the links. The node sends a return error message to the source with incrementing the sequence number.



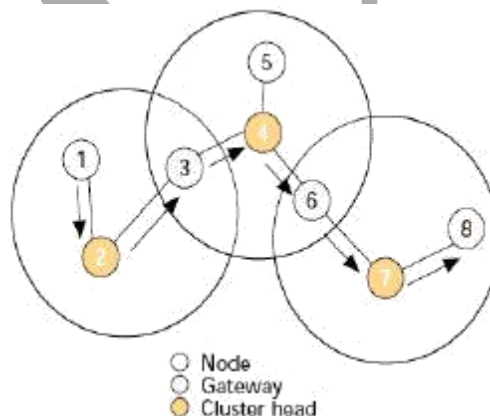
Once the source receives the route error, it re-initiates the route discovery process.



A routing table entry maintaining a reverse path is purged after a timeout interval. Timeout should be long enough to allow RREP to come back. A routing table entry maintaining a forward path is purged if *not used* for a *active_route_timeout* interval. If no data is being sent using a particular routing table entry, that entry will be deleted from the routing table (even if the route may actually still be valid).

Cluster-head Gateway Switch Routing (CGSR)

The cluster-head gateway switch routing (CGSR) is a hierarchical routing protocol. It is a proactive protocol. When a source routes the packets to destination, the routing tables are already available at the nodes. A cluster higher in hierarchy sends the packets to the cluster lower in hierarchy. Each cluster can have several daughters and forms a tree-like structure in CGSR. CGSR forms a cluster structure. The nodes aggregate into clusters using an appropriate algorithm. The algorithm defines a cluster-head, the node used for connection to other clusters. It also defines a gateway node which provides switching (communication) between two or more cluster-heads. There will thus be three types of nodes— (i) internal nodes in a cluster which transmit and receive the messages and packets through a cluster-head, (ii) cluster-head in each cluster such that there is a cluster-head which dynamically schedules the route paths. It controls a group of ad-hoc hosts, monitors broadcasting within the cluster, and forwards the messages to another cluster-head, and (iii) gateway node to carry out transmission and reception of messages and packets between cluster-heads of two clusters.



The cluster structure leads to a higher performance of the routing protocol as compared to other protocols because it provides gateway switch-type traffic redirections and clusters provide an effective membership of nodes for connectivity.

CGSR works as follow:

- periodically, every nodes sends a hello message containing its ID and a monotonically increasing sequence number

- Using these messages, every cluster-head maintains a table containing the IDs of nodes belonging to it and their most recent sequence numbers.
- Cluster-heads exchange these tables with each other through gateways; eventually, each node will have an entry in the affiliation table of each cluster-head. This entry shows the node's ID & cluster-head of that node.
- Each cluster-head and each gateway maintains a routing table with an entry for every cluster-head that shows the next gateway on the shortest path to that cluster head.

Disadvantages:

- The same disadvantage common to all hierarchal algorithms related to cluster formation and maintenance.

Hierarchal State Routing (HSR)

A hierarchal link state routing protocol that solves the location management problem found in MMWN by using the logical subnets. A logical subnet is : a group of nodes that have common characteristics (e.g. the subnet of students, the subnet of profs , employees etc.). Nodes of the same subnet do not have to be close to each other in the physical distance.

HSR procedure:

1. Based on the physical distance, nodes are grouped into clusters that are supervised by cluster-heads. There are more than one level of clustering.
2. Every node has two addresses :
 - I. a hierarchal-ID ,(HID), composed of the node's MAC address prefixed by the IDs of its parent clusters.
 - II. a logical address in the form <subnet,host>.
3. Every logical subnet has a home agent, i.e. a node that keeps track of the HID of all members of that subnet.
4. The HID's of the home agents are known to all the cluster-heads, and the cluster-head can translate the subnet part of the node's logical address to the HID of the corresponding home agent.
5. when a node moves to a new cluster, the head of the cluster detects it and informs the node's home agents about node's new HID.
6. When a home agent moves to a new cluster, the head of the cluster detects it and informs all other cluster-heads about the home agent's new HID.

To start a session:

1. The source node informs its cluster-head about the logical address of the destination node.

2. The cluster-head looks up the HID of the destination node's home agent and uses it to send query to the home agent asking about the destination's HID.
3. After knowing the destination's HID, the cluster-head uses its topology map to find a route to the destination's cluster-head.

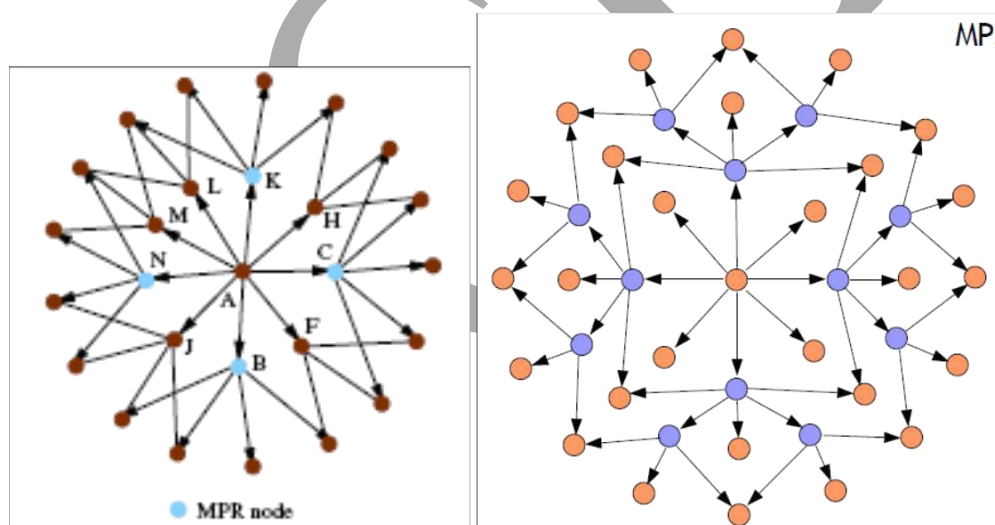
Disadvantages: cluster formation and maintenance.

Optimized Link State Routing Protocol

Optimized link state routing protocol (OLSR) has characteristics similar to those of link state flat routing table driven protocol, but in this case, only required updates are sent to the routing database. This reduces the overhead control packet size and numbers.

OSLR uses controlled flood to disseminate the link state information of each node.

- Every node creates a list of its one hop neighbors.
- Neighbor nodes exchange their lists with each other.
- Based on the received lists, each node creates its MPR.



The multipoint relays of each node, (MPR), is the minimal set of 1-hop nodes that covers all 2-hop points.

- The members of the MPR are the only nodes that can retransmit the link state information in an attempt to limit the flood.

Security in MANET's

Securing wireless ad-hoc networks is a highly challenging issue. Understanding possible form of attacks is always the first step towards developing good security solutions. Security of communication in MANET is important for secure transmission of information. Absence of any central co-ordination mechanism and shared wireless medium makes MANET more vulnerable

to digital/cyber attacks than wired network there are a number of attacks that affect MANET. These attacks can be classified into two types:

1. External Attack: External attacks are carried out by nodes that do not belong to the network. It causes congestion sends false routing information or causes unavailability of services.

2. Internal Attack: Internal attacks are from compromised nodes that are part of the network. In an internal attack the malicious node from the network gains unauthorized access and impersonates as a genuine node. It can analyze traffic between other nodes and may participate in other network activities.

- ❖ **Denial of Service attack:** This attack aims to attack the availability of a node or the entire network. If the attack is successful the services will not be available. The attacker generally uses radio signal jamming and the battery exhaustion method.
- ❖ **Impersonation:** If the authentication mechanism is not properly implemented a malicious node can act as a genuine node and monitor the network traffic. It can also send fake routing packets, and gain access to some confidential information.
- ❖ **Eavesdropping:** This is a passive attack. The node simply observes the confidential information. This information can be later used by the malicious node. The secret information like location, public key, private key, password etc. can be fetched by eavesdropper.
- ❖ **Routing Attacks:** The malicious node makes routing services a target because it's an important service in MANETs. There are two flavors to this routing attack. One is attack on routing protocol and another is attack on packet forwarding or delivery mechanism. The first is aimed at blocking the propagation of routing information to a node. The latter is aimed at disturbing the packet delivery against a predefined path.
- ❖ **Black hole Attack::** In this attack, an attacker advertises a zero metric for all destinations causing all nodes around it to route packets towards it.[9] A malicious node sends fake routing information, claiming that it has an optimum route and causes other good nodes to route data packets through the malicious one. A malicious node drops all packets that it receives instead of normally forwarding those packets. An attacker listen the requests in a flooding based protocol.
- ❖ **Wormhole Attack:** In a wormhole attack, an attacker receives packets at one point in the network, —tunnels them to another point in the network, and then replays them into the network from that point. Routing can be disrupted when routing control message are tunnelled. This tunnel between two colluding attacks is known as a wormhole.
- ❖ **Replay Attack:** An attacker that performs a replay attack are retransmitted the valid data repeatedly to inject the network routing traffic that has been captured previously. This attack usually targets the freshness of routes, but can also be used to undermine poorly designed security solutions.

Mobile Ad Hoc Networks (MANETs)

- ❖ **Jamming:** In jamming, attacker initially keep monitoring wireless medium in order to determine frequency at which destination node is receiving signal from sender. It then transmit signal on that frequency so that error free receptor is hindered.
- ❖ **Man- in- the- middle attack:** An attacker sites between the sender and receiver and sniffs any information being sent between two nodes. In some cases, attacker may impersonate the sender to communicate with receiver or impersonate the receiver to reply to the sender.
- ❖ **Gray-hole attack:** This attack is also known as routing misbehavior attack which leads to dropping of messages. Gray-hole attack has two phases. In the first phase the node advertise itself as having a valid route to destination while in second phase, nodes drops intercepted packets with a certain probability.

Mobile Agents

A mobile agent is a program that is autonomous and can move through a heterogeneous network under its own control, migrating from a host to host and interacting with other agents. It decides when and where to migrate. It can execute at any point or suspend its execution, move to another host and continue its execution on that host.

Mobile agents have certain features such as autonomy, mobility, goal driven, temporarily continuous, intelligence, cooperation, learning, reactivity etc. Because of these features, they are well adapted to the domain of mobile computing. For instance, a mobile agent can move from a PDA to Internet to collect interested information for the user. Since it is on the network and does not have to transfer the multiple requests/responses across the low bandwidth connection, it can access necessary resources efficiently.

Mobile agents simplify the development, testing and implementation of distributed applications because of their ability to hide the communication channels and show the computation logic. They can distribute and redistribute themselves throughout the network and can act as either clients or servers depending on their goals. They can also increase the scalability of the applications because of their ability to move work to an appropriate location.

Advantages of Mobile Agents

Mobile agents have several advantages over traditional client/server model. Some of these include

Reduction of network traffic

In distributed systems, performing a simple job involves multiple interactions, resulting in increased network traffic. In mobile agent paradigm, the objective is to move the computation to the

data rather than the data to the computation, thus consuming fewer network resources and thereby increasing efficiency.

Overcome network latency

Management of critical real time systems with substantial size networks creates latencies, which are unacceptable. Deployment of mobile agents will overcome this problem since these agents can execute locally upon the central controller's directions.

Encapsulation of protocols

Often, as new communication protocols which improve efficiency and security emerge, businesses need to upgrade their protocols, otherwise they turn into a legacy problem. This whole process is cumbersome. Mobile agents will resolve this problem, as they can migrate to a remote host and establish channels based on proprietary protocols

Asynchronous and autonomous execution

For jobs that require a continuous open connection, deployment of mobile agents will result in cost savings. Mobile agents with embedded tasks can be dispatched into the network where they operate independently and asynchronously.

Dynamic adoption

Mobile agents can perceive the surrounding environment and can act dynamically.

Seamless system integration

Both from the perspectives of hardware and software, networking computing are heterogeneous. Mobile agents can provide seamless system integration since they are dependent only on those environments in which they execute.

Robust and fault-tolerant

Since mobile agents can act and react dynamically in presence of unfavorable conditions, it is easy to build a robust and fault-tolerant distributed system with mobile agents.

Applications of Mobile Agents

Several applications benefit from the use of mobile agent technology. Some of these include

Electronic Commerce

Many commercial transactions require access to resources in real time. The ability of a mobile agent to personify their creators intentions and to act and negotiate on behalf of them makes it well suited for electronic commerce.

Personal Assistance

An agent can act as a personal assistant to the user and perform tasks for user on a remote host regardless of whether or not user is connected to the network. For instance, to schedule a meeting, a user can dispatch a mobile agent onto the network to interact with agents belonging to other users. The

agent can negotiate with other agents the convenient time for all of the users and can schedule a meeting.

Secure brokering

Mobile agent technology is an attractive solution to brokering, particularly in the context of untrustworthy collaborators. In such a situation, the interested parties can let the agents meet and negotiate at a mutually agreed impartial secure host and form alliance

Distributed information retrieval

Mobile agent technology provides efficient information retrieval. When dealing with large amounts of data, rather than moving all the data to search engine to create search indexes, user can simply dispatch mobile agents to remote sources to create those indexes locally and to ship them back later to its origin.

Telecommunication networks services

Mobile agents provide an effective and flexible solution to the management of advanced telecommunication services by providing dynamic network reconfiguration and user customization

Workflow applications and groupware

Mobile agents because of their features such as autonomy, mobility etc, provide autonomy to the workflow item and support the information flow between co-workers

Monitoring and notification

As a local representative for remote services, an agent can perform tasks on behalf a user irrespective of whether or not user is connected to the network. For instance, a user can dispatch a mobile agent to the internet to monitor the stock prices and to notify him/her only when certain thresholds are reached

Information dissemination

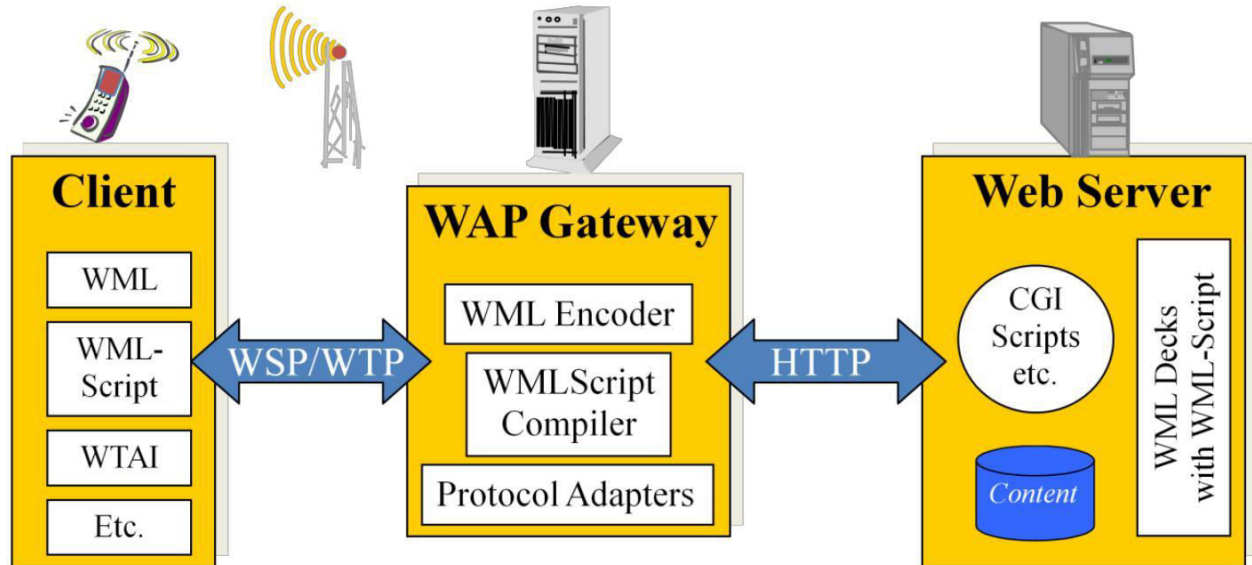
Mobile Agents, exemplifying the Internet Push Model can distribute information such as news and software updates for vendors. The agents carry the upgrades and installation procedures directly to the user's personal computer and with out any user's intervention updates and manages the software on the computer

Parallel processing

Mobile Agent technology can provide administration for parallel processing tasks. If a computation requires large amount of processing power so as it to distribute it among multiple processors, mobile agents can help get the processes out there.

