1. Fundamental Concepts

The term "**ubiquitous computing**" was first coined by Mark Weiser (PARC) in his 1991 paper *The Computer for the 21st Century*. He speaks of tabs, pads and boards, and of computers residing in the human world, comparing them to the use of cat's whiskers and subsequently radios (*geek toys* becoming universally accepted).

Since Weiser's paper in 1991, as well as expected advances in processing power and memory, and batteries, other notable developments have occurred or become mainstream such as digital wireless communications, GPS, RFID and the Web, with its increasing image and sound content. Social changes have taken place too, such as the acceptance of video surveillance. Such systems are becoming less important in themselves than the facilities they offer.

Computing has moved from **mainframes** (60s/70s) to **desktops** (80s/90s) to **mobile** devices (90s/00s) to **ubiquitous** (now).

Computers are everywhere; *most* are embedded. However, most are also poorly connected, or unconnected. There is a gap between the physical (real) world and the digital world.

Enabling technologies include: widespread wireless communication; miniaturisation; battery capacity; machine learning (i.e. more "intelligent" software); better, cheaper sensors (light, sound, chemical, electrical, biological, physical); changing interfaces.

Ethical issues: security, trust (*vs.* trustworthiness) and privacy: "privacy is not an absolute good; what matters is making a sensible trade-off between private and shared interests" – Neil Gerschenfeld, MIT Media Lab.

Challenges to overcome: cost, power consumption, technical specifications, robustness, tolerance, security.

2. Service Discovery…

…the act of finding out who can talk to who, to accomplish what. Essentially, dynamic peer-to-peer networking (with more infrastructure for larger networks).

Existing protocols include Jini (Sun, proprietary) and UPnP (Microsoft and others, open), SLP, Salutation and Bluetooth (standards):

Jini is Java-based, from Sun, and is licensed free of charge, and assumes a JVM. Jini is transport independent. At its heart are three protocols *discover*, *join* and *lookup*. It is a *directory-based* service discovery, using RMI and auto-downloading of Java code (in place of device drivers). The code may even *be* the service! Jini *leases* services.

UPnP is Microsoft's distributed architecture that uses HTTP over UDP, and XML for seamless proximity, and assumes IP connectivity. It uses DCHP to automatically obtain an IP address, then automatically announces its capabilities and learns about those of others (*Peer-to-Peer*, non-directory-based service discovery). A discovery message includes a URI pointing to an XML description of a service. UPnP does not have a lightweight authentication protocol and as a result, many devices are shipped with UPnP security turned off.

SLP (Service Location Protocol) is the IETF effort, can work with or without directories. With active discovery, User agents (UA) or Service Agents (SA) send service requests to the Directory Agent (DA) via multicast, which will respond via unicast. Passive discovery, DA multicasts advertisements. With no DA, SAs pick up the service requests directly from UA multicasts, and SAs multicast their own advertisements.

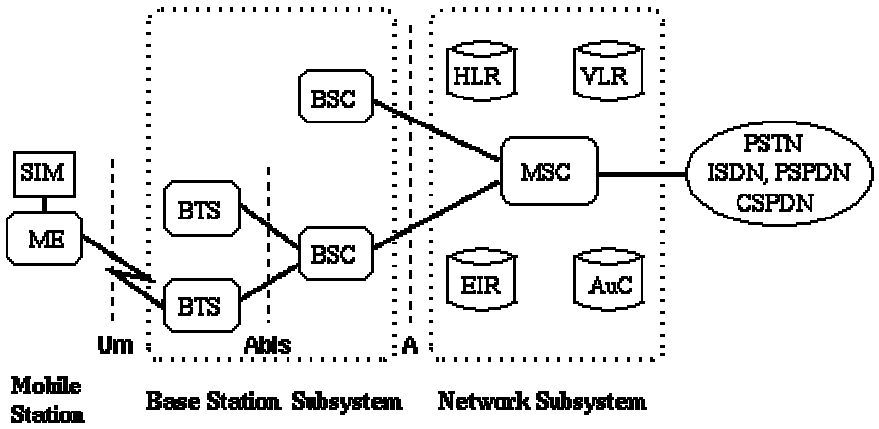
Salutation is produced by a not-for-profit organisation, with royalty-free licenses. It is open-standard, independent of communications protocols and hardware platforms. *Salutation Managers* operate as service brokers (e.g. DA). Transport independent. Also Salutation-Lite for IR & Bluetooth.
Bluetooth is also known as IEEE 802.15.1, whereby one device acts as a master with up to 7 slave devices in a *piconet*. *Peer-to-Peer* service discovery, but no functionality for accessing services (can use SLP or Salutation for this).

