La concorrenza è la situazione in cui più processi gestiti da un so vengono eseguiti contemporaneamente, spesso con condivisione di risorse. Questa modalità di esecuzione può portare, se non gestita correttamente, a diversi problemi, come la race condition (situazione in cui la manipolazione delle risorse condivise risente dell’ordine di accesso dei vari processi, col rischio di sovrascrizione o manipolazione errata, non controllata), la starvation (situazione in cui un processo non riesce ad ottenere delle risorse di cui ha bisogno per continuare nella sua esecuzione, nonostante il suo stato di readiness, e rimane quindi bloccato ad un certo punto della sua esecuzione), o il deadlock. Il deadlock è la problematica più complessa: si tratta della situazione in cui ogni processo è bloccato, ed è in attesa di essere sbloccato da un altro processo (dalla sua terminazione o dalla fine del suo turno) che a sua volta è bloccato da un altro processo ancora. Ci sono 3 condizioni necessarie per il deadlock, che sono mutual exclusion, hold & wait, no preemption, ed una condizione necessaria e sufficiente, che è la circular wait (che corrisponde alla presenza di un ciclo nel grafo delle attese dei vari processi).

L’unico modo per evitare il deadlock è avere un so che “vede nel futuro”: dovrebbe infatti sapere, prima dell’inizio dell’esecuzione dei processi, se una delle varie allocazioni delle risorse condivise porta a deadlock. È più facile prevenire un deadlock (tramite una prevenzione indiretta, che evita una delle 3 condizioni necessarie, o diretta, che evita la condizione sufficiente) o trovarlo e risolverlo (uccidendo tutti i processi in deadlock, insieme o uno alla volta, o deallocando tutte le risorse condivise, insieme o una alla volta).

La sincronizzazione è la condizione per cui è garantita la concorrenza con la corretta gestione delle risorse condivise tramite un accesso “protetto” alla sezione critica, cioè ad una porzione di programma accessibile con mutua esclusione forzata: accede un solo processo per volta, tale processo non può bloccarsi in cs, non può andare in deadlock né in starvation, nessun processo da fuori può impedirne l’accesso in cs, e questo accesso avrà un tempo finito ma nessuna condizione sulla durata di esso.

La mutua esclusione forzata si ottiene tramite algoritmi, semafori, block/unblock messages, hardware.

Il message passing è utilizzato sia per la inter process comunication (sia tra processi indipendenti che cooperanti) che per la sincronizzazione: usa le primitive send e receive, che possono contenere o meno l’indirizzo del mittente o del destinatario, e possono essere bloccanti o meno. I canali usati per il message passing sono le pipe (dispositivo logico monodirezionale che fornisce un descrittore per la lettura e uno per la scrittura. Questi descrittori vanno sempre chiusi per non rischiare deadlock. I processi che sfruttano le pipe per comunicare devono essere parenti) e le fifo (dispositivi logici bidirezionali, con due descrittori e apertura bloccante. Non richiede la parentela).

IOT è un sistema distribuito che riguarda l’interconnessione di dispositivi smart (qualsiasi dispositivo connesso ad internet con capacità di comunicazione), oggetti embedded, cioè piccoli dispositivi con poca potenza di calcolo, funzionalità di base, capacità di connessione ad internet per lo scambio di informazioni.

Alcune componenti dei dispositivi: sensori, attuatori, microcontrollori, ricetrasmettitori. Queste servono a raccogliere informazioni dallo spazio circostante, misurarle e trasformarle dall’analogico al digitale, per essere in grado poi di comunicarle via cavo o wireless.

I sistemi operativi in funzione su questo tipo di dispositivi devono avere memoria limitata, supportare hardware eterogeneo e applicazioni realtime, ottimizzati dal punto di vista della gestione dell’energia.

Un sistema distribuito è una tipologia di sistema informatico costituito da un insieme di processi interconnessi tra loro, in esecuzione su un insieme di calcolatori autonomi connessi fra loro tramite una rete che gli permette di coordinare le proprie attività e di condividere le risorse del sistema in modo che gli utenti percepiscano il sistema come un unico servizio, come se fosse proveniente dall’esecuzione di un singolo programma su una singola macchina.

La comunicazione avviene tramite RPC (Remote Procedure Call), cioè un’ estensione del concetto di chiamata di procedura, che consente cioè l’interazione tra procedure di applicazioni diverse oppure che girano su macchine distinte. Le RPC permettono quindi a programmi che girano su macchine distinte di comunicare tramite semplici procedure di call/return. Le RPC possono essere sincrone o asincrone.

Una di queste procedure è il client/server binding (che può essere persistent o non persistent), che consiste nell’apertura di una socket a cui un processo si connette per scambiare informazioni.

TCP e UDP sono due protocolli dello stato di trasporto e si basano sul protocollo IP.

Protocollo IP: è un protocollo per l’indirizzamento di rete e l’instradamento dei pacchetti, senza connessione, con consegna best effort.

Protocollo TCP: è un protocollo orientato alla connessione, con controllo di congestione e di flusso. È più lento dell’UDP ma garantisce la consegna dei segmenti.

Protocollo UDP: è un protocollo non orientato alla connessione, senza garanzia di consegna (non si occupa della ricostruzione della sequenza dei pacchetti ricevuti né della gestione dei duplicati). È usato quando il focus è l’efficienza, più che l’integrità dei dati.