Protocols and Platforms for Mobile Computing : WAP, Bluetooth, XML, J2ME, JavaCard, PalmOS, Windows CE, SymbianOS, Linux for Mobile Devices, Android.

The Wireless Application Protocol (WAP) is an open, global specification that empowers mobile users with wireless devices to easily access and interact with information and services instantly.



WAP is a global standard and is not controlled by any single company. Ericsson, Nokia, Motorola, and Unwired Planet founded the **WAP Forum** in the summer of 1997 with the initial purpose of defining an industry-wide specification for developing applications over wireless communications networks. The WAP specifications define a set of protocols in application, session, transaction, security, and transport layers, which enable operators, manufacturers, and applications providers to meet the challenges in advanced wireless service differentiation and fast/flexible service creation.

All solutions must be:

- **interoperable**, i.e., allowing terminals and software from different vendors to communicate with networks from different providers
- **scaleable**, i.e., protocols and services should scale with customer needs and number of customers
- **efficient**, i.e., provision of QoS suited to the characteristics of the wireless and mobile networks

- **reliable**, i.e., provision of a consistent and predictable platform for deploying services; and
- **secure**, i.e., preservation of the integrity of user data, protection of devices and services from security problems.

Why Choose WAP?

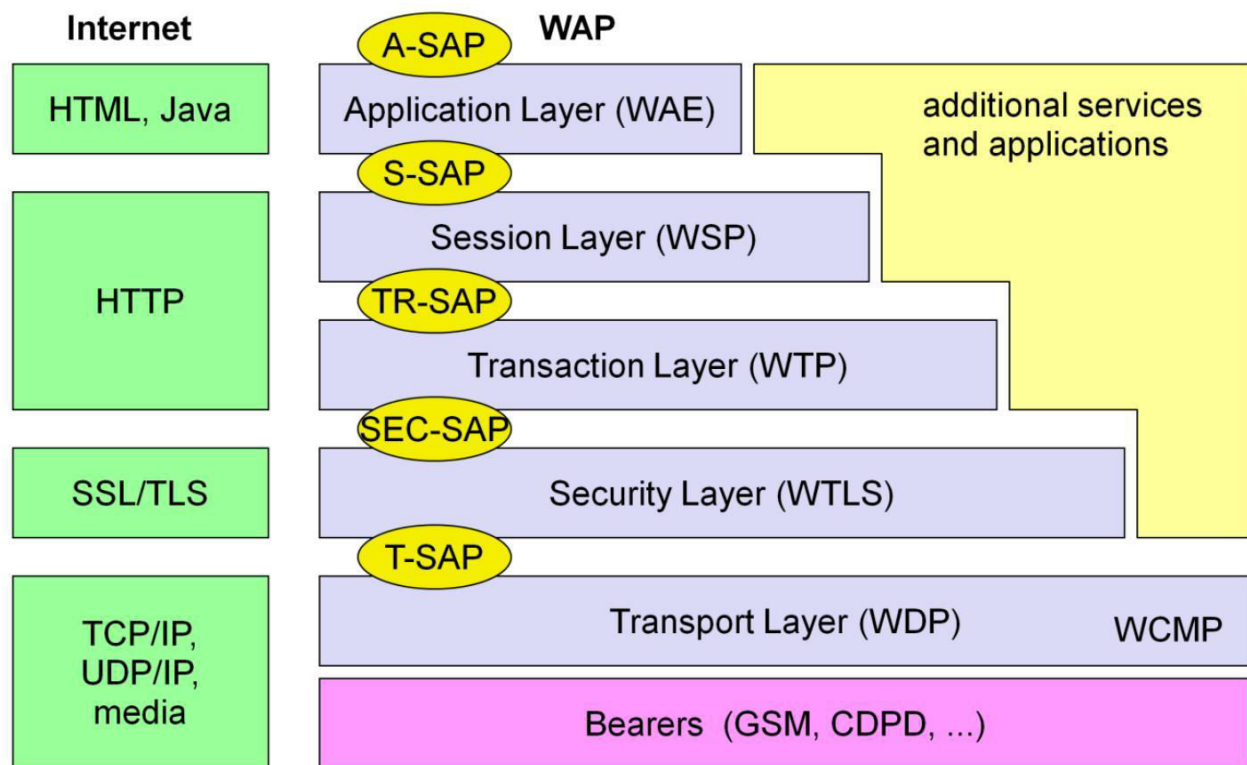
In the past, wireless Internet access has been limited by the capabilities of handheld devices and wireless networks.

- ❖ WAP utilizes Internet standards such as XML, user datagram protocol (UDP), and Internet protocol (IP). Many of the protocols are based on Internet standards such as hypertext transfer protocol (HTTP) and TLS but have been optimized for the unique constraints of the wireless environment: low bandwidth, high latency, and less connection stability.
- ❖ Internet standards such as hypertext markup language (HTML), HTTP, TLS and transmission control protocol (TCP) are inefficient over mobile networks, requiring large amounts of mainly text-based data to be sent. Standard HTML content cannot be effectively displayed on the small-size screens of pocket-sized mobile phones and pagers.
- ❖ WAP utilizes binary transmission for greater compression of data and is optimized for long latency and low bandwidth. WAP sessions cope with intermittent coverage and can operate over a wide variety of wireless transports.
- ❖ WML and wireless markup language script (WML Script) are used to produce WAP content. They make optimum use of small displays, and navigation may be performed with one hand. WAP content is scalable from a two-line text display on a basic device to a full graphic screen on the latest smart phones and communicators.
- ❖ The lightweight WAP protocol stack is designed to minimize the required bandwidth and maximize the number of wireless network types that can deliver WAP content. Multiple networks will be targeted, with the additional aim of targeting multiple networks. These include global system for mobile communications (GSM) 900, 1,800, and 1,900 MHz; interim standard (IS)-136; digital European cordless communication (DECT); time-division multiple access (TDMA), personal communications service (PCS), FLEX, and code division multiple access (CDMA). All network technologies and bearers will also be supported, including short message service (SMS), USSD, circuit-switched cellular data (CSD), cellular digital packet data (CDPD), and general packet radio service (GPRS).
- ❖ As WAP is based on a scalable layered architecture, each layer can develop independently of the others. This makes it possible to introduce new bearers or to use new transport protocols without major changes in the other layers.

- ❖ WAP will provide multiple applications, for business and customer markets such as banking, corporate database access, and a messaging interface.

WAP Architecture

The following figure gives an overview of the WAP architecture, its protocols and components, and compares this architecture with the typical internet architecture when using the World Wide Web. The basis for transmission of data is formed by different **bearer services**. WAP does not specify bearer services, but uses existing data services and will integrate further services. Examples are message services, such as short message service (SMS) of GSM, circuit-switched data, such as high-speed circuit switched data (HSCSD) in GSM, or packet switched data, such as general packet radio service (GPRS) in GSM. Many other bearers are supported, such as CDPD, IS-136, PHS.



WAE comprises WML (Wireless Markup Language), WML Script, WTAI etc.

WDP: The WAP datagram protocol (WDP) and the additional Wireless control message protocol (WCMP) is the transport layer that sends and receives messages via any available bearer network, including SMS, USSD, CSD, CDPD, IS-136 packet data, and GPRS. The *transport layer*

service access point (T-SAP) is the common interface to be used by higher layers independent of the underlying network.

WTLS: The next higher layer, the security layer with its wireless transport layer security protocol WTLS offers its service at the **security SAP (SEC-SAP)**. WTLS is based on transport layer security (TLS, formerly SSL, secure sockets layer). WTLS has been optimized for use in wireless networks with narrow-band channels. It can offer data integrity, privacy, authentication, and (some) denial-of-service protection.

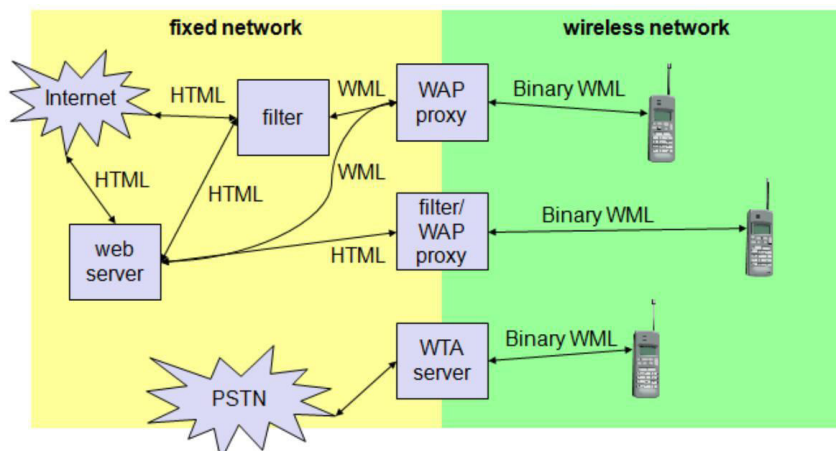
WTP: The WAP transaction protocol (WTP) provides transaction support, adding reliability layer to the datagram service provided by WDP **transaction SAP (TR-SAP)**. at the

WSP: The session layer with the wireless session protocol (WSP) currently offers two services at the session-SAP (S-SAP), one connection-oriented and one connectionless if used directly on top of WDP. A special service for browsing the web (WSP/B) has been defined that offers HTTP/1.1 functionality, long-lived session state, session suspend and resume, session migration and other features needed for wireless mobile access to the web.

WAE: The application layer with the wireless application environment (WAE) offers a framework for the integration of different www and mobile telephony applications.

Working of WAP

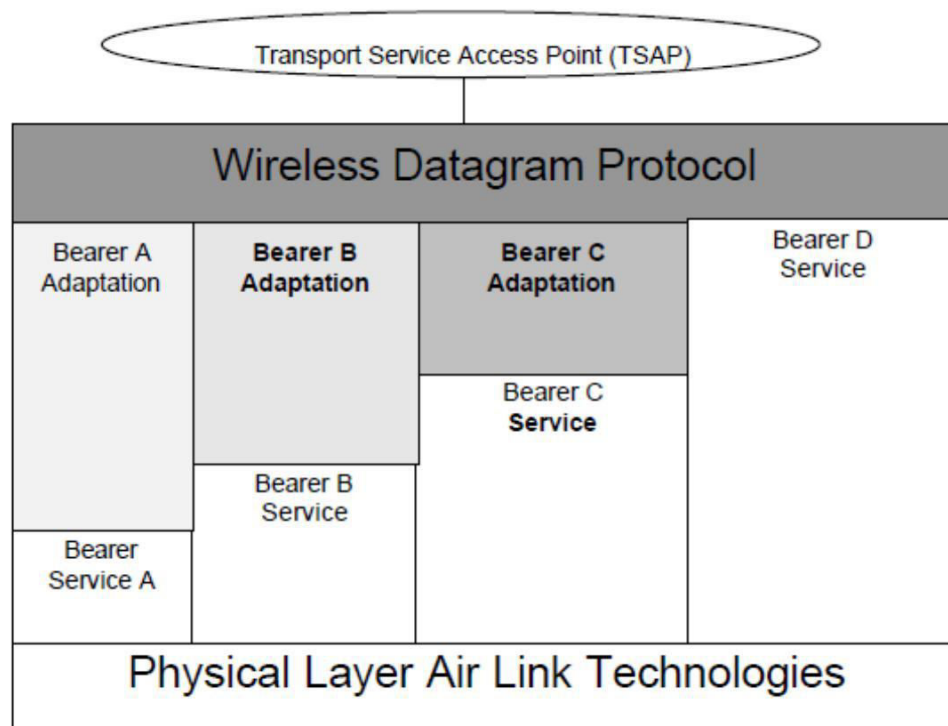
WAP does not always force all applications to use the whole protocol architecture. Applications can use only a part of the architecture. For example, if an application does not require security but needs the reliable transport of data, it can **directly** use a service of the transaction layer. Simple applications can directly use WDP.



Different scenarios are possible for the integration of WAP components into existing wireless and fixed networks. On the left side, different fixed networks, such as the traditional internet and the public switched telephone network (PSTN), are shown. One cannot change protocols and services of these existing networks so several new elements will be implemented between these networks and the WAP-enabled wireless, mobile devices in a wireless network on the right-hand side.

Wireless Datagram Protocol (WDP)

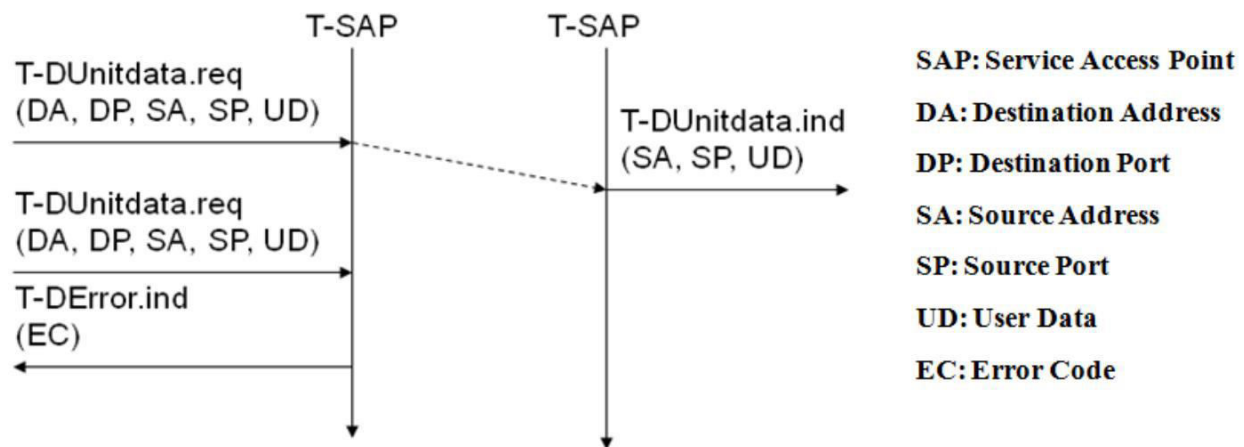
Wireless Datagram Protocol defines the movement of information from receiver to the sender and resembles the User Datagram Protocol in the Internet protocol suite.



Wireless Datagram Protocol Architecture

WDP offers a consistent service at the Transport Service Access Point to the upper layer protocol of WAP. This consistency of service allows for applications to operate transparently over different available bearer services. WDP can be mapped onto different bearers, with different characteristics. In order to optimise the protocol with respect to memory usage and radio transmission efficiency, the protocol performance over each bearer may vary.