There are many steps in discovering and services, and in pervasive computing this is complicated by heterogeneity in hardware, software and protocols, in the highly dynamic and volatile nature of ad-hoc networks, and because providers can be users and vice versa. Printing may be a one-way service, but file-sharing, for example, is not.
Of the many questions that require answering, there are: service naming (format?), initial communications method (broadcast, unicast?), discovery (push or pull?), registration (access control?), service status, selection (manual or automatic?), invocation, usage (leased or continuous? billing?) and termination (implicit or explicit?).

Mapping between protocols becomes important for seamless integration. This is still in its infancy.

3. Mobility (wireless telephony & mobile IP)

Spectrum is crowded and expensive (£22bn UMTS sell-off in UK in April 2000; expected $10~15bn in USA, Summer 2006). Need to make efficient use of bandwidth.
History: Cellular systems with adjacent cells avoiding interference by frequency, time or code division were first developed independently by Bell Labs and MCI (1960s).
1G: (UK:1985) TACS – analogue – just Cellnet & Vodafone, **2G**: (UK:1991) GSM – digital, TDMA/FDMA – (pan-European), **3G**: (UK:2003) UMTS (Global). Throughput improved through multi-frequency and multi-band antennas and CDMA. **SS7** protocols (splitting signals and data), used by PSTNs since 1981 and adopted by mobile telephony and VoIP applications.
Challenges: signal noise; water (rain) absorbs signal; switching cells on the fly; security (digital encryption vs analogue snooping); addressing & routing (mobile IP could have solved this but we're stuck with IPv4); power consumption (harvesting from movement, solar, heat, etc); interface – small, visual, aural; limited storage.



SIM Subscriber Identity Module
ME Mobile Equipment
BTS Base Transceiver Station
BSC Base Station Controller
HLR Home Location Register
VLR Visitor Location Register
MSC Mobile services Switching Center
EIR Equipment Identity Register
AuC Authentication Center

GSM (Global System for Mobile communications) consists of the mobile equipment (handsets and SIMs), base station subsystem (**BSS** –masts and mast controllers), the network switching subsystem (**NSS** – central switching centre, home and roaming registers, and authentication centre) and also the operation and support subsystem (**OSS**), e.g. hierarchical topology:
handset >> mast >> controller >> central unit > public network (PSTN).
GSM supports 4 types of handover: between channels *within* a cell and between cells in the same cluster (both are done locally by the BSC), between cells in different clusters (by the MSC) and between MSCs.
Security is administered by the system, using primarily the SIM but also the equipment id, to allow access and billing to take place. Digital communications can also be **encrypted**, to prevent eavesdropping.
[Summary of GSM at https://styx.uwaterloo.ca/~jscouria/GSM/gsmreport.html]