WDP offers **source** and **destination port numbers** used for multiplexing and demultiplexing of data respectively. The service primitive to send a datagram is **T-DUnitdata.req** with the **destination address (DA)**, **destination port (DP)**, **Source address (SA)**, **source port (SP)**, and **user data (UD)** as mandatory parameters. Destination and source address are unique addresses for the receiver and sender of the user data. These could be MSISDNs (i.e., a telephone number), IP addresses, or any other unique identifiers. The **T-DUnitdata.ind** service primitive indicates the reception of data. Here destination address and port are only optional parameters.



WDP service primitives

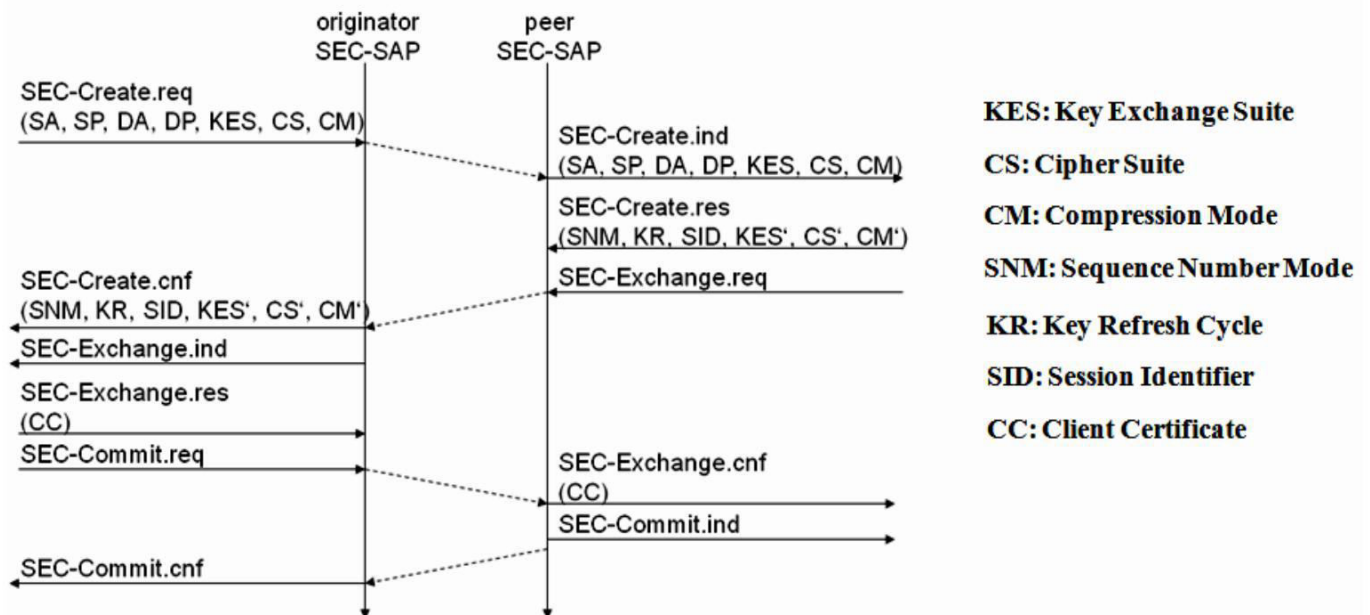
If a higher layer requests a service the WDP cannot fulfil, this error is indicated with the **T-DError.ind** service primitive. An **error code (EC)** is returned indicating the reason for the error to the higher layer. WDP is not allowed to use this primitive to indicate problems with the bearer service. It is only allowed to use the primitive to indicate local problems, such as a user data size that is too large. If any errors happen when WDP datagrams are sent from one WDP entity to another, the **wireless control message protocol (WCMP)** provides error handling mechanisms for WDP and should therefore be implemented. WCMP contains control messages that resemble the internet control message protocol messages and can also be used for diagnostic and informational purposes. WCMP can be used by WDP nodes and gateways to report errors.

Typical WCMP messages are **destination unreachable** (route, port, address unreachable), **parameter problem** (errors in the packet header), **message too big**, **reassembly failure**, or **echo request/reply**. An additional **WDP management entity** supports WDP and provides information about changes in the environment, which may influence the correct operation of WDP.

Wireless Transport Layer Security (WTLS)

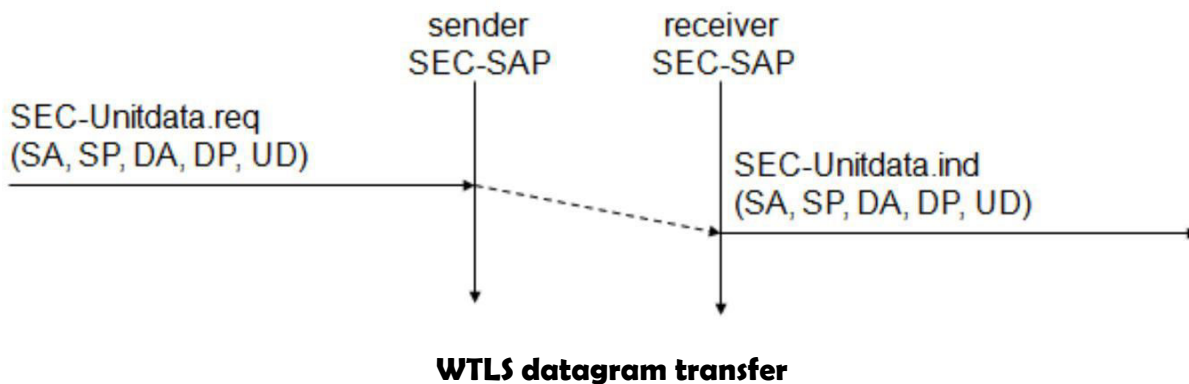
WTLS can provide different levels of security (for privacy, data integrity, and authentication) and has been optimized for low bandwidth, high-delay bearer networks. WTLS takes into account the low processing power and very limited memory capacity of the mobile devices for cryptographic algorithms. WTLS supports datagram and connection-oriented transport layer protocols. WTLS took over many features and mechanisms from TLS, but it has an optimized handshaking between the peers.

Before data can be exchanged via WTLS, a secure session has to be established. This session establishment consists of several steps: The following figure illustrates the sequence of service primitives needed for a so-called 'full handshake'.



The first step is to initiate the session with the **SEC-Create** primitive. Parameters are **source address (SA)**, **source port (SP)** of the originator, **destination address (DA)**, **destination port (DP)** of the peer. The originator proposes a **key exchange suite (KES)** (e.g., RSA, DH, ECC), a **cipher suite (CS)** (e.g., DES, IDEA), and a **compression method (CM)**. The peer answers with parameters for the **sequence number mode (SNM)**, the **key refresh cycle (KR)** (i.e., how often keys are refreshed within this secure session), the **session identifier (SID)** (which is unique with each peer), and the selected **key exchange suite (KES')**, **cipher suite (CS')**, **compression method (CM')**. The peer also issues a **SEC-Exchange** primitive. This indicates that the peer wishes to perform public-key authentication with the client, i.e., the peer

requests a **client certificate (CC)** from the originator. The first step of the secure session creation, the negotiation of the security parameters and suites, is indicated on the originator's side, followed by the request for a certificate. The originator answers with its certificate and issues a **SEC-Commit.req** primitive. This primitive indicates that the handshake is completed for the originator's side and that the originator now wants to switch into the newly negotiated connection state. The certificate is delivered to the peer side and the SEC-Commit is indicated. The WTLS layer of the peer sends back a confirmation to the originator. This concludes the full handshake for secure session setup.

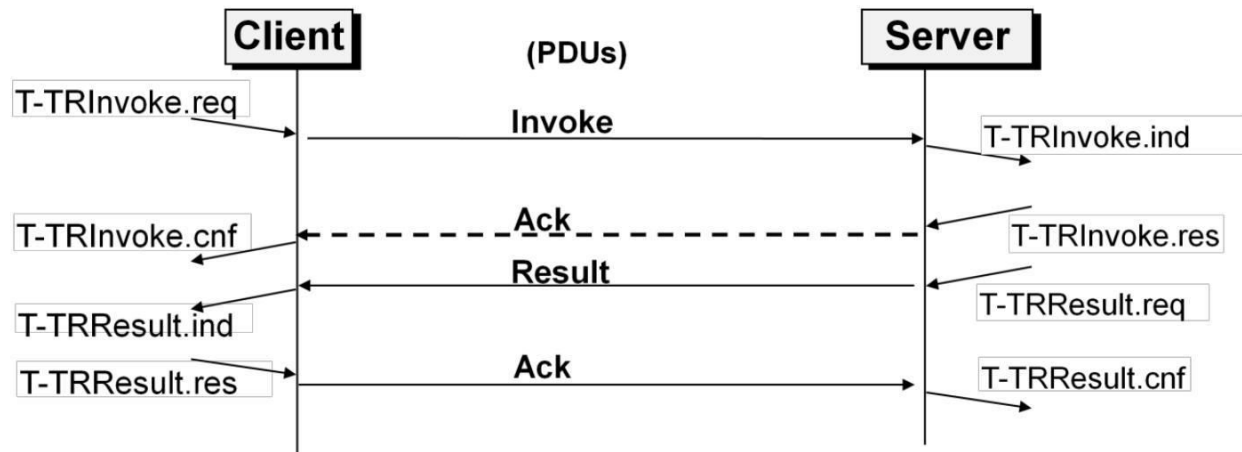


After setting up a secure connection between two peers, user data can be exchanged. This is done using the simple **SEC-Unitdata** primitive as shown in above figure. SEC-Unitdata has exactly the same function as T-DUnitdata on the WDP layer, namely it transfers a datagram between a sender and a receiver. This data transfer is still unreliable, but is now secure. This shows that WTLS can be easily plugged into the protocol stack on top of WDP.

Wireless Transaction Protocol (WTP)

WTP has been designed to run on very thin clients, such as mobile phones. WTP offers several advantages to higher layers, including an improved reliability over datagram services, improved efficiency over connection-oriented services, and support for transaction-oriented services such as web browsing. WTP offers many features to the higher layers. The basis is formed from three **classes of transaction service**. Class 0 provides unreliable message transfer without any result message. Classes 1 and 2 provide reliable message transfer, class 1 without, class 2 with, exactly one reliable result message (the typical request/response case). WTP achieves reliability using **duplicate removal, retransmission, acknowledgements** and unique **transaction identifiers**.

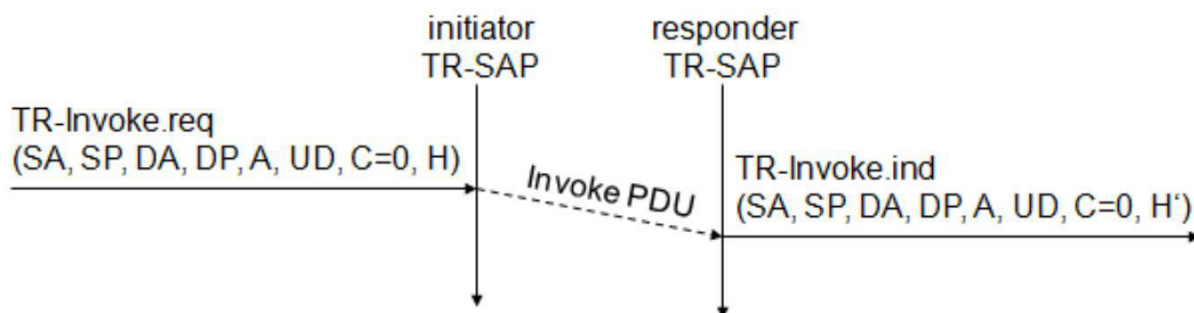
WTP allows for **asynchronous transactions**, **abort of transactions**, **concatenation of messages**, and can **report success or failure** of reliable messages (e.g., a server cannot handle the request). The three service primitives offered by WTP are **TR-Invoke** to initiate a new transaction, **TR-Result** to send back the result of a previously initiated transaction, and **TR-Abort** to abort an existing transaction.



The PDUs exchanged between two WTP entities for normal transactions are the **invoke PDU**, **ack PDU**, and **result PDU**. A special feature of WTP is its ability to provide a **user acknowledgement** or, alternatively, an **automatic acknowledgement** by the WTP entity.

WTP Class 0

Class 0 offers an unreliable transaction service without a result message. The transaction is stateless and cannot be aborted. The service is requested with the **TR-Invoke.req** primitive as shown below. Parameters are same as in WDP.

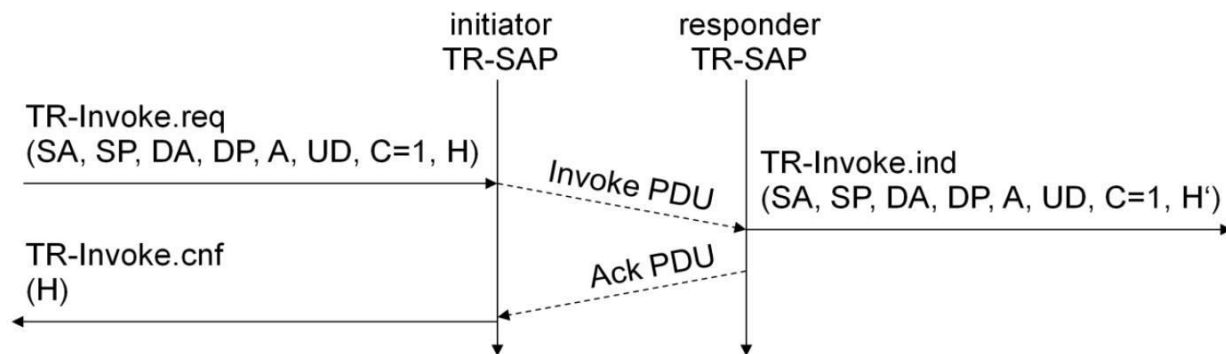


Additionally, with the **A** flag, the user of this service can determine, if the responder WTP entity should generate an **acknowledgement** or if a user acknowledgement should be used. The WTP layer will transmit the **user data (UD)** transparently to its destination. The class type **C** indicates here class 0. Finally, the transaction **handle H** provides a simple index to uniquely

identify the transaction and is an alias for the tuple (SA, SP, DA, DP), i.e., a socket pair, with only local significance. The WTP entity at the initiator sends an invoke PDU which the responder receives. The WTP entity at the responder then generates a **TR-Invoke.ind** primitive with the same parameters as on the initiator's side, except for H' which is now the local handle for the transaction on the responder's side. WTP class 0 augments the transaction service with a simple datagram like service for occasional use by higher layers.

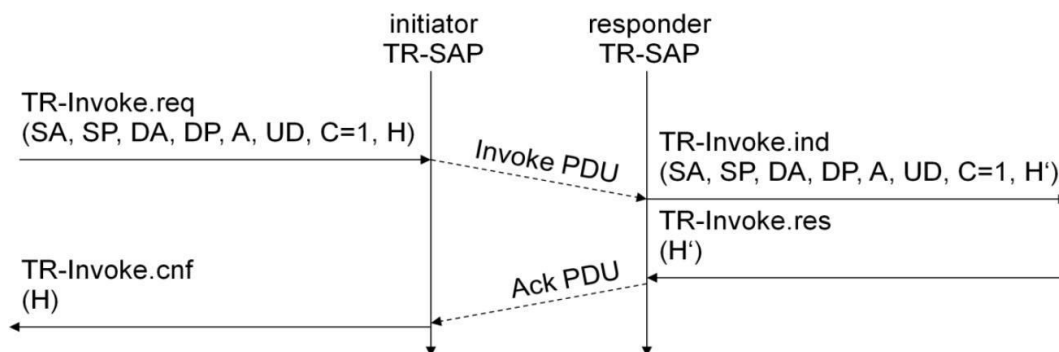
WTP Class 1

Class 1 offers a reliable transaction service but without a result message. Again, the initiator sends an invoke PDU after a **TR-Invoke.req** from a higher layer. This time, class equals '1', and no user acknowledgement has been selected as shown below.



The responder signals the incoming invoke PDU via the **TR-Invoke.ind** primitive to the higher layer and acknowledges automatically without user intervention. For the initiator the transaction ends with the reception of the acknowledgement. The responder keeps the transaction state for some time to be able to retransmit the acknowledgement if it receives the same invoke PDU again indicating a loss of the acknowledgement.

If a user of the WTP class 1 service on the initiator's side requests a user acknowledgement on the responder's side, the sequence diagram looks like the following figure.

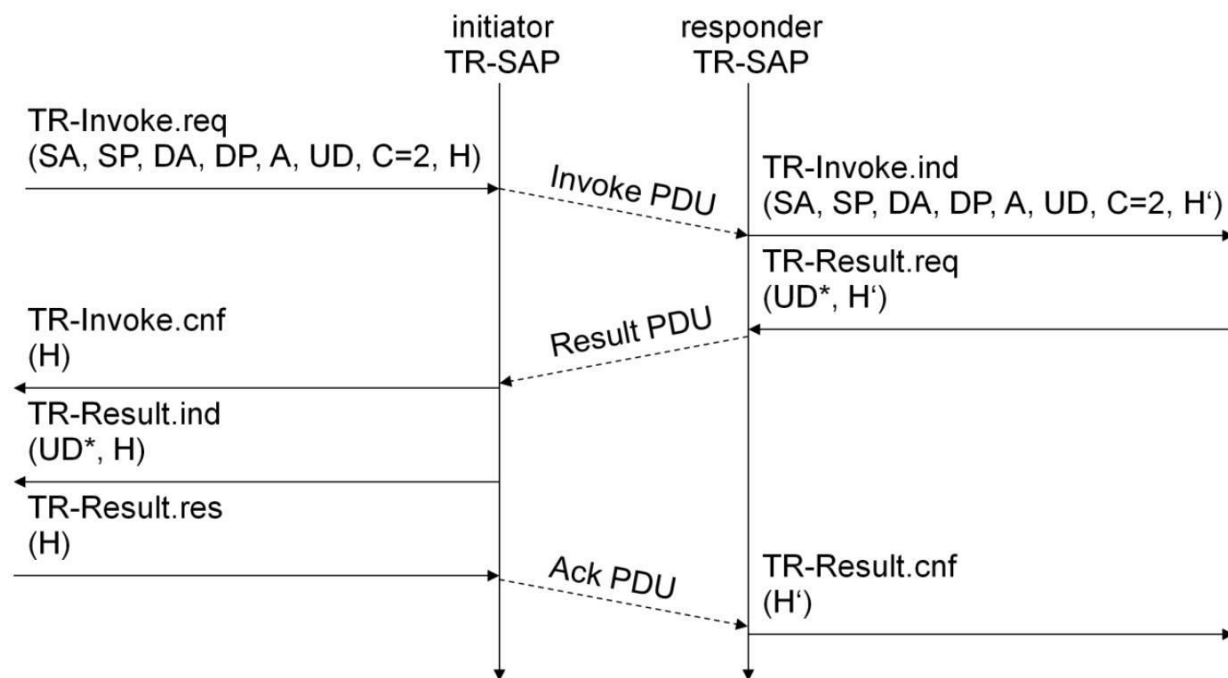


Now the WTP entity on the responder's side does not send an acknowledgement automatically, but waits for the **TR-Invoke.res** service primitive from the user. This service primitive must have the appropriate local handle H' for identification of the right transaction. The WTP entity can now send the ack PDU. Typical uses for this transaction class are reliable push services.

WTP Class 2

class 2 transaction service provides the classic reliable request/response transaction known from many client/server scenarios. Depending on user requirements, many different scenarios are possible for initiator/responder interaction. Three examples are presented below.

Example-1 scenario is shown below. A user on the initiator's side requests the service and the WTP entity sends the invoke PDU to the responder. The WTP entity on the responder's side indicates the request with the **TR-Invoke.ind** primitive to a user. The responder now waits for the processing of the request, the user on the responder's side can finally give the result UD^* to the WTP entity on the responder side using **TR-Result.req**. The **result PDU** can now be sent back to the initiator, which implicitly acknowledges the invoke PDU.



Example 1: WTP Class 2 Transaction, no user acknowledgement, no hold on

The initiator can indicate the successful transmission of the invoke message and the result with the two service primitives **TR-Invoke.cnf** and **TR-Result.ind**. A user may respond to this result with **TR-Result.res**. An acknowledgement PDU is then generated which finally triggers the **TR-Result.cnf** primitive on the responder's side. This example clearly shows the combination of

If the calculation of the result takes some time, the responder can put the initiator on “hold on” to prevent a retransmission of the invoke PDU as the initiator might assume packet loss if no result is sent back within a certain timeframe, which is shown above. After a time-out, the responder automatically generates an acknowledgement for the Invoke PDU. This shows the initiator that the responder is still alive and currently busy processing the request. After more time, the result PDU can be sent to the initiator.

Wireless Session Protocol (WSP)

The **wireless session protocol (WSP)** has been designed to operate on top of the datagram service WDP or the transaction service WTP. WSP provides a shared state between a client and a server to optimize content transfer. WSP offers the following general features needed for content exchange between cooperating clients and servers:

- **Session management:** WSP introduces sessions that can be **established** from a client to a server and may be long lived. Sessions can also be **released** in an orderly manner. The capabilities of **suspending** and **resuming** a session are important to mobile applications.
- **Capability negotiation:** Clients and servers can agree upon a common level of protocol functionality during session establishment. Example parameters to negotiate are maximum client SDU size, maximum outstanding requests, protocol options, and server SDU size.
- **Content encoding:** WSP also defines the efficient binary encoding for the content it transfers. WSP offers content typing and composite objects, as explained for web browsing.

While WSP is a general-purpose session protocol, WAP has specified the **wireless session protocol/browsing (WSP/B)** which comprises protocols and services most suited for browsing-type applications, which offers the following features adapted to web browsing.

- **HTTP/1.1 functionality:** WSP/B supports the functions HTTP/1.1 offers, such as extensible request/reply methods, composite objects, and content type negotiation.
- **Exchange of session headers:** Client and server can exchange request/reply headers that remain constant over the lifetime of the session
- **pull data transfer:** Pulling data from a server is the traditional mechanism of the web. This is also supported by WSP/B using the request/response mechanism from HTTP/1.1. Additionally, WSP/B supports three push mechanisms for data transfer: a confirmed data push within an existing session context, a non-confirmed data push within an

existing session context, and a non-confirmed data push without an existing session context.

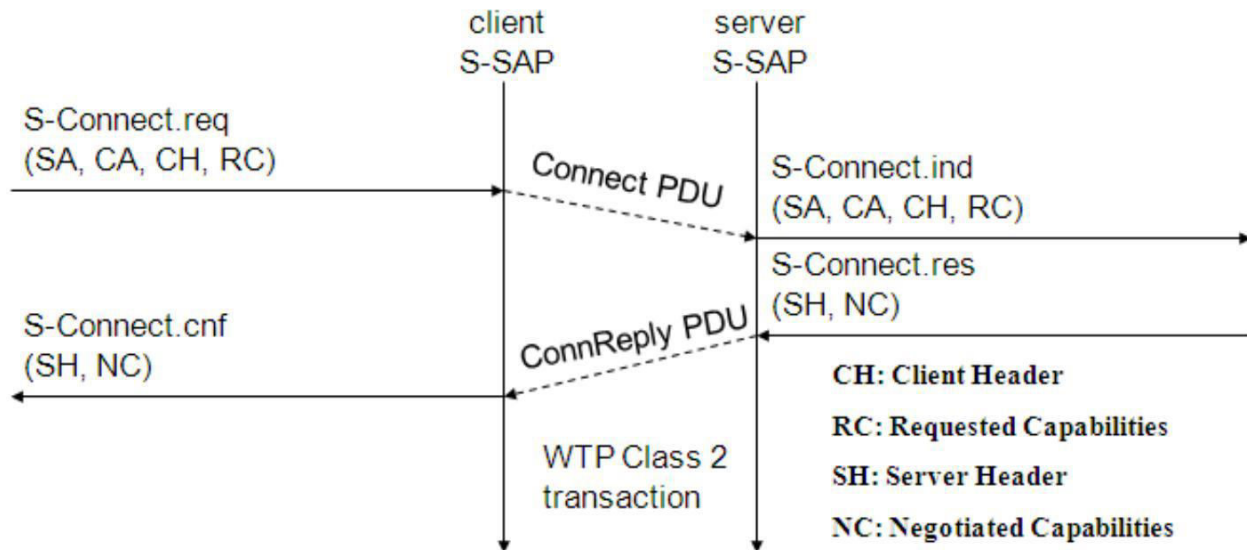


Asynchronous requests: Optionally, WSP/B supports a client that can send multiple requests to a server simultaneously. This improves efficiency for the requests and replies can now be coalesced into fewer messages.

WSP/B over WTP

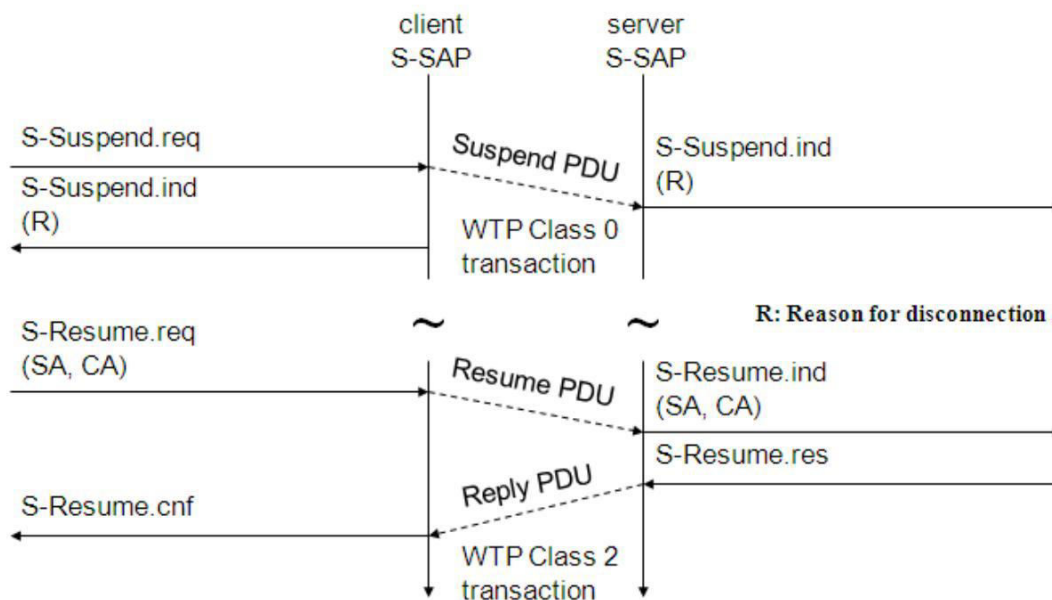
WSP/B uses the three service classes of WTP where, Class 0 is used for unconfirmed push, session resume, and session management. Confirmed push uses class 1, method invocation, session resume, and session management class 2.

The first example of session establishment of WSP/B using WTS class 2 transactions is shown below:



With the **S-Connect.req** primitive, a client can request a new session. Parameters are the **server address (SA)**, the **client address (CA)**, and the optional **client header (CH)** and **requested capabilities (RC)**. The session layer directly uses the addressing scheme of the layer below. WTP transfers the **connect PDU** to the server S-SAP where an **S-Connect.ind** primitive indicates a new session. Parameters are the same, but now the capabilities are mandatory. If the server accepts the new session it answers with an **S-Connect.res**, parameters are an optional **server header (SH)** with the same function as the client header and the **negotiated capabilities (NC)** needed for capability negotiation. WTP now transfers the **connreply PDU** back to the client; **S-Connect.cnf** confirms the session establishment and includes the **server header** (if present) and the **negotiated capabilities** from the server. WSP/B includes several procedures to refuse a session or to abort session establishment.

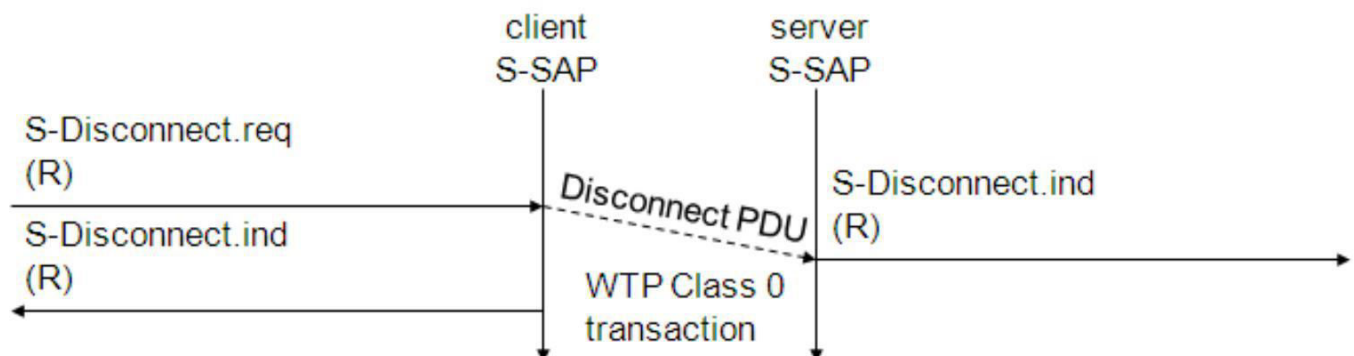
A very useful feature of WSP/B **session suspension** and **session resume** is shown below. A client can suspend the session because of several reasons. Session suspension will automatically abort all data transmission and freeze the current state of the session on the client and server side. A client suspends a session with **S-Suspend.req**, WTP transfers the **suspend PDU** to the server with a class 0 transaction, i.e., unconfirmed and unreliable. WSP/B will signal the suspension with **S-Suspend.ind** on the client and server side. The only parameter is the **reason R** for suspension. Reasons can be a user request or a suspension initiated by the service provider.