Mobile IP (first considered in RFC1688, 1994) provides transparent routing of IP datagrams to mobile nodes, i.e. seamless, always-on connection to the Internet.
The **problem**: Existing IP connections require the source and destination IP address/port number to remain constant – invariant – during the lifetime of a connection. This makes it impossible to seamlessly relocate at the IP level. Dynamic DNS can help, or applications may reconnect, but these are not without delay, and the mobile node is not quickly and efficiently locatable.
The **solution**: The **mobile node** will maintain a home-address, which is always accessible, and which is linked to a care-of-address by the **Home Agent**, a router on a mobile node's home network which **tunnels** datagrams for delivery to the mobile node when it is away from home, and maintains current location information for the mobile node. The mobile node will register itself with a **Foreign Agent**, a router on a mobile node's visited network which provides routing services to the mobile node while registered. The foreign agent detunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent. For datagrams sent by a mobile node, the foreign agent may serve as a default router for registered mobile nodes.
Data security is not part of Mobile IP - VPNs provide security when outside the corporate environment.
None of this is magical, but just standardises the mechanism by which seamless connections (from the application perspective) are maintained.
For **IPv6**, Home Agents and encapsulation are still retained, but the need for the Foreign Agent is removed.

4. Location Sensing

There are several **techniques**: **proximity** (e.g. touch, pressure, capacitive field, e.g. touchpads, monitoring transactions, e.g. EPOS, *EZPass*, Oyster), **scene analysis** (can be visual, electromagnetic, etc: ✓ passive; × prior knowledge required, dependant on landmarks not changing), **celestial** (✓ passive; × only works at night in good weather), **triangulation** (using angles, such as the VOR aircraft navigation system), **trilateration** (using distances, as used by GPS), IP address.
Different methods offer different degrees of accuracy.

We can **measure** by **direct** (touch), **attenuation** (strength of signal) or **time-of-flight** (either 1-way or return). These can be affected by weather, obstacles, reflection, etc.
A combination of these methods can be used to generate more accurate results.

Location positions can be physical (e.g. 47ºN 122ºW) or symbolic (e.g. "in Senate House"), absolute (as above) or relative (e.g. 500m at 23º).

Location systems can be defined in many ways: by accuracy and precision (reliability), by scale, by recognition, by cost, and by limitations.

Global Positioning System (**GPS**) is the generic term for location sensing by satellite. Hugely expensive and requiring incredibly accurate atomic clocks (were caesium, now rubidium, will be hydrogen maser), GPS offered a massive improvement over radio-navigation (incl. WWII system LORAN). Ivan Getting established the basis for GPS. [See very good GPS tutorial at http://www.trimble.com/gps]
2-D **location fixing** requires 3 reference points, 3-D requires 4 points. We can use the fact we're 'underneath' satellites to reduce by one point, but add one for resolving timing discrepancies ∴ still use 4 satellites (to determine lat., long., alt. & time).
Navstar/GPS was the world's first GPS system, getting space-borne between 1978 & March 1994, and proving itself during Operation Desert Storm in Kuwait in 1991.
GPS III, a rival to Galileo, is behind schedule, and not expected to begin operation before 2013. Both are 24-satellite systems.
Glonass is the Russian equivalent. Their first satellites were launched in the early 1980s, but not completed until 1995. It fell into decline, and is not fully operational at the present time (only 12 satellites operational; 24 planned by 2011). An agreement has been made with India to provide some launch facilities.
Galileo is the 30-satellite European-funded alternative, due to be running by 2008 and fully operational by 2010, accurate to better than 1m globally. Where GPS and Glonass are military, Galileo is civilian. Galileo will offer greater accuracy, availability and reliability, a faster location fix and an integrity indicator. Galileo's 5 services will include open access (free), commercial (encrypted, and offering QoS), safety of life (e.g. aeronautical), public regulated (e.g. Gov't) and Search & Rescue. GIOVE-A, the first Galileo test satellite, was launched in 2005; GIOVE-B is due to be launched in late 2006, and will carry a hydrogen maser clock, the most accurate clock in space!

A **Global Navigation Satellite System** (GNSS) is being implemented in two stages: GNSS-1 (2004) combines EGNOS (European Geostationary Navigation Overlay Service) and GPS to improve accuracy over Europe and possibly Africa and South America. GNSS-2 (2008) combines Galileo and GPS for sub 1m accuracy.

Differential GPS offers great improvements over existing GPS. By using a fixed receiver with a known location, it can derive timing errors and transmit those to roving receivers to use to correct its measurements. Even better improvements, to sub-centimetre accurary, can be made using *carrier-phase* instead of *code-phase*.