WSP/B session suspend/resume

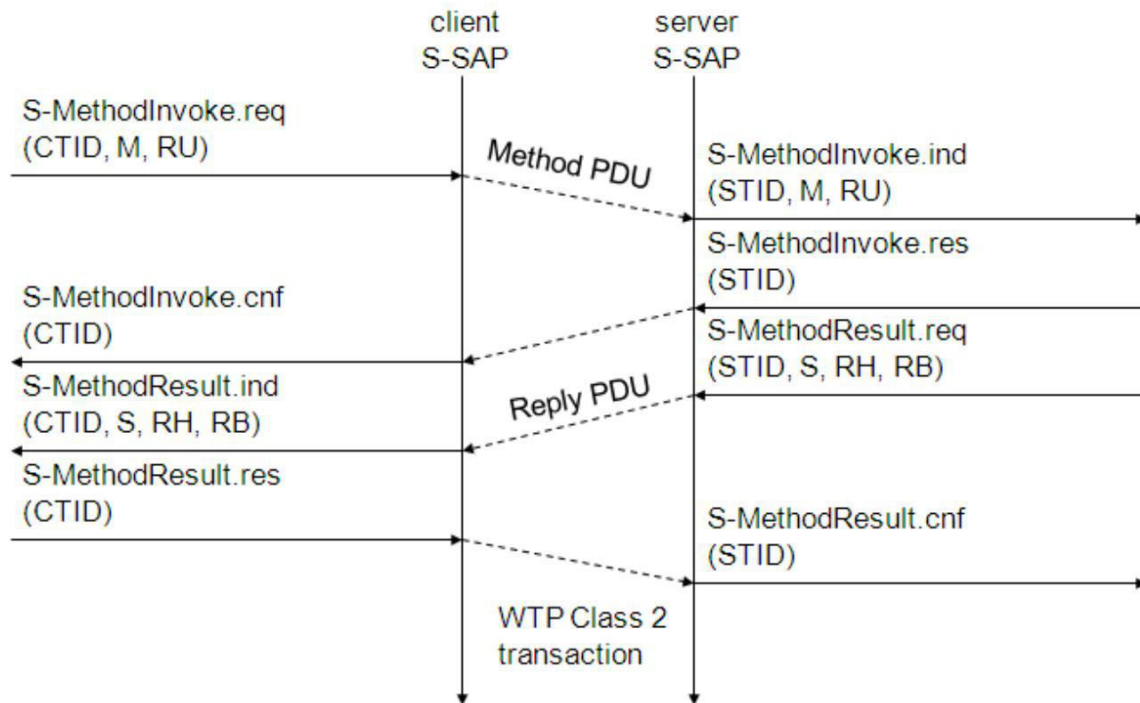
Also shown above that, a client can later resume a suspended session with **S-Resume.req**. Parameters are **server address (SA)** and **client address (CA)**. Resuming a session is a confirmed operation. It is up to the server's operator how long this state is conserved.

Terminating a session is done by using the **S-Disconnect.req** service primitive as shown below.



This primitive aborts all current method or push transactions used to transfer data. Disconnection is indicated on both sides using **S-Disconnect.ind**. The **reason R** for disconnection can be, e.g., network error, protocol error, peer request, congestion, and maximum SDU size exceeded.

The **S-MethodInvoke** primitive is used to request that an operation is executed by the server. The result, if any, is sent back using the **S-MethodResult** primitive as shown below:

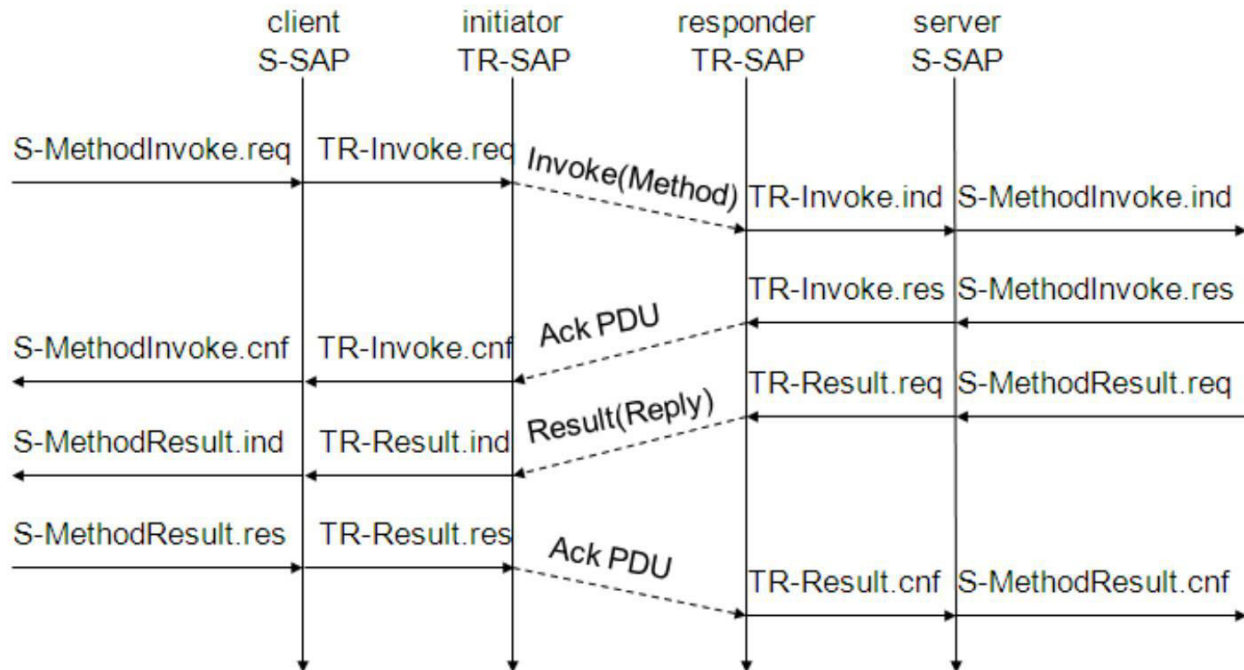


WSP/B method invoke

A client requests an operation with **S-MethodInvoke.req**. Parameters are the **client transaction identifier CTID** to distinguish between pending transactions, the **method M** identifying the requested operation at the server, and the **request URI (Uniform Resource Identifier RU)**. The WTP class 2 transaction service now transports the **method PDU** to the server. A method PDU can be either a get PDU or a post PDU.

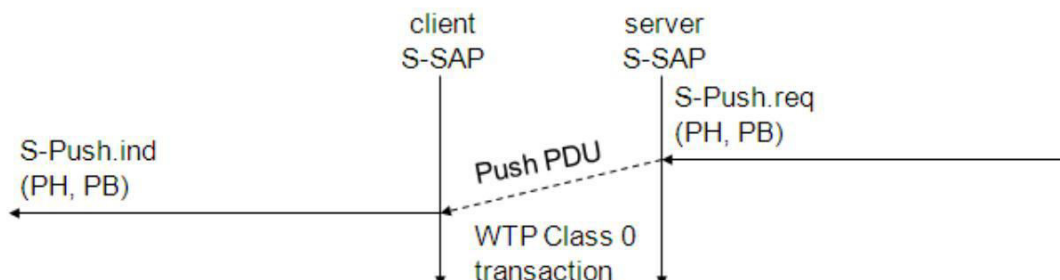
On the server's side, **S-MethodInvoke.ind** indicates the request. In this case, a **server transaction identifier STID** distinguishes between pending transactions. The server confirms the request, so WSP/B does not generate a new PDU but relies on the lower WTP layer. Similarly, the result of the request is sent back to the client using the **SMethodResult** primitive. Additional parameters are now the **status (S)**, the **response header (RH)**, and the **response body (RB)**.

WSP does not introduce PDUs or service primitives just for the sake of symmetric and aesthetic protocol architecture. The following figure shows how WSP (thus also WSP/B) uses the underlying WTP services for its purposes. The **S-MethodInvoke.req** primitive triggers the **TR-Invoke.req** primitive, the parameters of the WSP layer are the user data of the WTP layer. The **invoke PDU** of the WTP layer carries the **method PDU** of the WSP layer inside.



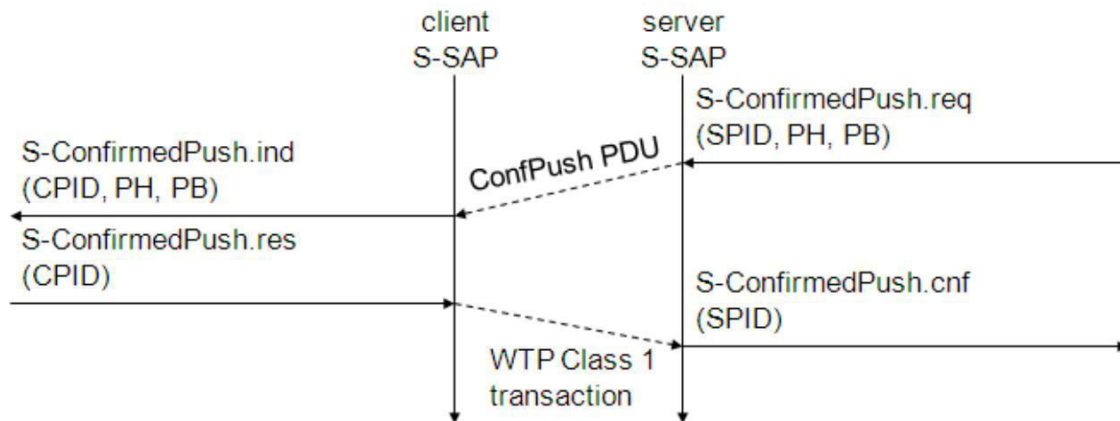
For the confirmation of its service primitives the WSP layer has none of its own PDUs but uses the **acknowledgement PDUs** of the WTP layer. **S-MethodInvoke.res** triggers **TR-Invoke.res**, the **ack PDU** is transferred to the initiator, here **TR-Invoke.cnf** confirms the invoke service and triggers the **S-MethodInvoke.cnf** primitive which confirms the method invocation service. This mingling of layers saves a lot of redundant data flow but still allows a separation of the tasks between the two layers.

With the help of push primitives, a server can push data towards a client if allowed. The simplest push mechanism is the non-confirmed push as shown below.



The server sends unsolicited data with the **S-Push.req** primitive to the client. Parameters are the **push header (PH)** and the **push body (PB)** again, these are the header and the body known from HTTP. The unreliable, unconfirmed WTP class 0 transaction service transfers the **push PDU** to the client where **S-Push.ind** indicates the push event.

A more reliable push service offers the **S-ConfirmedPush** primitive as shown below.

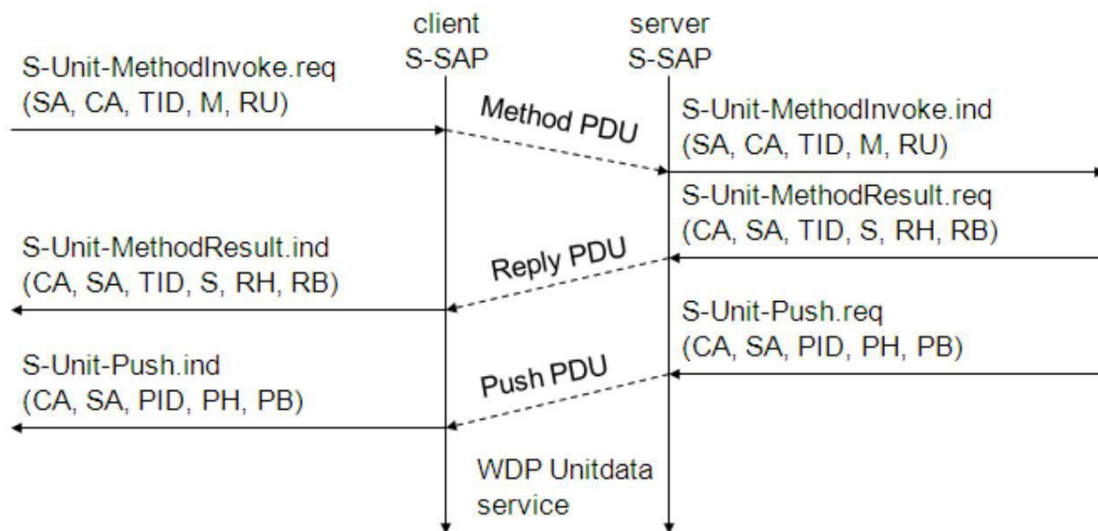


Here the server has to determine the push using a **server push identifier (SPID)**. This helps to distinguish between different pending pushes. The reliable WTP class 1 transaction service is now used to transfer the **confpush PDU** to the client. On the client's side a **client push identifier (CPID)** is used to distinguish between different pending pushes.

WSP/B as connectionless session service

WSP/B could be run on top of the connectionless, unreliable WDP service. As an alternative to WDP, WTLS can always be used if security is required. The service primitives are directly mapped onto each other. The following figure shows the three service primitives available for connectionless session service: **S-Unit-MethodInvoke.req** to request an operation on a server, **S-Unit-MethodResult.req** to return results to a client, and **S-Unit-Push.req** to push data onto a client. Transfer of the PDUs (**method**, **reply** and **push**) is done with the help of the standard unreliable datagram transfer service of WDP.

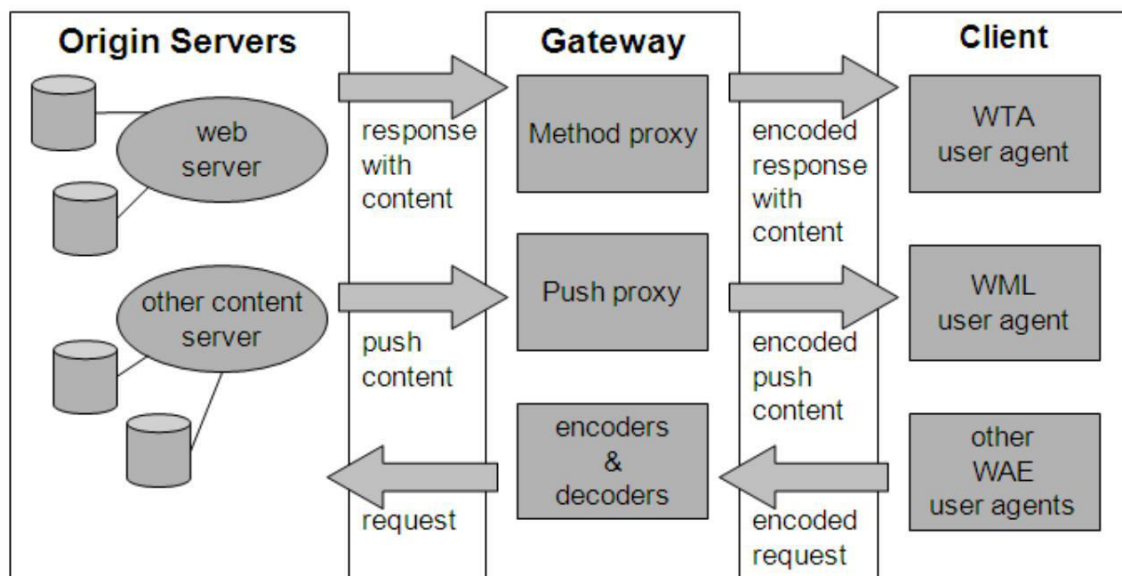
Besides the server address (SA), the client address (CA), the method (M), and the request URI (RU), the user of the **S-Unit-MethodInvoke.req** primitive can determine a transaction identifier (TID) to distinguish between different transactions on the user level. TID is communicated transparently from service user to service user.



The function of the **S-Unit-MethodResult** primitive remains the same as explained above: the **status (S)**, **response header (RH)**, and **response body (RB)** represent the result of the operation. The **S-Unit-Push** primitive has the parameters **client address (CA)**, **server address (SA)**, **push identifier (PID)**, **push header (PH)**, and **push body (PB)**.