The **Active Bat Location System** uses a similar, but inverse, mechanism to GPS, with multiple ultrasound sensors *above* the tags, which are tracked (as opposed to GPS receivers which are passive, deriving information *from* satellites).
Microsoft's **RADAR** location system uses *scene analysis* in the form of WiFi signal strength measurements to derive location within a building. The **Active Badges** system uses infra-red to signal their proximity to fixed-location sensors (cells).

PlaceLab uses a combination of wireless technologies for location sensing, referencing a cached database. It uses GPS, GSM, WiFi and Bluetooth to sense whatever it can (scene analysis), and is written in Java. PlaceLab is low-cost and ubiquitous, and open-source. Precision is secondary to coverage and cost, and privacy (being passive, as far as is possible). Prior knowledge of beacons (in the database) is necessary for detection of a beacon to be of any use. PlaceLab clients can be *spotters/mappers* (acquiring new information) or *trackers* (using this information to calculate location). PlaceLab works better in urban areas (more beacons) and in Europe (better GSM coverage). Good accuracy from WiFi, better coverage from GSM. Fusing the two produces good results. 100% availability (unlike GPS) is essential

5. Auto-Identification

e.g. **Bar codes** – 2-D passive identification, **RFID** (Radio Frequency Identification), MAC Addresses, IPv6, car number plates, ISBNs.

RFID **History**: Radar showed that objects could be detected using radio waves. The original RFID application was **IFF**, with transponders fitted in aircraft. In 1973 Charles Walton used a passive transponder to unlock a door without a **key**. In the 1970s scientists as Los Alamos devised and developed a vehicle identification system which went to on spawn **automated toll payment systems**. These scientists also developed passive RFID tags encapsulated in glass to be injected and used to **track cattle**. In 1999 the Auto-ID Center was established at MIT to investigate the use of RFID and the internet for **supply chain management**. RFID is increasingly used by large retailers (Wal-Mart, M&S) in asset-tracking and inventory management

The EPC (**Electronic Product Code**) is at the heart of RFID.
EPC is a code (64- or 96-bits), split into 4 fields, version, manufacturer, product, unique item. The Object Naming System (**ONS**), based on DNS, maps an EPC to a URI (a URN namespace, urn:epc:…, was proposed in 2005). Additional data may be stored at the URI location, allowing the tag to be simpler and slower, reducing cost.

EPCglobal Inc. is a not-for-profit company which licenses/regulates RFID at a business level. **Auto-ID Labs** is a not-for-profit research entity which investigates RFID at a technical level (with labs at MIT and Cambridge University).
As cost reduces, mote items will become tagged. The enablers are: cost, wireless comms, passive tags.

The EPCglobal network comprises the EPC itself, tags & readers, middleware, discovery services and EPC information services.
Middleware: Cisco uses routers, Sun uses Java, Oracle uses databases.
Tags can be printed using carbon ink for the antenna.

RFID systems can be *active* (broadcast, longer range ~100m, more expensive) or *passive* (reflective, shorter range, <5m, e.g. Oyster), *near-field* (e.g. Oyster) or *far-field* (e.g. DartTag), *read-only* (e.g. theft prevention) or *writable* (e.g. Oyster).

Applications: **Supply chain management** (can be used at different levels, e.g. pallet, SKU (Stock-Keeping Unit), item e.g. can of soft drink, or individual suit); **Automatic payment** (e.g. EZPass – patented 1977, Oyster, Mobil SpeedPass); **Access control** (since 1973, including remote keyless entry to cars, and anti-theft devices); **Animal tracking** (livestock & pets); **People tracking** (via wristbands, clothing, injectable tags). No longer just 1-bit, on/off (theft prevention) but now identification (product ID) and data (e.g. Oyster).