Wireless application environment (WAE)

The main idea behind the wireless application environment (WAE) is to create a general-purpose application environment based mainly on existing technologies and philosophies of the world wide web. One global goal of the WAE is to minimize over-the-air traffic and resource consumption on the handheld device, which is reflected in the logical model shown below:



A **client** issues an encoded request for an operation on a remote server. Encoding is necessary to minimize data sent over the air and to save resources on the handheld device. Decoders in a **gateway** now translate this encoded request into a standard request as understood by the **origin servers**. This could be a request to get a web page to set up a call. The gateway transfers this request to the appropriate origin server as if it came from a standard client. Origin servers could be standard web servers running HTTP and generating content using scripts, providing pages using a database, or applying any other (proprietary) technology.

The origin servers will respond to the request. The gateway now encodes the response and its content (if there is any) and transfers the encoded response with the content to the client. The WAE logical model not only includes this standard request/response scheme, but it also includes push services. Then an origin server pushes content to the gateway. The gateway encodes the pushed content and transmits the encoded push content to the client. Several user agents can reside within a client. User agents include such items as: browsers, phonebooks, message editors etc. WAE does not specify the number of user agents or their functionality, but assumes a basic **WML user agent** that supports WML, WMLscript, or both (i.e., a 'WML browser'). However, one more user agent has been specified with its fundamental services, the **WTA user agent**. This user agent handles access to, and interaction with, mobile telephone features (such as call control). As over time many vendor dependent user agents may develop, the standard defines a **user agent profile (UAProf)**, which describes the capabilities of a user agent.

Wireless Markup Language (WML)

The **wireless markup language (WML)** is based on the standard HTML known from the www and on HDML. WML is specified as an XML document type. Several constraints of wireless handheld devices had to be taken into account, when designing WML.

WML follows a deck and card metaphor. A WML document is made up of multiple **cards**. Cards can be grouped together into a **deck**. A WML deck is similar to an HTML page, in that it is identified by a URL and is the unit of content transmission. A user navigates with the WML browser through a series of WML cards, reviews the contents, enters requested data, makes choices etc. The WML browser fetches decks as required from origin servers. Either these decks can be static files on the server or they can be dynamically generated. WML describes the intent of interaction in an abstract manner. The user agent on a handheld device has to decide how to

best present all elements of a card. This presentation depends much on the capabilities of the device.

WML includes several basic features:

- ❖ **Text and images:** WML gives, as do other mark-up languages, hints how text and images can be presented to a user
- ❖ **User interaction:** WML supports different elements for user input. Examples are: text entry controls for text or password entry, option selections or controls for task invocation. **Navigation:** As with HTML browsers, WML offers a history mechanism with navigation through the browsing history, hyperlinks and other intercard navigation elements.
- ❖ **Context management:** WML allows for saving the state between different decks without server interaction, i.e., variable state can last longer than a single deck, and so state can be shared across different decks.

WML script

WMLScript complements to WML and provides a general scripting capability in the WAP architecture. While all WML content is static (after loading on the client), WMLScript offers several capabilities not supported by WML:

- ❖ **Validity check of user input:** before user input is sent to a server, WMLScript can check the validity and save bandwidth and latency in case of an error.
- ❖ **Access to device facilities:** WMLScript offers functions to access hardware components and software functions of the device.
- ❖ **Local user interaction:** Without introducing round-trip delays, WMLScript can directly and locally interact with a user, show messages or prompt for input.
- ❖ **Extensions to the device software:** With the help of WMLScript a device can be configured and new functionality can be added even after deployment.

WMLScript is based on JavaScript, but adapted to the wireless environment. WMLScript is event-based, i.e., a script may be invoked in response to certain user or environment events. WMLScript also has full access to the state model of WML, i.e., WMLScript can set and read WML variables. WMLScript provides many features known from standard programming languages such as functions, expressions, or while, if, for, return etc. The WAP Forum has specified several **standard libraries** for WMLScript (WAP Forum, 2000i). These libraries provide access to the core functionality of a WAP client so they, must be available in the client's scripting environment. The six libraries defined are **Lang, Float, String, URL, WML browser** and

BLUETOOTH

"Bluetooth" was the nickname of Harald Blåtand II, king of Denmark from 940 to 981, who united all of Denmark and part of Norway under his rule. **Bluetooth** is a proprietary open wireless technology standard for exchanging data over short distances (using short wavelength radio transmissions in the ISM band from 2400-2480 MHz) from fixed and mobile devices, creating personal area networks (PANs) with high levels of security. The Bluetooth technology aims at so-called **ad-hoc piconets**, which are local area networks with a very limited coverage and without the need for an infrastructure.

Bluetooth Features

- Bluetooth is wireless and automatic. You don't have to keep track of cables, connectors, and connections, and you don't need to do anything special to initiate communications. Devices find each other automatically and start conversing without user input, except where authentication is required; for example, users must log in to use their email accounts.
- Bluetooth is inexpensive. Market analysts peg the cost to incorporate Bluetooth technology into a PDA, cell phone, or other product at a minimum cost.
- The ISM band that Bluetooth uses is regulated, but unlicensed. Governments have converged on a single standard, so it's possible to use the same devices virtually wherever you travel, and you don't need to obtain legal permission in advance to begin using the technology.
- Bluetooth handles both data and voice. Its ability to handle both kinds of transmissions simultaneously makes possible such innovations as a mobile hands-free headset for voice with applications that print to fax, and that synchronize the address books on your PDA, your laptop, and your cell phone.
- Signals are omni-directional and can pass through walls and briefcases. Communicating devices don't need to be aligned and don't need an unobstructed line of sight like infrared.
- Bluetooth uses *frequency hopping*. Its *spread spectrum* approach greatly reduces the risk that communications will be intercepted.

Bluetooth Applications

- File transfer.
- Ad-hoc networking: Communicating devices can spontaneously form a community of networks that persists only as long as it's needed

- Device synchronization: Seamless connectivity among PDAs, computers, and mobile phones allows applications to update information on multiple devices automatically when data on any one device changes.
- Peripheral connectivity.
- Car kits: Hands-free packages enable users to access phones and other devices without taking their hands off the steering wheel
- Mobile payments: Your Bluetooth-enabled phone can communicate with a Bluetooth-enabled vending machine to buy a can of Diet Pepsi, and put the charge on your phone bill.

The 802.11b protocol is designed to connect relatively large devices with lots of power and speed, such as desktops and laptops, where devices communicate at up to 11 Mbit/sec, at greater distances (up to 300 feet, or 100 meters). By contrast, Bluetooth is designed to connect small devices like PDAs, mobile phones, and peripherals at slower speeds (1 Mbit/sec), within a shorter range (30 feet, or 10 meters), which reduces power requirements. Another major difference is that 802.11b wasn't designed for voice communications, while any Bluetooth connection can support both data and voice communications.

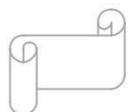
User scenarios

Many different user scenarios can be imagined for wireless piconets or WPANs:

Connection of peripheral devices: Today, most devices are connected to a desktop computer via wires (e.g., keyboard, mouse, joystick, headset, speakers). This type of connection has several disadvantages: each device has its own type of cable, different plugs are needed, wires block office space. In a wireless network, no wires are needed for data transmission. However, batteries now have to replace the power supply, as the wires not only transfer data but also supply the peripheral devices with power.

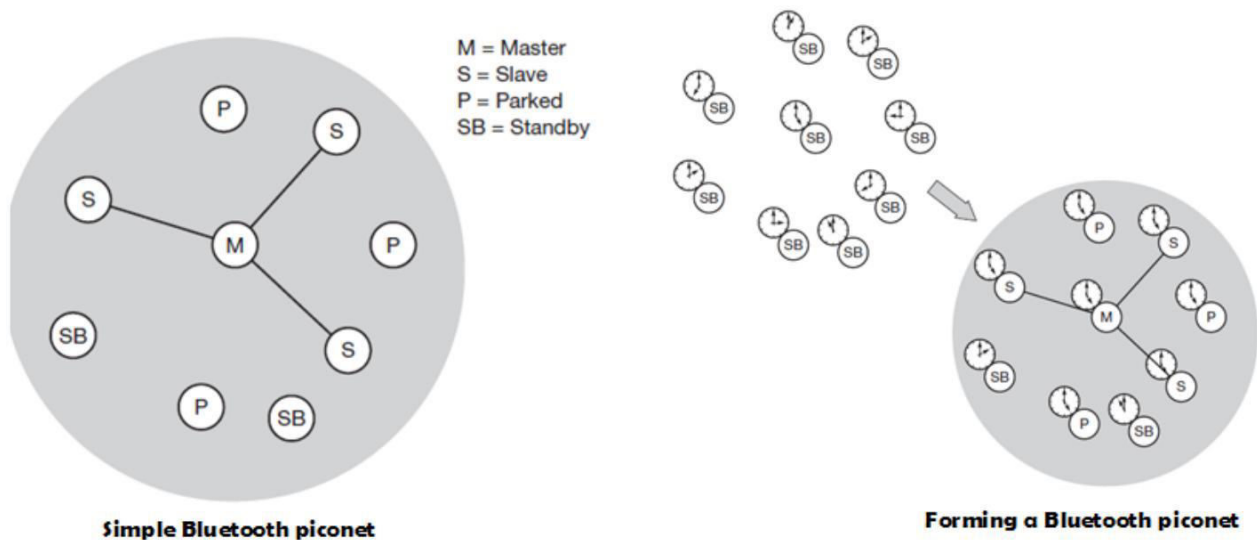
Support of ad-hoc networking: Imagine several people coming together, discussing issues, exchanging data (schedules, sales figures etc.). For instance, students might join a lecture, with the teacher distributing data to their personal digital assistants (PDAs). Wireless networks can support this type of interaction; small devices might not have WLAN adapters following the IEEE 802.11 standard, but cheaper Bluetooth chips built in.

Bridging of networks: Using wireless piconets, a mobile phone can be connected to a PDA or laptop in a simple way. Mobile phones will not have full WLAN adapters built in, but could have a Bluetooth chip. The mobile phone can then act as a bridge between the local piconet and, e.g., the global GSM network.



Networking in Bluetooth

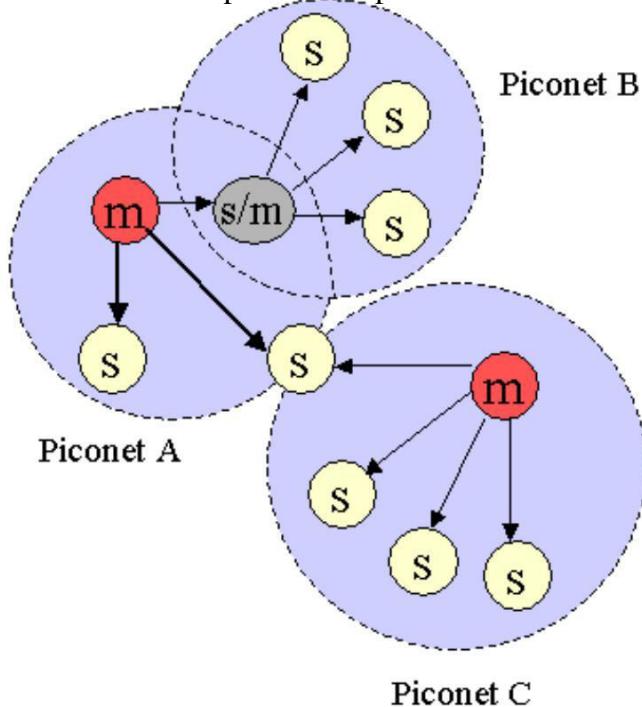
Bluetooth operates on 79 channels in the 2.4 GHz band with 1 MHz carrier spacing. Each device performs frequency hopping with 1,600 hops/s in a pseudo random fashion. A piconet is a collection of Bluetooth devices which are synchronized to the same hopping sequence. One device in the piconet can act as **master** (M), all other devices connected to the master must act as **slaves** (S). The master determines the hopping pattern in the piconet and the slaves have to synchronize to this pattern. Each piconet has a unique hopping pattern. If a device wants to participate it has to synchronize to this. A typical piconet is shown below:



Parked devices (P) can not actively participate in the piconet (i.e., they do not have a connection), but are known and can be reactivated within some milliseconds. Devices in stand-by (SB) do not participate in the piconet. Each piconet has exactly one master and up to seven simultaneous slaves. More than 200 devices can be parked. The first step in forming a piconet involves a master sending its clock and device ID. All the Bluetooth devices have the same capability to become a master or a slave and two or three devices are sufficient to form a piconet. The unit establishing the piconet automatically becomes the master, all other devices will be slaves. The hopping pattern is determined by the device ID, a 48-bit worldwide unique identifier.

The phase in the hopping pattern is determined by the master's clock. After adjusting the internal clock according to the master a device may participate in the piconet. All active devices are assigned a 3-bit **active member address** (AMA). All parked devices use an 8-bit **parked member address** (PMA). Devices in stand-by do not need an address.

A device in one piconet can communicate to another device in another piconet, forming a **scatternet**. A master in one piconet may be a slave in another piconet. Both piconets use a different hopping sequence, always determined by the master of the piconet. Bluetooth applies **FH-CDMA** for separation of piconets. A collision occurs if two or more piconets use the same



carrier frequency at the same time. This will probably happen as the hopping sequences are not coordinated. If a device wants to participate in more than one piconet, it has to synchronize to the hopping sequence of the piconet it wants to take part in. If a device acts as slave in one piconet, it simply starts to synchronize with the hopping sequence of the piconet it wants to join. After synchronization, it acts as a slave in this piconet and no longer participates in its former piconet. To enable synchronization, a slave has to know the identity of the master that determines the hopping sequence of a piconet. Before

leaving one piconet, a slave informs the current master that it will be unavailable for a certain amount of time. The remaining devices in the piconet continue to communicate as usual.

Bluetooth Protocol Stack

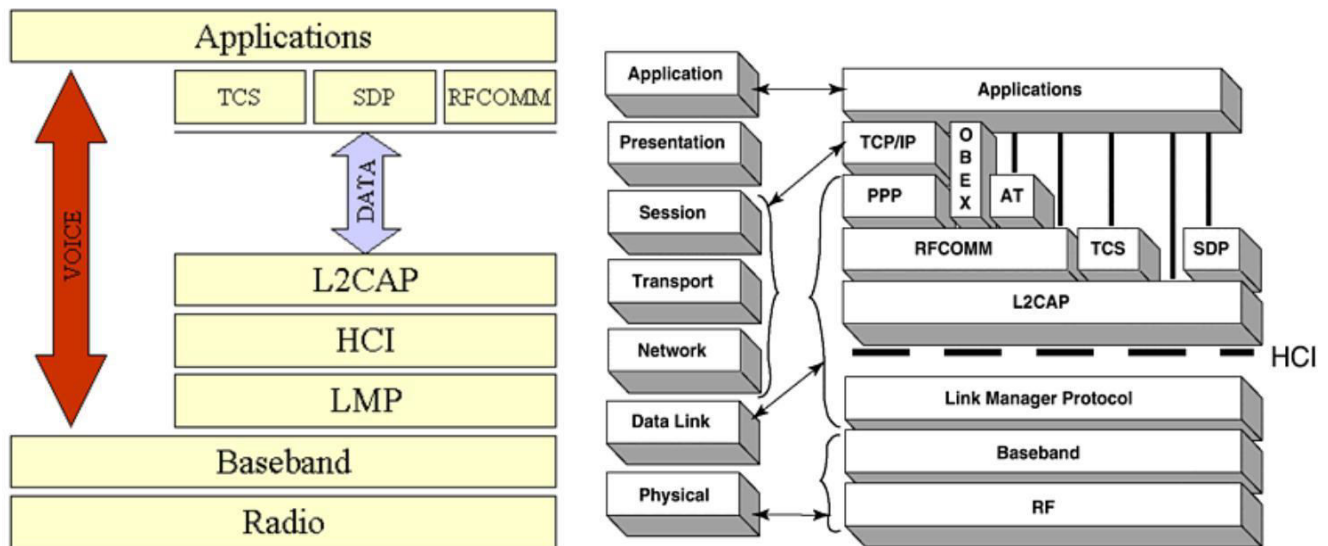
The Bluetooth protocol stack can be divided into a **core specification**, which describes the protocols from physical layer to the data link control together with management functions, and **profile specifications** describing many protocols and functions needed to adapt the wireless Bluetooth technology to legacy and new applications.