Privacy & **security** issues: Who *owns* the data in tags? Who can *use* that data? *How* should they be allowed to use that data?
Concerns include **sniffing** (without authentication or encryption, tag data can be easily read), **tracking** (clandestine monitoring), **spoofing** (misrepresentation or cloning of tags), **replay attacks** (record a tag for later use, e.g. road toll avoidance), **denial of service** (blocking or flooding the reading tags).
To counter these concerns, tags can be made to be silent unless reader authenticates (similar to not responding to an ICMP echo request). Lightweight encryption can be used to secure data. Tags can be disabled, either temporarily or permanently.

The **future**: RFID can be used to tag people too, using grain-of-rice-sized tags, inserted into the skin. Primarily used for authentication, identifying surgical patients, etc.
Add an RFID reader to a mobile phone, and with an internet connection you can do price comparisons with products, or initiate online purchases by reading tags embedded into posters!
Add readers to washing machines, fridges, medicine cabinets, etc, and tags to clothes, foodstuffs and medicines, for intelligent machines!

6. MAC protocols, in sensor networks (link layer, single-hop)

MACA, MACAW, PAMAS, IEEE 802.11 DCF (WiFi) (all CSMA), S-MAC (energy efficient),IEEE 802.11 PCF (TDMA) & Bluetooth 802.15.1 (TDMA & frequency hopping)
All deal with wireless contention, sharing bandwidth, minimise energy consumption
MAC **design goals**: ensure *reliable* communications, *maximise* use of bandwidth, ensure *fair* allocation, *minimise* delay, *minimize* energy consumption, accommodate *scalability*. Trade-offs are inevitable (Overall success is measured by performance of the network, not of each node).
CSMA (Carrier sense multiple access) – sense medium, if quiet transmit; if collision back-off (e.g. ALOHAnet, Abramson 1970); **TDMA** (Time division multiple access) – transmit in own time slot; requires good **synchronisation** – scalability issue: hard to achieve in larger networks. Master/Slave, polling.
Exposed node problem: nodes refrain from broadcasting even though no interference may occur (A ← B C ↛ D). C unnecessarily refrains from broadcasting.
Hidden node problem: CSMA does not avoid two nodes colliding when broadcasting to a third (A → B ← C). Both messages are dropped.
MACA – Multiple Access with Collision Avoidance. Sender transmits RTS, received transmits CTS (helps to avoid both hidden & exposed node problems).
MACAW (W='wireless')– adds a DS-Data-ACK handshake. DS (Data Sending) helps back-off for appropriate time.
IEEE 802.11 DCF – standard PHY & MAC layers for WiFi. No DS, but RTS & CTS include 'DS' duration value. All nodes overhearing RTS/CTS can sleep for duration.

PAMAS – Power-Aware Multiple-Access protocol, similar to MACA, but with two channels, control and data, for conserving energy. Turn off data channel unless required.
S-MAC – aims: reduce collisions, reduce unnecessary listening (costly). Reduced energy consumption at expense of delay (low duty cycle, e.g. 10%) and node fairness. Nodes try to adopt the sleep schedule of their neighbours, or raise their cycle. Large messages are fragmented to avoid the need for large retransmissions. **Adaptive-listen** involves waking as soon as a transmission has completed rather than waiting for the next cycle.

S-MAC with adaptive listen is a good choice for sensor networks.

7. (Ad-hoc) routing protocols (network layer, multi-hop)

Ad-hoc network nodes are *mobile*, and can be added, moved or removed *dynamically*.
Two routing methods: **Link-state** & **Distance-vector** (DV).
Two Distance-vector methods: **DSDV** (proactive, good for stable networks with traffic between many modes) and **AODV** (reactive, on demand, good for unstable networks and traffic between few nodes).
Networks are not generally fully-connected (all nodes within range of all other nodes). Short multi-hop paths are preferable (latency, energy, throughput), but for load balancing longer routes *can* be better.
Link-state approach: Centralised. Every node has a complete view of the network topology (not scalable, not good in a dynamic environment, computationally expensive).
Distance-vector approaches:
DSDV (Destination Sequenced DV): Distributed. Each node just knows the cost from itself to all other nodes. Each node maintains a routing table, with destination, next hop, cost and freshness (sequence number) attributes, e.g.:

Destination	Next Hop	Metric	Sequence Number
MH6	MH2	4	S200_MH6

Nodes advertise routing information, to allow topology changes to be propagated. A break to another node raises the cost to infinity, the sequence number is incremented and the node broadcasts this new information.
AODV (Ad-hoc On-demand DV). Similar to DSDV but less traffic, less computation. Routes acquired only on an *as-needed* basis, by instigating a **Route Discovery Process**: *RREQ* message broadcast, *hop count* incremented, until destination is reached, when *RREP* message unicast back along reverse route.

8. Resource-constrained devices (TinyOS)

In a **nutshell**: low-power consumption, low storage capacity, low-range communications, low processing power – everything we don't normally have to worry about! Cost to monitor: light, temp < sound, magnetism < imagery
Trade offs: sensors *vs.* processing *vs.* radio.
Radio costs are relatively high, so duty cycles (on/off) are important. Receiving (listening) is more expensive than transmitting, over short distances.
Special **challenges**: applications to run for months/years without intervention; very limited resources; self-organising networks.
Only have a recent history due to wireless and power source improvements. Moore's Law is producing cheaper & smaller devices, not necessarily more powerful ones.

TinyOS offers a **400 byte** footprint and an **event-based**, **concurrency** model.
nesC is a 'static' language – no kernel or process/memory management – build *app specific OS*, only hardware interrupts (events) & tasks. No task pre-emption (tasks run to completion). This allows for **whole program analysis** and **optimisation** at compile time.
A nesC application consists of a **FIFO scheduler** and a graph (or *wiring together*) of **components**. Components comprise their specification (interface) and their implementation. A component provides and uses interfaces. Interfaces are bidirectional, offering **commands** which may be called and **events** which must be implemented by the caller. **Configurations** *wire* components together, each of which should provide a StdControl interface (init, start, stop). Code can be classed as Synchronous Code (SC) – commands, tasks, and Asynchronous Code (AC) – can be reached from at least one interrupt handler.
Tmote Sky incorporates 802.15.4 radio and a TI microcontroller, 48KB flash, 10KB RAM.
nesC apps are node-centric. The sensor network must exhibit *overall* functionality, e.g. by *aggregating* information, finding *contours* in data, etc. TinyDB facilitates this.
SQL-like queries can be made across the sensor network, e.g.

"SELECT AVG(sound) FROM sensors SAMPLE PERIOD 10s".

Data can be returned from each node (bottleneck at gateway) or aggregated along the way (lose distinct values); data can be *pushed*, proactively, from sensor nodes, or *pulled* (requested) as needed. It depends on the application requirements or type of query.
GAF (Geographic Adaptive Fidelity) algorithm for conserving energy – identifies key routing nodes, allowing other to conserve energy. Sits on top of routing protocol.
Sensor networks come in many forms, e.g. *hierarchical* with *hub* nodes; flatter with multiple *gateway* nodes.
Shipping data out of network may be unnecessary, costly, impractical, impossible. Storing data in network is simple, cheaper.
Example sensor networks: Great Duck Island project determines occupancy of nests; Hawaii Volcanoes NP microclimate, structure integrity, processing plant, water quality monitoring, vehicle tracking, glacial movement monitoring, etc. Contrasting – and harsh – environments.

9. Directed diffusion

Directed diffusion in a sensor network is **data-centric** in that all communication is for named data. A node requests data by sending **interests** for named data. That interest is **diffused** (i.e. broadcast). Directed diffusion differs from TCP-type interaction in that it is not end-to-end, but directed at *regions*. An *interest* message is a query which specifies what a user wants. Data collected by sensor networks can be *events* which are short descriptions of the sensed phenomenon. This data is named using attribute-value pairs. The dissemination of the interest message sets up **gradients** within the network, designed to "draw" events, which flow towards the originators of interest along gradient paths. Tasks, or queries, are also *named* by lists of attribute-value pairs. Selecting a naming scheme is one of the first steps in designing directed diffusion.