A high-level view of the architecture is shown. The responsibilities of the layers in this stack are as follows:



The radio layer is the physical wireless connection. To avoid interference with other devices that communicate in the ISM band, the modulation is based on fast frequency hopping. Bluetooth divides the 2.4 GHz frequency band into 79 channels 1 MHz apart (from 2.402 to 2.480 GHz), and uses this spread spectrum to hop from one channel to another, up to 1600

times a second. The standard wavelength range is 10 cm to 10 m, and can be extended to 100 m by increasing transmission power.



Bluetooth Protocol Stack

- ❖ *The baseband layer is responsible for controlling and sending data packets over the radio link. It provides transmission channels for both data and voice. The baseband layer maintains Synchronous Connection-Oriented (SCO) links for voice and Asynchronous Connectionless (ACL) links for data. SCO packets are never retransmitted but ACL packets are, to ensure data integrity. SCO links are point-to-point symmetric connections, where time slots are reserved to guarantee timely transmission. A slave device is allowed to respond during the time slot immediately following an SCO transmission from the master. A master can support up to three SCO links to a single slave or to multiple slaves, and a single slave can support up to two SCO links to different slaves. Data transmissions on ACL links, on the other hand, are established on a per-slot basis (using slots not reserved for SCO links). ACL links support point-to-multipoint transmissions. After an ACL transmission from the master, only a slave addressed specifically may respond during the next time slot; if no device is addressed, the message is treated as a broadcast.*
- ❖ *The Link Manager Protocol (LMP) uses the links set up by the baseband to establish connections and manage piconets. Responsibilities of the LMP also include authentication and security services, and monitoring of service quality.*
- ❖ *The Host Controller Interface (HCI) is the dividing line between software and hardware. The L2CAP and layers above it are currently implemented in software, and the LMP and lower layers are in hardware. The HCI is the driver interface for the physical bus that connects these two components. The HCI may not be required. The L2CAP may be accessed directly*

by the application, or through certain support protocols provided to ease the burden on application programmers.

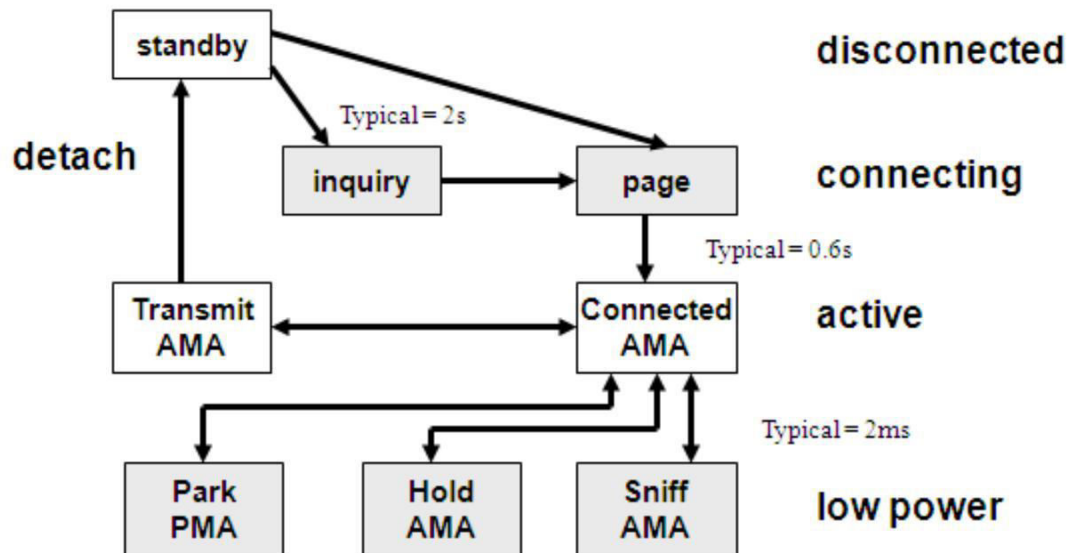
- ❖ *The Logical Link Control and Adaptation Protocol (L2CAP)* receives application data and adapts it to the Bluetooth format. Quality of Service (QoS) parameters are exchanged at this layer.

Link Manager Protocol

The link manager protocol (LMP) manages various aspects of the radio link between a master and a slave and the current parameter setting of the devices. LMP enhances baseband functionality, but higher layers can still directly access the baseband. The following groups of functions are covered by the LMP:

- ❖ **Authentication, pairing, and encryption:** Although basic authentication is handled in the baseband, LMP has to control the exchange of random numbers and signed responses. LMP is not directly involved in the encryption process, but sets the encryption mode (no encryption, point-to-point, or broadcast), key size, and random speed.
- ❖ **Synchronization:** Precise synchronization is of major importance within a Bluetooth network. The clock offset is updated each time a packet is received from the master.
- ❖ **Capability negotiation:** Not only the version of the LMP can be exchanged but also information about the supported features. Not all Bluetooth devices will support all features that are described in the standard, so devices have to agree the usage of, e.g., multi-slot packets, encryption, SCO links, voice encoding, park/sniff/hold mode, HV2/HV3 packets etc.
- ❖ **Quality of service negotiation:** Different parameters control the QoS of a Bluetooth device at these lower layers. The poll interval, i.e., the maximum time between transmissions from a master to a particular slave, controls the latency and transfer capacity. A master can also limit the number of slots available for slaves' answers to increase its own bandwidth.
- ❖ **Power control:** A Bluetooth device can measure the received signal strength. Depending on this signal level the device can direct the sender of the measured signal to increase or decrease its transmit power.
- ❖ **Link supervision:** LMP has to control the activity of a link, it may set up new SCO links, or it may declare the failure of a link.
- ❖ **State and transmission mode change:** Devices might switch the master/slave role, detach themselves from a connection, or change the operating mode

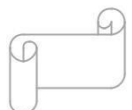
Bluetooth defines several low-power states for a device. The following figure shows the major states of a Bluetooth device and typical transitions. Every device, which is currently not participating in a piconet (and not switched off), is in **standby** mode. This is a low-power mode where only the native clock is running. The next step towards the **inquiry** mode can happen in two different ways. Either a device wants to establish a piconet or a device just wants to listen to see if something is going on.



Major baseband states of a Bluetooth device

- A device wants to establish a piconet: A user of the device wants to scan for other devices in the radio range. The device starts the inquiry procedure by sending an inquiry access code (IAC) that is common to all Bluetooth devices. The IAC is broadcast over 32 so-called wake-up carriers in turn.
- Devices in standby that listen periodically: Devices in standby may enter the inquiry mode periodically to search for IAC messages on the wake-up carriers. As soon as a device detects an inquiry it returns a packet containing its device address and timing information required by the master to initiate a connection. From that moment on, the device acts as slave.

If the inquiry was successful, a device enters the page mode. The inquiry phase is not coordinated, so it may take a while before the inquiry is successful. After a while, a Bluetooth device sees all the devices in its radio range.



During the **page** state two different roles are defined. After finding all required devices the master is able to set up connections to each device, i.e., setting up a piconet. As soon as a device synchronizes to the hopping pattern of the piconet it also enters the connection state. The connection state comprises the active state and the low power states: **park**, **sniff**, and **hold**.

In the **active** state the slave participates in the piconet by listening, transmitting, and receiving. ACL and SCO links can be used. A master periodically synchronizes with all slaves. All devices being active must have the 3-bit **active member address** (AMA). To save battery power, a Bluetooth device can go into one of three low power states:

- **Sniff state:** The sniff state has the highest power consumption of the low power states. Here, the device listens to the piconet at a reduced rate (not on every other slot as is the case in the active state). The interval for listening into the medium can be programmed and is application dependent. The master designates a reduced number of slots for transmission to slaves in sniff state. However, the device keeps its AMA.
- **Hold state:** The device does not release its AMA but stops ACL transmission. A slave may still exchange SCO packets. If there is no activity in the piconet, the slave may either reduce power consumption or participate in another piconet.
- **Park state:** In this state the device has the lowest duty cycle and the lowest power consumption. The device releases its AMA and receives a parked member address (PMA). The device is still a member of the piconet, but gives room for another device to become active (AMA is only 3 bit, PMA 8 bit). Parked devices are still FH synchronized and wake up at certain beacon intervals for re-synchronization. All PDUs sent to parked slaves are broadcast.

L2CAP

The logical link control and adaptation protocol (L2CAP) is a data link control protocol on top of the baseband layer offering logical channels between Bluetooth devices with QoS properties. L2CAP is available for ACLs only.

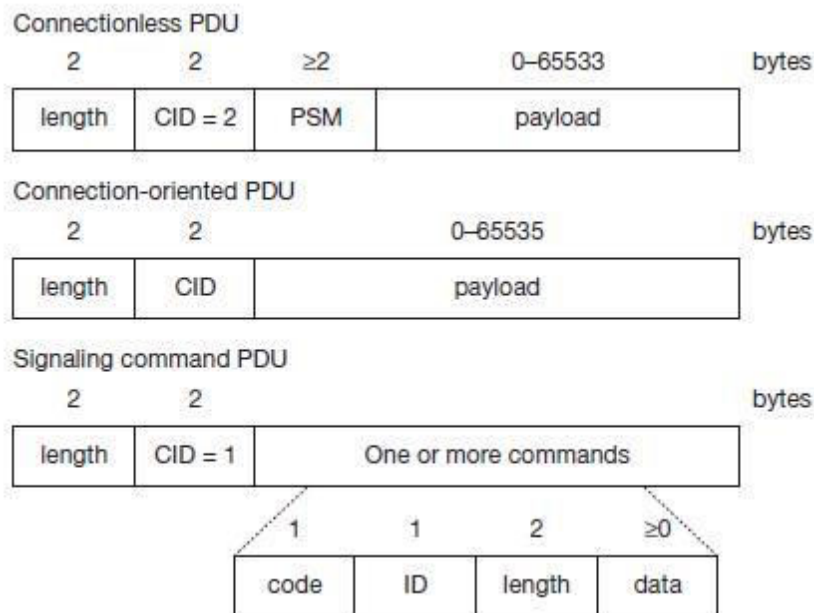
L2CAP provides three different types of logical channels that are transported via the ACL between master and slave:

- ❖ **Connectionless:** These unidirectional channels are typically used for broadcasts from a master to its slave(s).
- ❖ **Connection-oriented:** Each channel of this type is bi-directional and supports QoS flow specifications for each direction. These flow specs follow RFC 1363 and define average/peak data rate, maximum burst size, latency, and jitter.

- ❖ Signaling: This third type of logical channel is used to exchanging signaling messages between L2CAP entities.

Each channel can be identified by its **channel identifier (CID)**. Signaling channels always use a CID value of 1, a CID value of 2 is reserved for connectionless channels. For connection-oriented channels a unique CID (≥ 64) is dynamically assigned at each end of the channel to identify the connection.

The following figure shows the three packet types belonging to the three logical channel types.

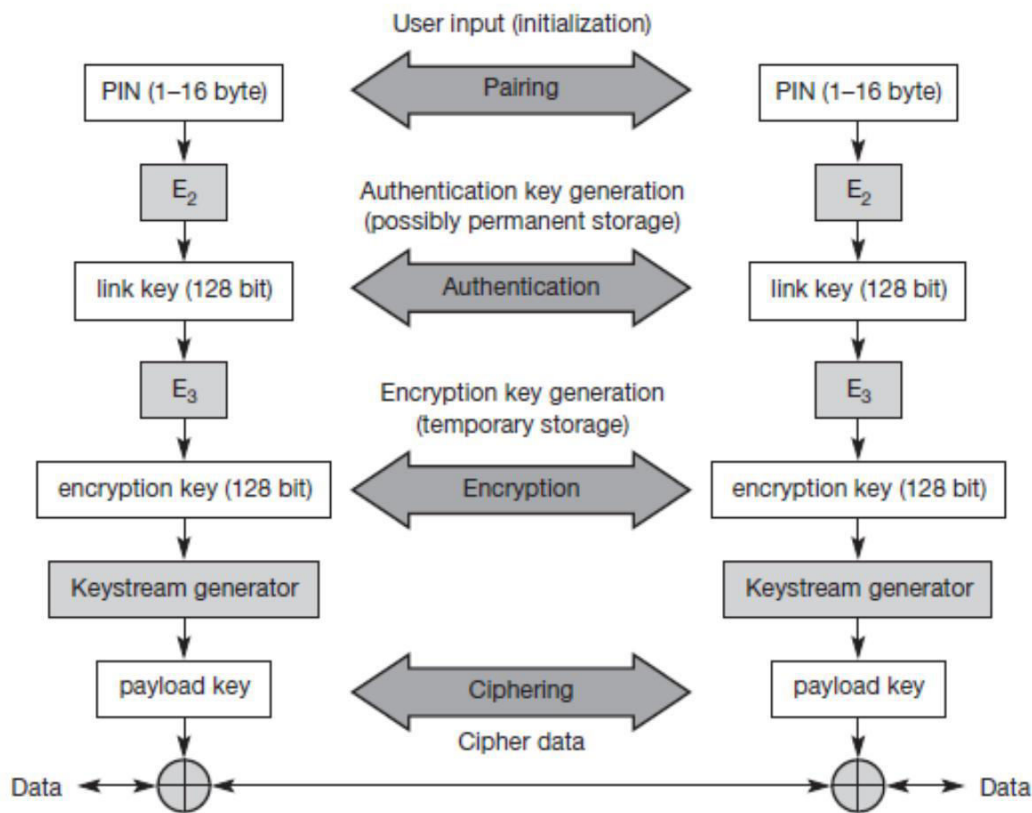


The **length** field indicates the length of the payload (plus PSM for connectionless PDUs). The **CID** has the multiplexing/demultiplexing function. For connectionless PDUs a **protocol/service multiplexor (PSM)** field is needed to identify the higher layer recipient for the payload. For connection-oriented PDUs the CID already fulfills this function. Several PSM values have been defined, e.g., 1 (SDP), 3 (RFCOMM), 5 (TCS-BIN). Values above 4096 can be assigned dynamically. The payload of the signaling PDU contains one or more **commands**. Each command has its own **code** (e.g., for command reject, connection request, disconnection response etc.) and an **ID** that matches a request with its reply. The **length** field indicates the length of the **data** field for this command.

Besides protocol multiplexing, flow specification, and group management, the L2CAP layer also provides segmentation and reassembly functions. Depending on the baseband capabilities, large packets have to be chopped into smaller segments.

Security

The main security features offered by Bluetooth include a challenge response routine for authentication, a stream cipher for encryption, and a session key generation. Each connection may require a one-way, two-way, or no authentication using the challenge-response routine. The security algorithms use the public identity of a device, a secret private user key, and an internally generated random key as input parameters. For each transaction, a new random number is generated on the Bluetooth chip. Key management is left to higher layer software. The following figure shows several steps in the security architecture of Bluetooth.



Bluetooth security components and protocols

The first step, called **pairing**, is necessary if two Bluetooth devices have never met before. To set up trust between the two devices a user can enter a secret PIN into both devices. This PIN can have a length of up to 16 byte. Based on the PIN, the device address, and random numbers, several keys can be computed which can be used as link key for **authentication**. The authentication is a challenge-response process based on the link key, a random number generated by a verifier (the device that requests authentication), and the device address of the claimant (the device that is authenticated).

Based on the link key, and again a random number an encryption key is generated during the **encryption** stage of the security architecture. This key has a maximum size of 128 bits and can be individually generated for each transmission. Based on the encryption key, the device address and the current clock a payload key is generated for ciphering user data. The payload key is a stream of pseudo-random bits. The **ciphering** process is a simple XOR of the user data and the payload key.

All Bluetooth-enabled devices must implement the Generic Access Profile, which contains all the Bluetooth protocols and possible devices. This profile defines a security model that includes three security modes:

- *Mode 1* is an insecure mode of operation. No security procedures are initiated.
- *Mode 2* is known as *service-level enforced security*. When devices operate in this mode, no security procedures are initiated before the channel is established. This mode enables applications to have different access policies and run them in parallel.
- *Mode 3* is known as *link-level enforced security*. In this mode, security procedures are initiated before link setup is complete.

Though Bluetooth offers a better security than WER in 802.11, it has several limitations. The PIN's are often fixed and some keys are permanently stored on the devices. The quality of the random number generators has not been specified.

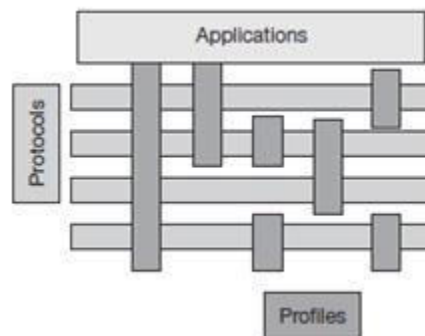
SDP

To find new services available in the radio proximity, Bluetooth defined the **Service Discovery Protocol (SDP)**. SDP defines only the discovery of services, not their usage. Discovered services can be cached and gradual discovery is possible. All the information an SDP server has about a service is contained in a **service record**. This consists of a list of service attributes and is identified by a 32-bit service record handle.

A service attribute consists of an attribute ID and an attribute value. The 16-bit attribute ID distinguishes each service attribute from other service attributes within a service record. The attribute ID also identifies the semantics of the associated attribute value. The attribute value can be an integer, a UUID (universally unique identifier), a string, a Boolean, a URL (uniform resource locator) etc.

Bluetooth Profiles

Bluetooth profiles are intended to ensure interoperability among Bluetooth-enabled devices and applications from different manufacturers and vendors. A profile defines the roles and capabilities for specific types of applications. **Profiles** represent default solutions for a certain usage model. They use a selection of protocols and parameter set to form a basis for interoperability. Protocols can be seen as horizontal layers while profiles are vertical slices as shown below:



The following **basic profiles** have been specified: generic access, service discovery, cordless telephony, intercom, serial port, headset, dialup networking, fax, LAN access, generic object exchange, object push, file transfer, and synchronization. **Additional profiles** are: advanced audio distribution, PAN, audio video remote control, basic printing, basic imaging, extended service discovery, generic audio video distribution, hands-free, and hardcopy cable replacement. Some of the profiles are given below:

- The *Generic Access Profile* defines connection procedures, device discovery, and link management. It also defines procedures related to use of different security models and common format requirements for parameters accessible on the user interface level. At a minimum all Bluetooth devices must support this profile.
- The *Service Discovery Application and Profile* defines the features and procedures for an application in a Bluetooth device to discover services registered in other Bluetooth devices, and retrieves information related to the services.
- The *Serial Port Profile* defines the requirements for Bluetooth devices that need to set up connections that emulate serial cables and use the RFCOMM protocol.
- The *LAN Access Profile* defines how Bluetooth devices can access the services of a LAN using PPP, and shows how PPP mechanisms can be used to form a network consisting of Bluetooth devices.

- The *Synchronization Profile* defines the application requirements for Bluetooth devices that need to synchronize data on two or more devices.

Java 2 Micro Edition (J2ME)

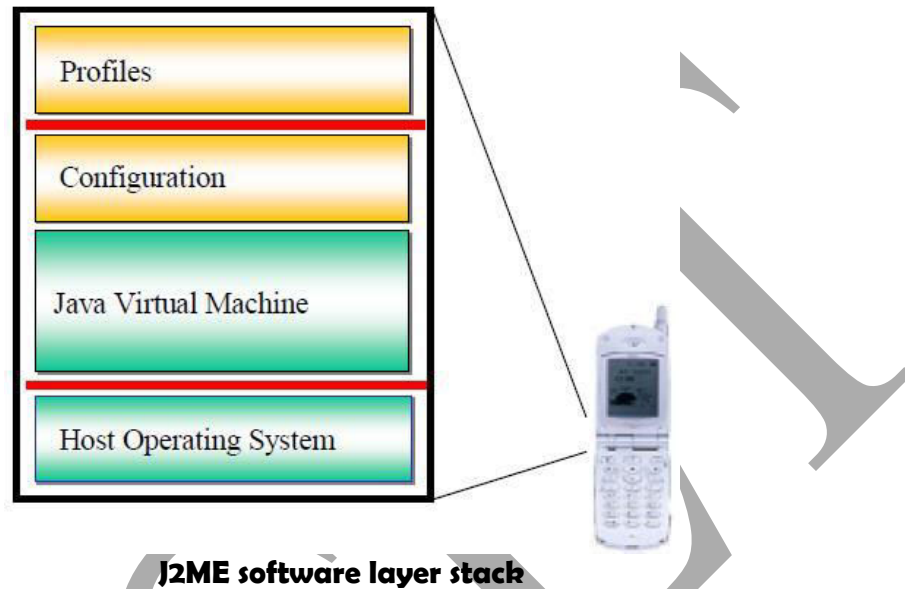
Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. Java 2 Micro Edition maintains the qualities that Java technology has become known for:

- built-in consistency across products in terms of running anywhere, anytime, on any device
- the power of a high-level object-oriented programming language with a large developer base;
- portability of code;
- safe network delivery; and
- upward scalability with J2SE and J2EE

While connected consumer devices such as cell phones, pagers, personal organizers and set-top boxes have many things in common, they are also diverse in form, function and features. Information appliances tend to be special-purpose, limited-function devices. To address this diversity, an essential requirement for J2ME is not only small size but also modularity and customizability. The J2ME architecture is modular and scalable so that it can support the kinds of flexible deployment demanded by the consumer and embedded markets. To support this kind of customizability and extensibility, two essential concepts are defined by J2ME:

- ❖ *Configuration.* A J2ME configuration defines a minimum platform for a "horizontal" category or grouping of devices, each with similar requirements on total memory budget and processing power. A configuration defines the Java language and virtual machine features and minimum class libraries that a device manufacturer or a content provider can expect to be available on all devices of the same category.
- ❖ *Profile.* A J2ME profile is layered on top of (and thus extends) a configuration. A profile addresses the specific demands of a certain "vertical" market segment or device family. The main goal of a profile is to guarantee interoperability within a certain vertical device family or domain by defining a standard Java platform for that market. Profiles typically include

class libraries that are far more domain-specific than the class libraries provided in a configuration.



Configurations

A configuration is a subset of profile. A configuration defines a Java platform for a “horizontal” category or grouping of devices with similar requirements on total memory budget and other hardware capabilities. More specifically, a configuration:

- specifies the Java programming language features supported,
- specifies the Java virtual machine features supported,
- specifies the basic Java libraries and APIs supported.

To avoid fragmentation, there will be a very limited number of J2ME configurations. Currently, the goal is to define two standard J2ME configurations:

➤ **Connected, Limited Device Configuration (CLDC).** The market consisting of personal, mobile, connected information devices is served by the CLDC. This configuration includes some new classes, not drawn from the J2SE APIs, designed specifically to fit the needs of small-footprint devices. It is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications.

➤ **Connected Device Configuration (CDC).** The market consisting of shared, fixed, connected information devices is served by the Connected Device Configuration (CDC). To ensure upward compatibility between configurations, the CDC shall be a superset of the CLDC. This

is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory.



MIDP Mobile Information Device Profile	PDAP Personal Digital Assistant Profile	Personal Profile
		Personal Basis Profile
		Foundation Profile
CLDC Connected, Limited Device Configuration		CDC Connected Device Configuration
J2ME Java 2, Micro Edition		

J2ME Universe

Profiles

The J2ME framework provides the concept of a *profile* to make it possible to define Java platforms for specific vertical markets. Profiles can serve two distinct portability requirements:

- A profile provides a complete toolkit for implementing applications for a particular kind of device, such as a pager, set-top box, cell phone, washing machine, or interactive electronic toy.
- A profile may also be created to support a significant, coherent group of applications that might be hosted on several categories of devices.

Foundation profile contains APIs of J2SE without GUIs. PersonalProfile is profile for embedded devices. Two profiles have been defined for J2ME and are built on CLDC: KJava and Mobile Information Device Profile (MIDP). These profiles are geared toward smaller devices.

MIDP 3.0 is the latest profile version, which is a profile for special-featured phones and handheld devices. It provides improved UI's, UI extensibility and interoperability between the devices. It supports multiple network interfaces in a device, IPv6, large display devices and high performance games. Development tools are used to develop MIDP applications. MIDP applications are composed of two parts:

- JAR File – Contains all of the classes and resources used by the application
- JAD File – Application descriptor, describes how to run the MIDP application

	Source Package	Sets of Java class libraries
1	<i>Java.lang</i>	standard java types and classes for String, Integer, Math, Thread, Security and Exception
2	<i>Java.io</i>	Standard java types and classes for input and output streams
3	<i>Java.util</i>	A set of classes such as Timers, Calenders, Dates, Hashtables, Vectors and others
4	<i>Javax.microedition.rms</i>	A record management system (RMS) API to retrieve and save data and limited querying capability
5	<i>Javax.microedition.pim</i>	Personal information management API (optional), access the device's address book
6	<i>Javax.microedition.pki</i>	Secure connections authenticate API's

MIDP source packages and sets of Java class libraries

K Virtual Machine

The KVM is a compact, portable Java virtual machine specifically designed from the ground up for small, resource-constrained devices. The high-level design goal for the KVM was to create the smallest possible “complete” Java virtual machine that would maintain all the central aspects of the Java programming language, but would run in a resource-constrained device with only a few hundred kilobytes total memory budget. More specifically, the KVM was designed to be:

- small, with a static memory footprint of the virtual machine core in the range of 40 kilobytes to 80 kilobytes (depending on compilation options and the target platform,)
- clean, well-commented, and highly portable,
- modular and customizable, as “complete” and “fast” as possible without sacrificing the other design goals.

The “K” in KVM stands for “kilo.” It was so named because its memory budget is measured in kilobytes (whereas desktop systems are measured in megabytes). KVM is suitable for 16/32-bit RISC/CISC microprocessors with a total memory budget of no more than a few hundred kilobytes (potentially less than 128 kilobytes). This typically applies to digital cellular phones, pagers, personal organizers, and small retail payment terminals.

Mobile Devices and Systems

Handheld Pocket Computers and Mobile System Operating Systems

Handheld Pocket Computers

- Come in many manifestations
- For example, the smart phone
- Pocket-sized PCs
- Differ from smart phones and multimedia phones in that they can be programmed for customized applications
- Offer a variety of application and programming tools not included in new generation mobile phones
- Unlike smart phones, which usually use the text-on-nine-keys format, handheld computers have full text keypad or a touch screen keypad.
- Touch screens or keypad generally used to enter data into handheld devices.
- Some allow the user to write on the screen using a stylus and incorporate special software for handwriting recognition

Handheld Pocket Computers differences with Laptop

- Pocket PCs no CD drives and hard disks
- Use flash memory
- Allow the insertion of a memory stick (A memory stick is a removable flash memory card.)
- Clock speeds of pocket computer processors are limited up to 200 MHz due to considerations about battery life

- Unlike laptops and notebooks, which use regular microcomputer operating systems, pocket computers have specially designed operating systems
- OS scaled to the requirements of the software, hardware, and peripherals used in handheld computers

Windows CE

- An operating system from Microsoft
- Support multitasking on handheld devices
- Real-time operating system meant for handheld computers and embedded systems
- Kernel different from the kernel of the desktop versions of Windows
- Computing devices with low storage and can be run in about 1 MB of memory.
- But the Windows CE OS memory needs are larger as compared to Palm OS
- Support a wider range of hardware than Palm OS.
- Support different CPUs such as NEC MIPS, Intel StrongARM, AMD X86, etc.

Features in Windows CE devices

- Digital camera card
- Games
- Microsoft Windows Media player and other media players

Symbian OS

1. Most widely used operating system for smart phones
2. Runs exclusively on ARM processors
3. Structure much like that of some desktop operating systems
4. Offers pre-emptive multitasking, multithreading
5. Memory protection
6. Initially designed for handheld devices with limited resources, strongly emphasizes on memory conservation
7. Embodies event-based programming and when applications are not directly concerned with events, the CPU is switched off
8. Such techniques are very useful in conserving battery life

Features of a recent version of Symbian OS

- Support for WLAN Hindi and Vietnamese language support to serve a larger range of consumers
- Native support for Wi-Fi
- Support for FOTA (firmware over-the-air)
- Improved memory management
- Low boot-time
- Native support for Push-to-talk

Linux for Mobile Devices

- Linux can be modified easily to suit different sorts of hardware and software applications
- Being an open source OS, it enables the user to customize their device to suit their specific needs
- Considered to be more secure than most other operating systems.

- Linux support is easily available from the many forums and associations that promote this OS

- Many international mobile phone manufacturers turning to Linux for their OS requirements

Android

Android 2.2 Features

- Linux
- Apache Server
- CDMA/EVDO, 802.1x, Updated technology support for CDMA/EVDO, 802.1x
- VPNs (Virtual private networks)
- Function as tethered modem on connecting to USB
- Function as WiFi hotspot functionality
- Softkey pad with auto word prediction; Gestures based UIs, Track multi-touch events
- Virtual keyboard
- Camera, Camcorder, Gallery interface and interface for users to select multiple photos for deletion,
- Built in flash support for Camera and , Digital Zoom
- Animated screen transitions, Voice Search
- Search bookmarks, history, contacts, and the web from the home screen
- WGA screen resolution
- HTML 5 support, Just In Time (JIT) implementation improves speed
- JavaScript engine integrated to browser, fFile upload fields in the Browser
- Google Maps 3.1.2 and, Microsoft Exchange support
- Bluetooth 2.1,; Contact sharing over Bluetooth
- Live Wallpapers
- Application launcher
- Installing applications to the expandable memory
- Voice dialing; Text-to-Speech engine
- Adobe flash 10.1 support